A Overview

Compared to the original implementation [1], our implementation benefits from repaired assets (Sec B), improved reward functions and better training schemes (Sec C). Other differences include observation and action spaces. We introduce in observations the target positions in the base frame in addition to those in the end-effector frame. The arm action is defined in the joint configuration space (7-dim) rather than the end-effector Euclidean space (3-dim with no orientation).

B Dataset and Episodes

[1] keeps updating the ReplicaCAD dataset. The major fix is "minor furniture layout modifications in order to better accommodate robot access to the full set of receptacles" [5]. The agent radius is also decreased from 0.4m to 0.3m to generate navigation meshes with higher connectivity. Besides, [1] also improves the episode generator [6] to ensure stable initialization of objects. Those improvements eliminate most unachievable episodes in the initial version. The episodes used in our experiments are generated with the ReplicaCAD v1.4 and the latest habitat-lab [7].

For *TidyHouse*, each episode includes 20 clutter objects and 5 target objects along with their goal positions, located at 7 different receptacles (chair, 2 tables, tv stand, two kitchen counters, sofa). For *PrepareGroceries*, each episode includes 21 clutter objects located at 8 different receptacles (the 7 receptacles used in *TidyHouse* and the top shelf of the fridge) and 1 clutter object located at the middle shelf of the fridge. 2 target objects are located at the middle shelf, and each of their goal positions is located at one of two kitchen counters. The third target object is located at one of two kitchen counters, and its goal position is at the middle shelf. *SetTable* generates episodes similar to *PrepareGroceries*, except that two target objects, bowl and apple, are initialized at one of 3 drawers and at the middle fridge shelf respectively. Each of their goal positions is located at one of two tables.

C Skill Learning

Each skill is trained to accomplish a subtask and reset its end-effector at the resting position. The robot arm is first initialized with predefined resting joint positions, such that the corresponding resting position of the end-effector is $(0.5, 1.0, 0.0)$ in the base frame [8]. The initial end-effector position is then perturbed by a Gaussian noise $\mathcal{N}(0, 0.025)$ clipped at $0.05m$. The base position is perturbed by a Gaussian noise $\mathcal{N}(0, 0.1)$ truncated at $0.2m$. The base orientation is perturbed by a Gaussian noise $\mathcal{N}(0, 0.25)$ truncated at 0.5 radian. The maximum episode length is 200 steps for all the manipulation skills, and 500 steps for the navigation skill. The episode terminates on success or failure. We use the same reward function for both stationary and mobile manipulation skills, unless specified.

For all skills, d^o_{ee} is the distance between the end-effector and the object, d^r_{ee} is the distance between the end-effector and the resting position, d^h_{ee} is the distance between the end-effector and a predefined manipulation handle (a 3D position) of the articulated object, d^q_a is the distance between the joint position of the articulated object and the goal joint position. $\Delta^b_a = d^b_a(t-1) - d^b_a(t)$ stands for the (negative) change in distance between a and b. For example, Δ^o_{ee} is the change in distance between the end-effector and the object. $\mathbb{I}_{holding}$ indicates if the robot is holding an (correct) object or handle. $\mathbb{I}_{succ}$ indicates the task success. C_t refers to the current collision force, and $C_{1:t}$ stands for the accumulated collision force.

The 7-dim arm action stands for the delta joint positions added to the current target joint positions of the PD controller. The input arm action is assumed to be normalized to $[-1, 1]$, and will be scaled by 0.025 (radian). The 2-dim base action stands for linear and angular velocities. The base movement in the Habitat 2.0 is implemented by kinematically setting the robot's base transformation. The collision between the robot base and navigation meshes is taken into consideration. The input base action is assumed to be normalized to $[-1, 1]$, and will be scaled by 3 (navigation skill) or 1.5 (manipulation skills). For the navigation skill, we follow [1] to use a discrete action space and

[5] `https://github.com/facebookresearch/habitat-sim/pull/1694`

[6] `https://github.com/facebookresearch/habitat-lab/pull/764`

[7] `https://github.com/facebookresearch/habitat-lab/pull/837`

[8] The positive x and y axes point forward and upward in Habitat.

translate the discrete action into the continuous one. Concretely, the (normalized) linear velocity from -0.5 to 1 is discretized into 4 choices ($\{-0.5, 0, 0.5, 1\}$), and the (normalized) angular velocity from -1 to 1 is discretized into 5 choices (($\{-1, -0.5, 0, 0.5, 1\}$). The stop action corresponds to the discrete action representing zero velocities.

