

483

Appendix

485

486

Table of Contents

487

A Summary of Appendices	15
--------------------------------	-----------

488

B LLM Distillation Implementation	15
--	-----------

489

C SPINE	15
----------------	-----------

490

C.1 Action primitives	15
---------------------------------	----

491

C.2 Output format	15
-----------------------------	----

492

C.3 Observation space	16
---------------------------------	----

493

C.4 Tasks	16
---------------------	----

494

D SayCan	17
-----------------	-----------

495

D.1 Action primitives	17
---------------------------------	----

496

D.2 Output format	17
-----------------------------	----

497

D.3 Observation space	17
---------------------------------	----

498

D.4 Tasks	17
---------------------	----

499

E LLM-Planner	17
----------------------	-----------

500

E.1 Action primitives	18
---------------------------------	----

501

E.2 Observation space	19
---------------------------------	----

502

E.3 Output format	19
-----------------------------	----

503

E.4 Tasks	19
---------------------	----

504

F Scenario generation LLM	19
----------------------------------	-----------

505

F.1 SPINE data generator prompt	19
---	----

506

F.2 LLM-Planner data generator prompt	20
---	----

507

F.3 SayCan data generator prompt	21
--	----

508

G Example synthesized environments, tasks, and plans	22
---	-----------

509

G.1 SPINE synthesized data	22
--------------------------------------	----

510

G.2 SPINE elicited plan	24
-----------------------------------	----

511

G.3 LLM Planner synthesized data	24
--	----

512

G.4 LLM Planner elicited plan	25
---	----

513

G.5 SayCan synthesized data	25
---------------------------------------	----

514

G.6 SayCan elicited plan	26
------------------------------------	----

515

516

517

518

A Summary of Appendices

We report the implementation details of our LLM distillation procedure in §E. We then provide further detail on the action primitives, output format, observation space, and evaluation tasks for the three planners used in our evaluation — SPINE (§C), SayCan (§D) and LLM-Planner (§E). We then provide the prompts used for the scenario generation LLM as applied to each planner (§F). Finally, we provide examples of synthesized environments and elicited plans for each planner (§G).

B LLM Distillation Implementation

We use the Unsloth¹ Library, which is built on top of the Huggingface Transformers library and PyTorch. We report hyperparameters used for distilling SLMs via supervised fine tuning in Tab. 4

Parameter	SPINE	SayCan	LLM Planner
learning rate	1e-4	1e-4	2e-4
weight decay	1e-3	1e-3	5e-2
epochs	5	5	5
warmup steps	5	5	5
lora rank	32	32	16
lora alpha	16	16	16

Table 4: SLM hyperparameters

C SPINE

SPINE [9]: The SPINE planner is designed for language-driven mapping, navigation, and exploration in partially-known and unstructured environments. The planner operates in a closed-loop manner: actively mapping its scene and updating its plan based on its findings.

C.1 Action primitives

SPINE composes plans with the following action primitives.

1. `map_region(region_node)`: Go to a region and search for objects.
2. `inspect(object_node, query)`: Inspect an object for attributes described in the query.
3. `explore_region(region_node, exploration_radius)`: Go to a region and explore within a given radius.
4. `goto(region_node)`: goto a region node.
5. `answer(message_to_user)`: Terminate task and provide a message to the user.

C.2 Output format

At each iteration, the LLM provides its primary goal (`primary_goal`) — which is particularly relevant for under-specified tasks — relevant portions of the semantic graph for achieving that goal (`relevant_graph`), its reasoning for its plan, and the plan, all via JSON

```
{"primary_goal": "",  
 "relevant_graph": "",  
 "reasoning": "",  
 "plan": ""}
```

¹<https://unsloth.ai/>

545 C.3 Observation space

546 The observation space is a semantic graph with nodes of type `regions` and `objects` and edges
 547 defined between an object and region (`object_edges`) or two regions (`region_edges`). The initial
 548 observation is provided to the LLM via the JSON format below.

```
{
  "regions": "",
  "objects": "",
  "region_connections": "",
  "object_connections": ""
}
```

549
 550 During an episode, the planner receives updated observations via the graph-
 551 manipulation API: `add_nodes`, `remove_node`, `add_connections`, `remove_connections`,
 552 `update_robot_location`, `update_node_attributes`.

