# Habitat-Web: Learning Embodied Object-Search from Human Demonstrations at Scale

Ram Ramrakhya[♡], Eric Undersander[♢], Dhruv Batra[♢,♡], Abhishek Das[♢]
[♡]Georgia Institute of Technology, [♢]Facebook AI Research
[♡]ram.ramrakhya@gatech.edu [♢]{eundersander,dbatra,abhshkdz}@fb.com

## Abstract

*We present a large-scale study of imitating human demonstrations on tasks that require a virtual robot to search for objects in new environments – (1) ObjectGoal Navigation (*e.g. 'find & go to a chair'*) and (2)* PICK&PLACE *(*e.g. 'find mug, pick mug, find counter, place mug on counter'*).*

*First, we develop a virtual teleoperation data-collection infrastructure – connecting Habitat simulator running in a web browser to Amazon Mechanical Turk, allowing remote users to teleoperate virtual robots, safely and at scale. We are collecting $100k$ demonstrations for* OBJECTNAV *and $12k$ demonstrations for* PICK&PLACE[1]*, which is an order of magnitude larger than existing human demonstration datasets in simulation and 2 orders of magnitude larger than existing datasets on real robots. Our virtual teleoperation data (collected so far) contains $\sim 22.1M$ actions, is equivalent to $13.4k$ hours of real-world teleoperation time, and illustrates rich diverse strategies for solving the tasks.*

*Second, we use this data to answer the question – how does large-scale imitation learning (IL) (which has not been hitherto possible) compare to large-scale reinforcement learning (RL) (which is the status quo)? On* OBJECTNAV*, we find that IL (with no bells or whistles) using only $50k$ human demonstrations outperforms RL using $240k$ agent-gathered trajectories. This effectively establishes an 'exchange rate' – a single human demonstration appears to be worth $\sim 5$ agent-gathered ones. More importantly, we find the IL-trained agent learns efficient object-search behavior from humans – it peeks into rooms, checks corners for small objects, turns in place to get a panoramic view – none of these are exhibited as prominently by the RL agent, and to induce these behaviors in RL agents would take tedious dense reward engineering. Finally, accuracy* vs. *training data size plots show promising scaling behavior, suggesting that simply collecting more demonstrations is likely to advance the state of art further. On* PICK&PLACE*, the comparison is even starker – IL agents achieve $\sim 18\%$ success on episodes with new object-receptacle locations when trained with $9.5k$ human demonstrations, while RL agents fail to get beyond $0\%$ success. Overall, our work provides compelling evidence for investing in large-scale imitation learning.*

## 1. Introduction

General-purpose robots that can perform a diverse set of embodied tasks in a diverse set of environments *have* to be good at visual exploration. Consider the canonical example of asking a household robot, *'Where are my keys?'*. To answer this (assuming the robot does not remember the answer from memory), the robot would have to search the house, often guided by intelligent priors – *e.g.* peeking into the washroom or kitchen might be sufficient to be reasonably sure the keys are not there, while exhaustively searching the living room might be much more important since keys are more likely to be there. While doing so, the robot has to internally keep track of where all it has been to avoid redundant search, and it might also have to interact with objects, *e.g.* check drawers and cabinets in the living room (but not those in the washroom or kitchen!).

This example illustrates fairly sophisticated exploration behavior, involving a careful interplay of various implicit objectives (semantic priors, exhaustive search, efficient navigation, interaction, *etc.*). Many recent tasks of interest in the embodied AI community – *e.g.* ObjectGoal Navigation [1, 2], rearrangement [3, 4], language-guided navigation [5, 6] and interaction [7], question answering [8–11] – involve some flavor of this visual exploration. With careful reward engineering, RL approaches to these tasks have achieved commendable success [12–16]. However, engineering the 'right' reward function so that the learned policy exhibits the desired behavior is unintuitive and frustrating (even for domain experts), expensive (requiring multiple rounds of retraining under different rewards), and not scalable to new tasks or desired behaviors. For complex tasks (*e.g.* object rearrangement or tasks abstractly specified in open-ended natural language), RL from scratch may not even get off the ground.

[1] $73k$ (out of $100k$) demonstrations for OBJECTNAV and $12k$ demonstrations of PICK&PLACE have been collected, which are used for all results in this paper. We expect the complete data collection to be in place by the final version.

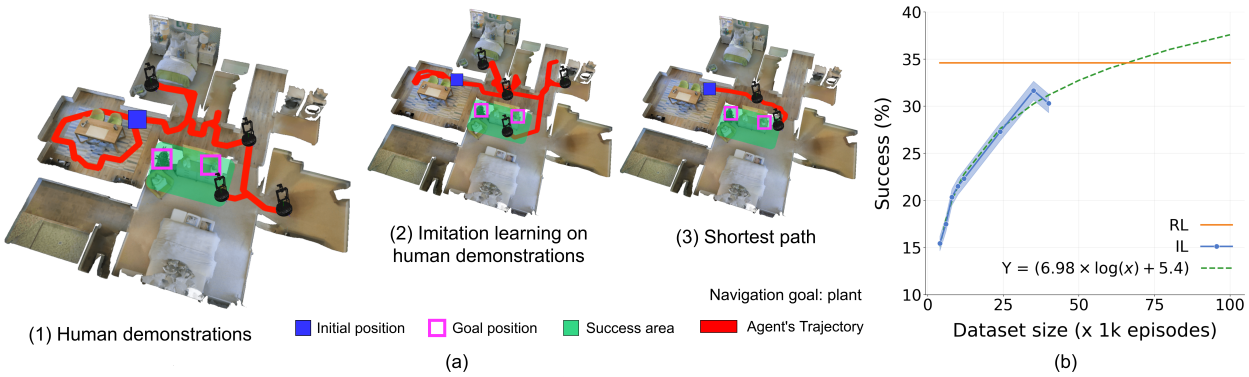



**Figure 1.** a) Example OBJECTNAV 1) human demonstration, 2) an agent trained on human demonstrations, and 3) a shortest path. Notice how humans demonstrate sophisticated exploration behavior to succeed at this task in unseen environments, which is hard to engineer into the right reward for an RL agent and is unlikely to be captured in shortest path demonstrations. An agent trained on human demonstrations learns this exploration and object-search behavior. b) Success on the OBJECTNAV val split *vs*. no. of human demonstrations for training.