Pick(s_0)

- Objective: pick the object initialized at s_0
- Initial base position (noise is applied in addition):
  - Stationary: the closest navigable position to s_0
  - Mobile: a randomly selected navigable position within 2m of s_0
- Reward: $\mathbb{I}_{pick}$ indicates whether the correct object is picked and $\mathbb{I}_{wrong}$ indicates whether a wrong object is picked.

$$r_t = 4\Delta^o_{ee}\mathbb{I}_{!holding} + \mathbb{I}_{pick} + 4\Delta^r_{ee}\mathbb{I}_{holding} + 2.5\mathbb{I}_{succ}$$
$$- \max(0.001C_t, 0.2) - \mathbb{I}_{[C_{1:t}>5000]} - \mathbb{I}_{wrong} - \mathbb{I}_{[d^o_{ee}>0.09]}\mathbb{I}_{holding} - 0.002$$

- Success: The robot is holding the target object and the end-effector is within 5cm of the resting position. $\mathbb{I}_{succ} = \mathbb{I}_{holding} \wedge d^r_{ee} \leq 0.05$
- Failure:
  - $\mathbb{I}_{[C_{1:t}>5000]} = 1$: The accumulated collision force is larger than $5000N$.
  - $\mathbb{I}_{wrong} = 1$: A wrong object is picked.
  - $\mathbb{I}_{[d^o_{ee}>0.09]}\mathbb{I}_{holding} = 1$: The held object slides off the gripper.
- Observation space:
  - Depth images from head and arm cameras.
  - The current arm joint positions.
  - The current end-effector position in the base frame.
  - Whether the gripper is holding anything.
  - The starting position s_0 in both the base and end-effector frame.
- Action space: The gripper is disabled to release.

Place(s_*)

- Objective: place the held object at s_*
- Initial base position (noise is applied in addition):
  - Stationary: the closest navigable position to s_*
  - Mobile: a randomly selected navigable position within 2m of s_*
- Reward: $\mathbb{I}_{place}$ indicates whether the object is released within 15cm of the goal position, and $\mathbb{I}_{drop}$ indicates whether the object is released beyond 15cm.

$$r_t = 4\Delta^{s_*}_{o}\mathbb{I}_{holding} + \mathbb{I}_{place} + 4\Delta^r_{ee}\mathbb{I}_{!holding} + 2.5\mathbb{I}_{succ}$$
$$- \min(0.001C_t, 0.2) - \mathbb{I}_{[C_{1:t}>7500]} - \mathbb{I}_{drop} - \mathbb{I}_{[d^o_{ee}>0.09]}\mathbb{I}_{holding} - 0.002$$

- Success: The object is within 15cm of the goal position and the end-effector is within 5cm of the resting position. $\mathbb{I}_{succ} = d^{s_*}_{o} \leq 0.15 \wedge \mathbb{I}_{!holding} \wedge d^r_{ee} \leq 0.05$
- Failure:
  - $\mathbb{I}_{[C_{1:t}>7500]} = 1$: The accumulated collision force is larger than $7500N$.
  - $\mathbb{I}_{drop} = 1$: The object is released beyond 15cm of the goal position.
  - $\mathbb{I}_{[d^o_{ee}>0.09]}\mathbb{I}_{holding} = 1$: The held object slides off the gripper.
- Observation space:
  - Depth images from head and arm cameras.
  - The current arm joint positions.
  - The current end-effector position in the base frame.
  - Whether the gripper is holding anything.
  - The goal position s_* in both the base and end-effector frame.
- Action space: The gripper is disabled to grasp after releasing the object.