553 C.4 Tasks

554 Experimental tasks for the UAV and UGV are reported in Tab. 5, Tab. 6, and Tab. 7.

Fully-specified Indoor	Under-specified Indoor
Go to the desk	Is the path to the exit clear?
Go to the table	Find a place to sit
Inspect the chair	Find a place to work
Map the area near the desk	I want to watch a movie. Help
Go to the door	I need to do some robotics experiments
Go to the ground_1	Go to the southern part of the scene
Inspect the television	Help me find my backpack. I last saw it when I was working.
Go inspect the table then return to where you started	I lost my laptop. Help.
Inspect the plant then return to where you started	What is near the person?
Inspect the chair then inspect the table	I will be giving a tour of the lab for kids. Is there anything dangerous around?

Table 5: Indoor UGV tasks

Fully-specified Outdoor	Under-specified Outdoor
Map the path	Where can I park my car?
Go to the trees	I want a shady place to read
Inspect the car	Are the chairs under the trees occupied?
Go to sidewalk 2	Is my car damaged?
Go to parking lot 1	Is there debris on the path?
	Is construction blocking the sidewalk?
	I lost my car. Help.
	What is near the person?
	Is the path clear?
	Can I get to the parking lot?

Table 6: Outdoor UGV tasks

Fully-specified	Under-specified
Go to parking lot 1	Is there space to park my car?
Map the building	Is construction blocking the road?
Go to the canine center	Find my dog
Go to the landing strip	Has rain damaged any roofs?
Map the bike rack	Is the playground built?
Return to the landing strip	Return to land

Table 7: UAV tasks

555 D SayCan

556 SayCan [1] is a hierarchical language-guided planner that combines an LLM’s task-level reasoning
557 with lower-level affordance estimates. We follow the tabletop rearrangement environment. Lower
558 level language skill primitives are implemented with CLIPort [59].

559 D.1 Action primitives

560 Actions are specified as `robot.pick_and_place(object, place)`. Objects and places must be
561 listed in the observation. For example, “`robot.pick_and_place(red block, blue bowl)`”.

562 D.2 Output format

563 Plans are sequences of actions that must end with the action “done()”. Unlike the original set-
564 ting where the probability of the model is evaluated for each possible action given the observed
565 environment (i.e. pick and place actions for all combinations of objects) , we simply prompt the
566 LLM to output the next action to accommodate our textual input-output abstraction. For example:
567 “`robot.pick_and_place(red block, blue bowl)`
568 `robot.pick_and_place(blue block, red bowl)`
569 `done()`”

570 D.3 Observation space

571 We list all objects with non-zero affordances as observation inputs, e.g. “`objects = [red block,`
572 `blue bowl]`”. Following [18] affordances are implemented with a ViLD object detector [17].

573 D.4 Tasks

574 The environment is a tabletop setting in the Pybullet simulator [48] with a UR5 robot and randomly
575 generated placings of sets of colored blocks and bowls. We specify 10 tasks each with two different
576 object configurations, with varying levels of difficulty and ambiguity.

577 E LLM-Planner

578 LLM-Planner [16]: The planner queries an LLM with the task specification and an object set de-
579 scription of the observed environment, and 9 in-context demonstrations. The demonstrations are
580 selected via a nearest neighbor search using BERT embeddings from a small set of task-plan expert
581 demonstrations (100 samples from training set). The LLM returns a sequence of semantic sub-goals
582 (e.g., `Navigation fridge`, `OpenObject fridge`). A separate low-level controller then grounds
583 each sub-goal to primitive actions in the partially observed scene.

Task	Objects
Give every bowl an odd number of blocks—or	red/green/blue block, red/blue/green bowls red/blue block, red/blue/green bowls
Put pairs of same-colored blocks into bowls. Leave single blocks.	red/red/blue/green block, red/blue/green bowls red/red/blue/yellow/yellow block, red/blue bowls
Move only the colors that appear exactly once into the yellow bowl, sort all remaining blocks into bowls that match their color.	red/blue/yellow block, yellow bowl red block/red block/blue block/blue block/red/blue/green/yellow bowls
Put the blue stuff together	blue/red/yellow block, blue/red/yellow bowls blue/blue/red/green block, blue/red/green bowls
Place all blocks on top of their matching bowls	red/blue/green block, red/blue/green bowls red/red/blue block, green block/red/blue/green bowls
Separate the blocks by color, putting each color group into a different bowl.	red/blue/green/green block, red/blue/green/yellow bowls red/blue/yellow/yellow block, red/blue bowls
Distribute the blocks as evenly as possible	red/blue/yellow/red block, red/blue bowls red/blue/green/yellow/red block, red/blue bowls
Stack the blocks that aren't red	red/blue/green/yellow block, blue/green/yellow bowls red/red/green block, blue/green bowls
Put blocks in bowls with non-matching colors	red/blue/green/blue/green/yellow bowls red/yellow/green block, blue/green/yellow bowls
Place every block into a bowl, ensuring no block is in a bowl of its own color.	red/blue/green block, red/blue/green/yellow bowls red/blue/green/yellow block, red/blue/green bowls

Table 8: Block–bowl manipulation tasks used for the SayCan planner. Bowl and block positions are randomized.