Open drawer(s)

- Objective: open the drawer containing the object initialized at s. The goal joint position of the drawer is $g = 0.45m$.
- Initial base position (noise is applied in addition):
  - Stationary: a navigable position randomly selected within a $[0.80, -0.35] \times [0.95, 0.35]$ region in front of the drawer.
  - Mobile: a navigable position randomly selected within a $[0.3, -0.6] \times [1.5, 0.6]$ region in front of the drawer.
- Reward: $\mathbb{I}_{open} = d_a^g \leq 0.05$ indicates whether the drawer is open. $\mathbb{I}_{release}$ indicates whether the handle is released when the drawer is open. $\mathbb{I}_{grasp}$ indicates whether the correct handle is grasped. a_{base} is the (2-dim) base action.

$$r_t = 2\Delta_{ee}^h \mathbb{I}_{!open} + \mathbb{I}_{grasp} + 2\Delta_a^g \mathbb{I}_{holding} + \mathbb{I}_{release} + 2\Delta_{ee}^r \mathbb{I}_{open} + 2.5\mathbb{I}_{succ} \\ -\mathbb{I}_{wrong} - \mathbb{I}_{[d_{ee}^h > 0.2]} \mathbb{I}_{holding} - \mathbb{I}_{out} - 0.004\|a_{base}\|_1$$

- Success: The drawer is open, and the end-effector is within 15cm of the resting position. $\mathbb{I}_{succ} = \mathbb{I}_{open} \wedge \mathbb{I}_{!holding} \wedge d_{ee}^r \leq 0.15$
- Failure:
  - $\mathbb{I}_{wrong} = 1$: The wrong object or handle is picked.
  - $\mathbb{I}_{[d_{ee}^h > 0.2]} \mathbb{I}_{holding} = 1$: The grasped handle slides off the gripper.
  - $\mathbb{I}_{out} = 1$: The robot moves out of a predefined region (a $2m \times 3m$ region in front of the drawer).
  - $\mathbb{I}_{\mathbb{I}_{[open(t-1) \wedge !open(t)]}} = 1$: The drawer is not open after being opened.
  - The gripper releases the handle when the drawer is not open ($\mathbb{I}_{!open} = 1$).
  - $\Delta_a^g >= 0.1$: The drawer is opened too fast.
- Observation space:
  - Depth images from head and arm cameras.
  - The current arm joint positions.
  - The current end-effector position in the base frame.
  - Whether the gripper is holding anything.
  - The starting position s in both the base and end-effector frame.

Close drawer(s)

- Objective: close the drawer containing the object initialized at s. The goal joint position is $g = 0m$.
- Initial joint position: $q_a \in [0.4, 0.5]$, where q_a is the joint position of the target drawer. A random subset of other drawers are slightly open ($q_a' \leq 0.1$).
- Initial base position (noise is applied in addition):
  - Stationary: a navigable position randomly selected within a $[0.3, -0.35] \times [0.45, 0.35]$ region in front of the drawer.
  - Mobile: a navigable position randomly selected within a $[0.3, -0.6] \times [1.0, 0.6]$ region in front of the drawer.
- Reward: It is almost the same as *Open drawer* by replacing *open* with *close*. $\mathbb{I}_{close} = d_a^g \leq 0.1$.
- Success: The drawer is closed, and the end-effector is within 15cm of the resting position.
- Failure: It is almost the same as *Open drawer* by replacing *open* with *close*, except that the last constraint $\Delta_a^g >= 0.1$ is not included.

Open fridge(s)

- Objective: open the fridge containing the object initialized at s. The goal joint position is $g = \frac{\pi}{2}$.
- Initial base position (noise is applied in addition): a navigable position randomly selected within a $[0.933, -1.5] \times [1.833, 1.5]$ region in front of the fridge.

- Reward: $\mathbb{I}_{open} = g - q_a > 0.15$, where q_a is the joint position (radian) of the fridge. To avoid the robot from penetrating the fridge due to simulation defects, we add a collision penalty but excludes collision between the end-effector and the fridge.

$$r_t = 2\Delta^h_{ee}\mathbb{I}_{!open} + \mathbb{I}_{grasp} + +2\Delta^g_a\mathbb{I}_{holding} + \mathbb{I}_{release} + \Delta^r_{ee}\mathbb{I}_{open} + 2.5\mathbb{I}_{succ} \\ -\mathbb{I}_{C_{1:t}>5000} - \mathbb{I}_{wrong} - \mathbb{I}_{[d^h_{ee}>0.2]}\mathbb{I}_{holding} - \mathbb{I}_{out} - 0.004\|a_{base}\|_1$$

- Success: The fridge is open, and the end-effector is within 15cm of the resting position. $\mathbb{I}_{succ} = \mathbb{I}_{open} \wedge \mathbb{I}_{!holding} \wedge d^r_{ee} \leq 0.15$
- Failure:
  - $\mathbb{I}_{wrong} = 1$: The wrong object or handle is picked.
  - $\mathbb{I}_{[d^h_{ee}>0.2]}\mathbb{I}_{holding} = 1$: The grasped handle slides off the gripper.
  - $\mathbb{I}_{out} = 1$: The robot moves out of a predefined region (a $2m \times 3.2m$ region in front of the fridge).
  - $\mathbb{I}_{\mathbb{I}_{[open(t-1)\wedge!open(t)]}} = 1$: The fridge is not open after being opened.
  - The gripper releases the handle when the fridge is not open ($\mathbb{I}_{!open} = 1$).
- Observation space:
  - Depth images from head and arm cameras.
  - The current arm joint positions.
  - The current end-effector position in the base frame.
  - Whether the gripper is holding anything.
  - The starting position s in both the base and end-effector frame.

Close fridge(s)

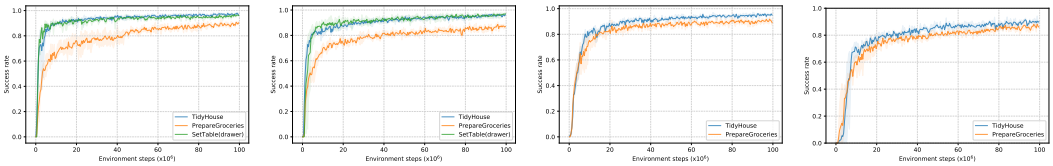
- Objective: close the fridge containing the object initialized at s. The goal joint position is $g = 0$.
- Initial joint position: $q_a \in [\frac{\pi}{2} - 0.15, 2.356]$, where q_a is the joint position of the target fridge.
- Initial base position (noise is applied in addition): a navigable position randomly selected within a $[0.933, -1.5] \times [1.833, 1.5]$ region in front of the fridge.
- Reward: It is almost the same as *Close fridge* by replacing *open* with *close*. $\mathbb{I}_{close} = d^g_a \leq 0.15$.
- Success: The fridge is close, and the end-effector is within 15cm of the resting position.

Navigate(s) (point-goal)

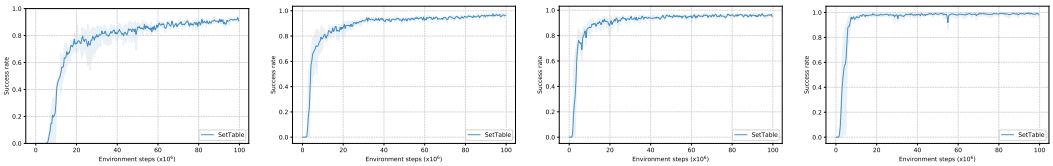
- Objective: navigate to the start of other skills specified by s
- Reward: refer to Eq 1. $r_{slack} = 0.002, \tilde{D} = 0.9, \lambda_{ang} = 0.25, \lambda_{succ} = 2.5$
- Success: The robot is within 0.3 meter of the goal, 0.5 radian of the target orientation, and has called the stop action at the current time step.
- Observation space:
  - Depth images from the head camera.
  - The goal position s_* in the base frame.