584 E.1 Action primitives

585 We include seven object interaction actions and a navigation action, as described in [17, Sec. 3.1]
586 and in [17, Sec. 4.1], respectively.

- 587 1. `PickObject` — pick up (grasp) the target object move it into the agent’s inventory.
- 588 2. `PutObject` — place the held object at the current focal location or receptacle (e.g., counter
589 top, table).
- 590 3. `OpenObject` — open an openable object such as a cabinet, drawer, microwave, or door.
- 591 4. `CloseObject` — close an object previously opened by `OpenObject`.
- 592 5. `ToggleOnObject` — switch a toggleable object to its “on” state (e.g., faucet, microwave,
593 stove burner).

- 594 6. `ToggleOffObject` — switch a toggleable object from “on” back to “off”.
- 595 7. `SliceObject` — use a knife currently held to slice a slice-able object (e.g., apple, bread,
- 596 potato) that is specified by the mask.
- 597 8. `Navigation` — discretised egocentric movement and viewpoint-control primitives:
- 598 `MoveAhead`, `RotateLeft/Right`, `LookUp/Down`, which translate or rotate the agent with-
- 599 out interacting with objects.

600 E.2 Observation space

601 The planner receives a string with visible objects, e.g. “Visible objects: fridge, cabinet,

602 countertop”.

603 E.3 Output format

604 Comma-separated string of Action Object pairs, e.g.: “*Navigation microwave, OpenObject mi-*

605 *crowave, PutObject egg microwave, CloseObject microwave, ToggleObjectOn microwave*”

606 E.4 Tasks

607 We consider tasks from Action Learning From Realistic Environments and Directives (AL-

608 FRED) [17]. We evaluate on the seven ALFRED templates:

- 609 • *Clean & Place* (e.g., “Wash the mug and put it in the cabinet”)
- 610 • *Heat & Place* (e.g., “Heat the potato and put it on the counter”)
- 611 • *Cool & Place* (e.g., “Cool the soda and put it on the table”)
- 612 • *Examine-in-Light* (e.g., “Find the keycard and examine it under the lamp”).

613 **Environments** Experiments are run in the *unseen* split of ALFRED — i.e., environments differ

614 from the in-context examples from the training set — comprising eight AI2-THOR [49] homes

615 with kitchens, bedrooms, bathrooms, and living rooms. The agent must explore cluttered, partially

616 observable rooms containing up to 115 object types.

617 F Scenario generation LLM

618 F.1 SPINE data generator prompt

```
You are generating data for training an llm-based planner,
like the SPINE paper from ravichandran et al.

Generate a scene graph for training in the following format <OBSERVATION_FORMAT>
for example, <EXAMPLE_GRAPH>

Make sure all nodes referenced in the connections are listed in the objects and
regions list. Provide your answer in the following JSON format:
<OUTPUT_FORMAT>

Add a "description" attribute to each node that provides information.
These will be hidden from the robot

Task generation instructions
- DO NOT reference specific objects or nodes. Make the planner infer these.
- Tasks should request specific information, not general exploration.
Make the planner map or inspect certain entities.
For example, start tasks with phrases such as "what", "I heard",
"find out", "map", "inspect", "Can I", "is there", and likewise
```