Navigate(s) (region-goal)

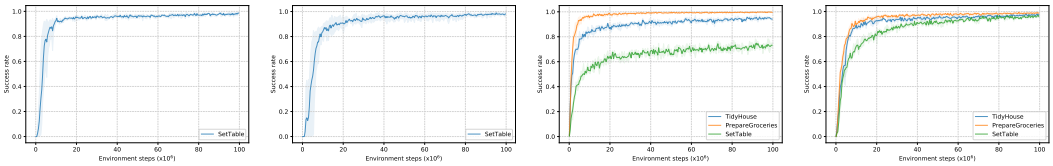
- Objective: navigate to the start of other skills specified by s
- Reward: refer to Eq 2. $r_{slack} = 0.002, r_{col} = \min(0.001C_t, 0.2), \lambda_{succ} = 2.5$
- Success: The robot is within 0.1 meter of any goal in the region, 0.25 radian of the target orientation at the current position, and has called the stop action at the current time step.
- Observation space:
  - Depth images from the head camera.
  - The goal position s_* in the base frame.



(a) Pick(stationary) (b) Pick(mobile) (c) Place(stationary) (d) Place(mobile)



(e) Open drawer(stationary) (f) Open drawer(mobile) (g) Close drawer(stationary) (h) Close drawer(mobile)



(i) Open fridge (j) Close fridge (k) Navigation (point-goal) (l) Navigation (region-goal)

Figure 6: Training curves for skills. The y-axis represents the success rate of the subtask (including resetting the end-effector at its resting position). Best viewed zoomed.

C.1 PPO Hyper-parameters

Our PPO implementation is based on the habitat-lab. The visual encoder is a simple CNN [9]. The coefficients of value and entropy losses are 0.5 and 0 respectively. We use 64 parallel environments and collect 128 transitions per environment to update the networks. We use 2 mini-batches, 2 epochs per update, and a clipping parameter of 0.2 for both policy and value. The gradient norm is clipped at 0.5. We use the Adam optimizer with a learning rate of 0.0003. The linear learning rate decay is enabled. The mean of the Gaussian action predicted by the policy network is activated by $\tanh$. The (log) standard deviation of the Gaussian action, which is an input-independent parameter, is initialized as -1.0. Fig 6 shows training curves of skills.

C.2 Other Implementation Details

The PPO algorithm implemented by the habitat-lab does not distinguish the termination of the environment (MDP) and the truncation due to time limit. We fix this issue in our implementation. Furthermore, we separately train all the skills for each HAB task to avoid potential ambiguity. For example, the starting position of an object in the drawer is computed when the drawer is closed at the beginning of an episode. However, the skill *Pick* needs to pick this object up when the drawer is open and the actual position of the object is different from the starting position. It is inconsistent with other cases when the object is in an open receptacle or the fridge. We observe such ambiguity can hurt performance. See Fig 6 for all task-specific variants of skills.

D Monolithic Baseline

For the monolithic baseline, a monolithic RL policy is trained for each HAB task. During training, the policy only handles one randomly selected target object, *e.g.*, picking and placing one object in *TidyHouse*. During inference, the policy is applied to each target object. We use the same observation space, action space and training scheme as those for our mobile manipulation skills. The main

[9] `https://github.com/facebookresearch/habitat-lab/blob/main/habitat_baselines/rl/models/simple_cnn.py`

challenge is how to formulate a reward function for those complicated long-horizon HAB tasks that usually require multiple stages. We follow [1] to composite reward functions for individual skills, given the sequence of subtasks. Concretely, at each time step during training, we infer the current subtask given perfect knowledge of the environment, and use the reward function of the corresponding skill. To ease training, we remove the collision penalty and do not terminate the episode due to collision. Besides, we use the region-goal navigation reward for the navigation subtask. Thanks to our improved reward functions and better training scheme, our monolithic RL baseline is much better than the original implementation in [1]. However, although able to move the object to its goal position, the policy never learns to release the object to complete the subtask *Place* during training. It might be due to exploration difficulty since *Place* is the last subtask in a long sequence and previous subtasks all require the robot not to release. To boost its performance, we force the gripper to release anything held at the end of execution during evaluation.

E Evaluation

E.1 Sequential Skill Chaining

For evaluation, skills are sequentially executed in the order of their corresponding subtasks, as described in Sec 3.3. The main challenge is how to terminate a skill without privileged information. Basically, each skill will be terminated if its execution time exceeds its max episode length (200 steps for manipulation skills and 500 steps for the navigation skill). The termination condition of *Pick* is that an object is held and the end-effector is within 15cm of the resting position, which can be computed based on proprioception only. The gripper is disabled to release for *Pick*. The termination condition of *Place* is that the gripper holds nothing and the end-effector is within 15cm of the resting position. The gripper is disabled to grasp for *Place*. Besides, anything held will be released when *Place* terminates. For *Open* and *Close*, we use a heuristic from [1]: the skill will terminate if the end-effector is within 15cm of the resting position and it has moved at least 30cm away from the resting position during execution. *Navigate* terminates when it calls the stop action. Furthermore, since the manipulation skills only learn to reset its end-effector, we apply an additional operation to reset the whole arm after each skill. This reset operation is achieved by setting predefined joint positions as the target of the robot's PD controller.

E.2 Progressive Completion Rate

In this section, we describe how progressive completion rates are computed. The evaluation protocol is the same as [1] (see its Appendix F), and here we phrase it in a way more friendly to readers with little knowledge of task planning and Planning Domain Definition Language (PDDL). To partially evaluate a HAB task, we divide a full task into a sequence of stages (subgoals). For example, *TidyHouse* can be considered to consist of *pick_0*, *place_0*, *pick_1*, *etc*.Each stage can correspond to multiple subtasks. For example, the stage *pick_i* includes *Navigate(s_0^i)* and *Pick(s_0^i)*. Thus, to be precise, the completion rate is computed based on stages instead of subtasks. We define a set of predicates to measure whether the goal of a stage is completed. A stage goal is completed if all the predicates associated with it are satisfied. The predicates are listed as follows:

- `holding(target_obj|i)`: The robot is holding the i-th object.
- `at(target_obj_pos|i,target_goal_pos|i)`: The i-th object is within 15cm of its goal position.
- `opened_drawer(target_marker|i)`: The target drawer is open (the joint position is larger than 0.4m).
- `closed_drawer(target_marker|i)`: The target drawer is close (the joint position is smaller than 0.1m).
- `opened_fridge(target_marker|i)`: The target fridge is open (the joint position is larger than $\frac{\pi}{2}$ radian).
- `closed_fridge(target_marker|i)`: The target fridge is close (the joint position is smaller than 0.15 radian).

During evaluation, we evaluate whether the current stage goal is completed at each time step. If the current stage goal is completed, we progress to the next stage. Hence, the completion rate