620 The variable `OBSERVATION_FORMAT` takes the form:

628 **E.3 SayCan data generator prompt**

629 We provide the prompt for the LLM-Planner data generator below.

```
You are a task generator for the SayCan pick-and-place environment.
Each task should have a 'raw_input' field describing the goal and a 'config' field specifying 'pick'
↔ and 'place' lists. Output a JSON array of task objects.
Examples:
objects = [red block, yellow block, blue block, green bowl]
# put the yellow one the green thing.
robot.pick_and_place(yellow block, green bowl)
done()

objects = [yellow block, blue block, red block]
# move the light colored block to the middle.
robot.pick_and_place(yellow block, middle)
done()

objects = [blue block, green bowl, red block, yellow bowl, green block]
# stack the blocks.
robot.pick_and_place(green block, blue block)
robot.pick_and_place(red block, green block)
done()

objects = [red block, blue block, green bowl, blue bowl, yellow block, green block]
# group the blue objects together.
robot.pick_and_place(blue block, blue bowl)
done()

objects = [green bowl, red block, green block, red bowl, yellow bowl, yellow block]
# sort all the blocks into their matching color bowls.
robot.pick_and_place(green block, green bowl)
robot.pick_and_place(red block, red bowl)
robot.pick_and_place(yellow block, yellow bowl)
done()

Generate 3 tasks using 4 blocks.
```

630

631 G Example synthesized environments, tasks, and plans

632 G.1 SPINE synthesized data

633 We provide an example synthesized graph and corresponding tasks below. Please note that the
634 output is split to enable readability.

```
{ "graph": {
  "objects": [
    {"name": "bench_1", "coords": [5, 10],
     "description": "A wooden bench facing the ocean."},
    {"name": "bench_2", "coords": [15, 10],
     "description": "A wooden bench near a lamppost."},
    {"name": "lamppost_1", "coords": [10, 12],
     "description": "A tall lamppost providing light at night."},
    {"name": "food_stall_1", "coords": [20, 8],
     "description": "A stall selling ice cream and snacks."},
    {"name": "food_stall_2", "coords": [25, 8],
     "description": "A stall offering hot dogs and beverages."},
    {"name": "kite_1", "coords": [30, 5],
     "description": "A colorful kite flying high in the sky."},
    {"name": "bicycle_1", "coords": [35, 15],
     "description": "A parked bicycle near the storefronts."},
    {"name": "scooter_1", "coords": [40, 15],
     "description": "A parked electric scooter."},
    {"name": "street_performer_1", "coords": [45, 10],
     "description": "A musician playing guitar."},
    "regions": [
      {"name": "boardwalk_1", "coords": [0, 10],
       "description": "The starting point of the boardwalk."},
      {"name": "boardwalk_2", "coords": [10, 10],
       "description": "A section of the boardwalk with benches."},
      {"name": "boardwalk_3", "coords": [20, 10],
       "description": "A section of the boardwalk with food stalls."},
      {"name": "boardwalk_4", "coords": [30, 10],
       "description": "A section of the boardwalk near kite flyers."},
      {"name": "boardwalk_5", "coords": [40, 10],
       "description": "A section of the boardwalk with street performers."},
      {"name": "beach_1", "coords": [0, 0],
       "description": "A sandy beach area with children playing."},
      {"name": "beach_2", "coords": [10, 0],
       "description": "A beach area with people sunbathing."},
      {"name": "beach_3", "coords": [20, 0],
       "description": "A beach area with kite flyers."},
      {"name": "beach_4", "coords": [30, 0],
       "description": "A beach area with volleyball nets."},
      {"name": "beach_5", "coords": [40, 0],
       "description": "A quieter section of the beach."},
      {"name": "storefront_1", "coords": [0, 20],
       "description": "A row of colorful storefronts."},
      {"name": "storefront_2", "coords": [10, 20],
       "description": "A section with souvenir shops."},
      {"name": "storefront_3", "coords": [20, 20],
       "description": "A section with cafes and restaurants."},
      {"name": "storefront_4", "coords": [30, 20],
       "description": "A section with clothing stores."},
      {"name": "storefront_5", "coords": [40, 20],
       "description": "A section with art galleries."},
      {"name": "parking_lot_1", "coords": [50, 15],
       "description": "A small parking lot for cars."},
      {"name": "palm_tree_1", "coords": [5, 15],
       "description": "A tall palm tree providing shade."},
      {"name": "palm_tree_2", "coords": [25, 15],
       "description": "A palm tree near the food stalls."},
      {"name": "palm_tree_3", "coords": [35, 15],
       "description": "A palm tree near the parked bicycles."},
      {"name": "ocean_1", "coords": [0, -5],
       "description": "The ocean with gentle waves."}]
    ]
  }
```