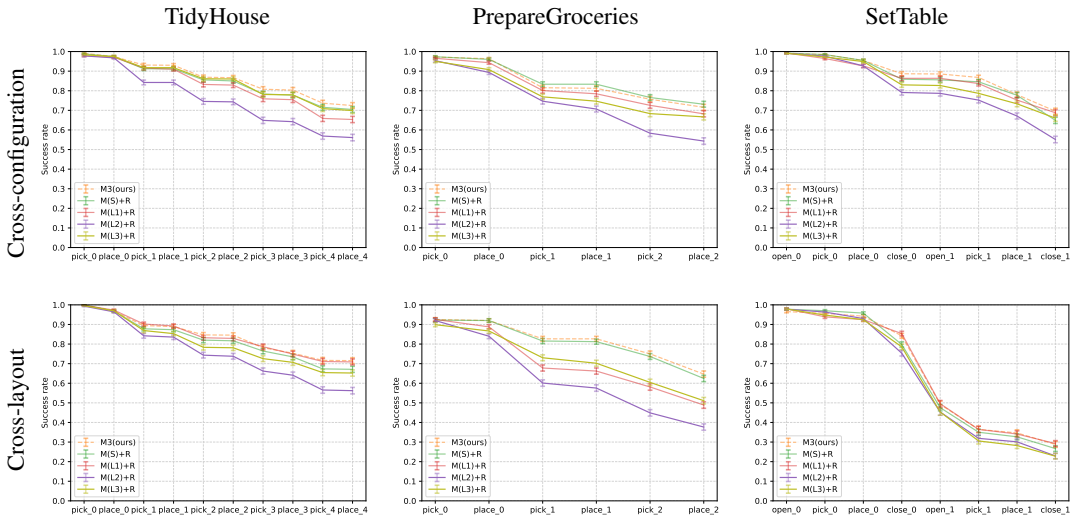


Figure 7: Progressive completion rates for HAB [1] tasks. The x-axis represents progressive subtasks. The y-axis represents the completion rate of each subtask. Results of ablation experiments are presented with solid lines. The mean and standard error for 100 episodes over 9 seeds are reported.

monotonically decreases. Listings 1, 2, 3 present the stages defined for each HAB task and the predicates associated with each stage. Note that the stage goal *place_i* only indicates that the object has been released at its goal position, but the placement can be unstable (*e.g.*, the object falls down the table), which can lead to the failure of the next stage. Besides, due to abstract grasp, it is difficult to place the object stably since the pose of the grasped object can not be fully controlled. Therefore, we modify the objective of *SetTable* to make the task achievable given abstract grasp. Concretely, instead of placing the fruit in the bowl, the robot only needs to place the fruit picked from the fridge at a goal position on the table.

F More Ablation Studies

In this section, we study the impact of different initial state distributions on mobile manipulation skills. We enlarge initial states by changing the distributions of the initial base position (the radius around the target) and orientation. For reference, the maximum radius around the target is set to 2m in the main experiments (Sec 5). Several experiments are conducted: **M(S)+R**, **M(L1)+R**, **M(L2)+R**, **M(L3)+R**. **M(S)+R**, **M(L1)+R** and **M(L2)+R** stand for the experiments where the maximum radii around the target are set to 1.5m, 2.5m and 4m respectively. **M(L3)+R** keeps the radius as 2m, but samples the initial base orientation from $[-\pi, \pi]$, instead of using the direction facing towards the target. Fig 7 shows the quantitative results. Enlarging the initial states in general leads to performance degradation. Compared to **M3** (71.2%/55.0%), **M(L1)+R** (67.4%/49.7%) and **M(L3)+R** (67.5%/46.4%) show moderate performance drop. **M(L2)+R** (55.2%/38.9%) shows the largest performance drop, which indicates that mobile manipulation skills are not able to handle long-range navigation yet. Moreover, **M(S)+R** (69.5%/52.1%) performs on par with **M3**. It implies that there usually exists a "sweet spot" of the initial state distribution for mobile manipulation skills as a trade-off between *achievability* and *composability*.

```
pick_0:
- "holding(target_obj|0)"
place_0:
- "not_holding()"
- "at(target_obj_pos|0,target_goal_pos|0)"
pick_1:
- "holding(target_obj|1)"
- "at(target_obj_pos|0,target_goal_pos|0)"
place_1:
- "not_holding()"
- "at(target_obj_pos|0,target_goal_pos|0)"
- "at(target_obj_pos|1,target_goal_pos|1)"
pick_2:
- "holding(target_obj|2)"
- "at(target_obj_pos|0,target_goal_pos|0)"
- "at(target_obj_pos|1,target_goal_pos|1)"
place_2:
- "not_holding()"
- "at(target_obj_pos|0,target_goal_pos|0)"
- "at(target_obj_pos|1,target_goal_pos|1)"
- "at(target_obj_pos|2,target_goal_pos|2)"
pick_3:
- "holding(target_obj|3)"
- "at(target_obj_pos|0,target_goal_pos|0)"
- "at(target_obj_pos|1,target_goal_pos|1)"
- "at(target_obj_pos|2,target_goal_pos|2)"
place_3:
- "not_holding()"
- "at(target_obj_pos|0,target_goal_pos|0)"
- "at(target_obj_pos|1,target_goal_pos|1)"
- "at(target_obj_pos|2,target_goal_pos|2)"
- "at(target_obj_pos|3,target_goal_pos|3)"
pick_4:
- "holding(target_obj|4)"
- "at(target_obj_pos|0,target_goal_pos|0)"
- "at(target_obj_pos|1,target_goal_pos|1)"
- "at(target_obj_pos|2,target_goal_pos|2)"
- "at(target_obj_pos|3,target_goal_pos|3)"
place_4:
- "not_holding()"
- "at(target_obj_pos|0,target_goal_pos|0)"
- "at(target_obj_pos|1,target_goal_pos|1)"
- "at(target_obj_pos|2,target_goal_pos|2)"
- "at(target_obj_pos|3,target_goal_pos|3)"
- "at(target_obj_pos|4,target_goal_pos|4)"
```