635

```

"object_connections": [
  ["bench_1", "boardwalk_2"],
  ["bench_2", "boardwalk_2"],
  ["lamppost_1", "boardwalk_2"],
  ["food_stall_1", "boardwalk_3"],
  ["food_stall_2", "boardwalk_3"],
  ["kite_1", "beach_3"],
  ["bicycle_1", "storefront_3"],
  ["scooter_1", "storefront_3"],
  ["street_performer_1", "boardwalk_5"]],
"region_connections": [
  ["boardwalk_1", "boardwalk_2"],
  ["boardwalk_2", "boardwalk_3"],
  ["boardwalk_3", "boardwalk_4"],
  ["boardwalk_4", "boardwalk_5"],
  ["beach_1", "beach_2"],
  ["beach_2", "beach_3"],
  ["beach_3", "beach_4"],
  ["beach_4", "beach_5"],
  ["storefront_1", "storefront_2"],
  ["storefront_2", "storefront_3"],
  ["storefront_3", "storefront_4"],
  ["storefront_4", "storefront_5"],
  ["boardwalk_2", "beach_2"],
  ["boardwalk_3", "beach_3"],
  ["boardwalk_4", "beach_4"],
  ["storefront_3", "parking_lot_1"],
  ["boardwalk_1", "storefront_1"],
  ["boardwalk_5", "storefront_5"],
  ["boardwalk_5", "ocean_1"],
  ["beach_5", "ocean_1"]],
"robot_location": "boardwalk_1"
{"tasks": [
  "What is the closest food stall to the starting point of the boardwalk?",
  "I heard there is a musician playing nearby. Can you find out where?",
  "Map the locations of all benches along the boardwalk.",
  "Inspect the area where kites are flying.",
  "Can I find a place to park a bicycle near the storefronts?",
  "Is there a lamppost providing light near any benches?",
  "Find out which section of the beach has volleyball nets.",
  "Map the palm trees along the boardwalk.",
  "What is the closest section of the boardwalk to the ocean?",
  "Inspect the parking lot for available spaces.",
  "Is there a food stall offering beverages nearby?",
  "Find out where the kite flyers are located.",
  "Map the sections of the boardwalk with street performers.",
  "Is there a quieter section of the beach nearby?",
  "Inspect the area with souvenir shops.",
  "What is the closest palm tree to the food stalls?",
  "Find out where the parked scooters are located.",
  "Map the sections of the boardwalk with lampposts.",
  "Is there a section of the beach with children playing?",
  "Inspect the area with cafes and restaurants.",
  "What is the closest section of the boardwalk to the parking lot?",
  "Find out where the art galleries are located.",
  "Map the sections of the beach with sunbathers.",
  "Is there a section of the boardwalk with cyclists?",
  "Inspect the area with clothing stores."
],
"description": "A bustling coastal boardwalk scene flanked by a sandy beach on one side and a row of colorful storefronts on the other. The environment includes a wooden boardwalk populated with scattered benches, lampposts, and occasional food stalls selling snacks and beverages. Close to the boardwalk are cyclists and pedestrians enjoying their time, with children playing and flying kites on the beach. The ocean waves gently lap against the shore, with occasional sailboats in the distance. Behind the storefronts, an array of parked bicycles and scooters lines the street, while a small parking lot accommodates cars. Tall palm trees provide intermittent shade along the boardwalk, and street performers attract crowds. The scene is vibrant and dynamic, with seagulls flying above and people capturing the picturesque views on their cameras."
}