Listing 1: Stage goals and their associated predicates defined for *TidyHouse*. The stages are listed in the order for progressive evaluation.

```
pick_0:
- "holding(target_obj|0)"
place_0:
- "not_holding()"
- "at(target_obj_pos|0,target_goal_pos|0)"
pick_1:
- "holding(target_obj|1)"
- "at(target_obj_pos|0,target_goal_pos|0)"
place_1:
- "not_holding()"
- "at(target_obj_pos|0,target_goal_pos|0)"
- "at(target_obj_pos|1,target_goal_pos|1)"
pick_2:
- "holding(target_obj|2)"
- "at(target_obj_pos|0,target_goal_pos|0)"
- "at(target_obj_pos|1,target_goal_pos|1)"
place_2:
- "not_holding()"
- "at(target_obj_pos|0,target_goal_pos|0)"
- "at(target_obj_pos|1,target_goal_pos|1)"
- "at(target_obj_pos|2,target_goal_pos|2)"
```

Listing 2: Stage goals and their associated predicates defined for *PrepareGroceries*. The stages are listed in the order for progressive evaluation.

```yaml
open_0:
- "opened_drawer(target_marker|0)"
pick_0:
- "holding(target_obj|0)"
place_0:
- "not_holding()"
- "at(target_obj_pos|0,target_goal_pos|0)"
close_0:
- "closed_drawer(target_marker|0)"
- "at(target_obj_pos|0,target_goal_pos|0)"
open_1:
- "closed_drawer(target_marker|0)"
- "at(target_obj_pos|0,target_goal_pos|0)"
- "opened_fridge(target_marker|1)"
pick_1:
- "closed_drawer(target_marker|0)"
- "at(target_obj_pos|0,target_goal_pos|0)"
- "opened_fridge(target_marker|1)"
- "holding(target_obj|1)"
place_1:
- "closed_drawer(target_marker|0)"
- "at(target_obj_pos|0,target_goal_pos|0)"
- "not_holding()"
- "at(target_obj_pos|1,target_goal_pos|1)"
close_1:
- "closed_drawer(target_marker|0)"
- "at(target_obj_pos|0,target_goal_pos|0)"
- "closed_fridge(target_marker|1)"
- "at(target_obj_pos|1,target_goal_pos|1)"
```

Listing 3: Stage goals and their associated predicates defined for *SetTable*. The stages are listed in the order for progressive evaluation.