```

637 G.2 SPINE elicited plan

638 We provide the elicited plan with the above semantic graph (§G.1) for the task “Is there a lamppost
639 providing light near any benches?” is below. We preface plans with “// plan” and observations with
640 “// observation”.

```
// plan
{'primary_goal': 'Determine if there is a lamppost
providing light near any benches.',
'relevant_graph': 'boardwalk_1, lamppost_1',
'reasoning': 'I cannot reach lamppost_1 from my current location, boardwalk_1.
I will map boardwalk_1 to try to find a connection.',
'plan': '[map_region(boardwalk_1)]}'

// observation
add_nodes({'boardwalk_2': {'coords': [10.0, 10.0], type: region, description:
A section of the boardwalk with benches., name: boardwalk_2}},
{'storefront_1': {'coords': [0.0, 20.0], type: region, description:
A row of colorful storefronts., name: storefront_1}}),
add_connections([boardwalk_1, boardwalk_2], [boardwalk_1, storefront_1]),
update_node_attributes({'name': boardwalk_1, description:
The starting point of the boardwalk.})

// plan
{'primary_goal': 'Determine if there is a lamppost providing light
near any benches.',
'relevant_graph': 'boardwalk_2, lamppost_1, bench_2',
'reasoning': 'I discovered a new region, boardwalk_2, which contains benches.
I will navigate to boardwalk_2 and explore the region to
see if there is a lamppost providing light near the benches.',
'plan': '[goto(boardwalk_2), explore_region(boardwalk_2, 3)]}'

// observation
update_robot_location(boardwalk_2)

// plan
{'primary_goal': 'Determine if there is a lamppost providing light near any
benches.',
'relevant_graph': 'boardwalk_2, lamppost_1, bench_2',
'reasoning': 'I have reached boardwalk_2, which contains benches.
I will now explore the region to see
if there is a lamppost providing light near the benches.',
'plan': '[explore_region(boardwalk_2, 3)]}'

// observation
add_nodes({'bench_1': {'coords': [5.0, 10.0], type: object,
description: A wooden bench facing the ocean., name: bench_1}},
{'boardwalk_3': {'coords': [20.0, 10.0], type: region,
description: A section of the boardwalk with food stalls., name: boardwalk_3}}),
add_connections([bench_1, boardwalk_2], [bench_2, boardwalk_2],
[boardwalk_2, lamppost_1], [boardwalk_2, boardwalk_3],
[bench_2, boardwalk_2]),
update_node_attributes({'name': boardwalk_2,
description: A section of the boardwalk with benches.})

// plan
{'primary_goal': 'Determine if there is a lamppost
providing light near any benches.',
'relevant_graph': 'boardwalk_2, lamppost_1, bench_1, bench_2',
'reasoning': 'I discovered that boardwalk_2 has connections to both
bench_1 and bench_2, as well as lamppost_1. This indicates that lamppost_1
is near the benches and likely provides light to them.',
'plan': '[answer(Yes, there is a lamppost, lamppost_1,
providing light near the benches, bench_1 and bench_2, on boardwalk_2.)]}'
```

642 G.3 LLM Planner synthesized data

643 We provide an example synthesized object set and corresponding tasks below.

```

{
  "Task description": "Navigate to the DiningTable and use the Knife to slice the bread. Then, place
↳ the sliced pieces onto a Plate.",
  "Completed plans": [],
  "Visible objects": [
    "DiningTable",
    "Bread",
    "Knife",
    "Plate",
    "AlarmClock",
    "Desk",
    "Bed",
    "Fridge",
    "Egg",
    "CounterTop",
    "CoffeeMachine"
  ],
  "Next Plans": [
    "Navigation DiningTable",
    "PickupObject Knife",
    "SliceObject Bread",
    "PutObject Knife DiningTable",
    "PickupObject Bread",
    "Navigation Plate",
    "PutObject Bread Plate"
  ]
}

```

644

645 G.4 LLM Planner elicited plan

646 We provide an example elicited plan below.

```

{
  "trial": "trial_T20190908_145356_918528",
  "scene": "FloorPlan10",
  "type": "pick_heat_then_place_in_recep",
  "repeat_idx": 1,
  "goal_instr": "Put a heated apple in the trash can.",
  "initial_high_level_plans": [
    "Navigation apple",
    "PickupObject apple",
    "Navigation microwave",
    "OpenObject microwave",
    "PutObject apple microwave",
    "CloseObject microwave",
    "ToggleObjectOn microwave",
    "ToggleObjectOff microwave",
    "OpenObject microwave",
    "PickupObject apple",
    "CloseObject microwave",
    "Navigation garbagecan",
    "PutObject apple garbagecan"
  ],
  "completed_plans": [
    "Navigation apple",
    "PickupObject apple",
    "Navigation microwave",
    "OpenObject microwave",
    "PutObject apple microwave",
    "CloseObject microwave",
    "ToggleObjectOn microwave",
    "ToggleObjectOff microwave",
    "OpenObject microwave",
    "PickupObject apple",
    "CloseObject microwave",
    "Navigation garbagecan",
    "PutObject apple garbagecan"
  ],
  "failed_plans": [],
  "success": true
}

```

647

648 G.5 SayCan synthesized data

649 We provide an example synthesized object set below.

650

```
objects = [red block, green block, blue block, red bowl, green bowl, blue bowl]  
# Place all blocks on top of their matching bowls
```

651 **G.6 SayCan elicited plan**

652 We provide an example elicited plan below.

653

```
objects = [red block, green block, blue block, red bowl, green bowl, blue bowl]  
# Place all blocks on top of their matching bowls  
robot.pick_and_place(red block, red bowl)  
robot.pick_and_place(blue block, blue bowl)  
robot.pick_and_place(green block, green bowl)  
done()
```

654