In this work, we advance the alternative research agenda of imitation learning [17] – *i.e*. collecting a large dataset of human demonstrations (that implicitly capture intelligent behavior we wish to impart to our agents) and learning policies directly from these human demonstrations.

First, we develop a safe scalable virtual teleoperation data-collection infrastructure – connecting the Habitat simulator running in a browser to Amazon Mechanical Turk (AMT). We develop this in way that enables collecting human demonstrations for a variety of tasks being studied within the Habitat [18, 19] ecosystem (*e.g*. PointNav [2], OBJECTNAV [1, 2], ImageNav [20], VLN-CE [6], MultiON [21], *etc*.).

We use this infrastructure to collect human demonstration datasets for 2 tasks requiring visual search – 1) ObjectGoal Navigation (*e.g*. *'find & go to a chair'*) and 2) PICK&PLACE (*e.g*. *'find mug, pick mug, find counter, place on counter'*). We are collecting $100k$ demonstrations for OBJECTNAV and $12k$ demonstrations for PICK&PLACE[1]. In contrast, the largest existing datasets have 3-7$k$ human demonstrations in simulation (TEACh [22], CVDN [23], WAY [24]) and 200-500 demonstrations on real robots [25], which are 1 and 2 orders of magnitude smaller respectively. This virtual teleoperation data contains $\sim$22.1$M$ actions, which is equivalent to $13,400$ hours of real-world teleoperation time assuming a LoCoBot motion model from [26] (details in appendix (Sec. 4)). The first thing this data provides is a 'human baseline' with sufficiently tight error-bars to be taken seriously. On the OBJECTNAV validation split, humans achieve $93.7_{\pm 0.1}\%$ success and $42.5_{\pm 0.5}\%$ Success Weighted by Path Length (SPL) [2] (*vs*. $34.6\%$ success and $7.9\%$ SPL for the 2021 Habitat ObjectNav Challenge winner [14]). The success rate ($93.7\%$) suggests that this task is difficult even for humans (but largely doable). The SPL ($42.5\%$) suggests that even humans need to explore significantly.

Beyond scale, the data is also rich and diverse in the strategies that humans use to solve the tasks. Fig. 1 shows an example trajectory of an AMT user controlling a LoCoBot looking for a 'plant' in a new house – notice the peeking into rooms, looping around the dining table – all of which is (understandably) absent from the shortest path to the goal.

We use this data to answer the question – how does large-scale imitation learning (IL) (which has not been hitherto possible) compare to large-scale reinforcement learning (RL) (which is the status quo)? On OBJECTNAV, we find that IL (with no bells or whistles) using only $50k$ human demonstrations outperforms RL using $240k$ agent-gathered trajectories. This effectively establishes an 'exchange rate' – a single human demonstration appears to be worth 5 agent-gathered ones. More importantly, we find the IL-trained agent learns *efficient object-search behavior* – as shown in Fig. 1 and Sec. 7. The IL agent learns to mimic human behavior of peeking into rooms, checking corners for small objects, turning in place to get a panoramic view – none of these are exhibited as prominently by the RL agent. Finally, the accuracy *vs*. training-data-size plot (Fig. 1b) shows promising scaling behavior, suggesting that simply collecting more demonstrations is likely to advance the state of art further. On PICK&PLACE, the comparison is even starker – IL-agents achieve $\sim 18\%$ success on episodes with new object-receptacle locations when trained with $9.5k$ human demonstrations, while RL agents fail to get beyond $0\%$.

On both tasks, we find that demonstrations from humans are essential; imitating shortest paths from an oracle produces neither accuracy nor the strategic search behavior. In hindsight, this is perfectly understandable – shortest paths (*e.g*. Fig. 1(a3)) do not contain any exploration but the task requires the agent to explore. Essentially, a shortest path is inimitable, but imitation learning is invaluable. Overall, our work provides compelling evidence for investing in large-scale imitation learning of human demonstrations.

## 2. Related work

**Embodied Demonstrations from Humans**. Prior expert demonstration datasets for embodied tasks combining vision and action (and optionally language) can be broadly categorized into either consisting of shortest-path trajectories from a planner with privileged information [5, 7, 8, 27], or consisting of human-provided trajectories [22–24]. While some works in the former collect natural language data from humans [5, 7], we contend that collecting navigation data from humans is equally crucial. Datasets with human-provided navigation trajectories are typically small. TEACh [22], CVDN [23] and WAY [24] have $<10k$ episodes, while the EmbodiedQA [8] dataset has $\sim$700 human-provided episodes – all prohibitively small for training proficient agents. A key contribution of our work is a scalable web-based infrastructure for collecting human navigation and interaction demonstrations, that is easily extensible to *any* task situated in the Habitat [18] simulator, including language-based tasks. We have collected $\sim$5x more demonstrations ($60k+$) so far and are scaling up to $100k$. TODO: cite bc-z, check slack.

**Exploration**. Learning how to explore an environment to gather sufficient information for use in downstream tasks has a rich history [28]. Curiosity-based approaches typically use reinforcement learning to maximize intrinsic rewards that capture the surprise or state prediction error of the agent [29–31]. State visitation count rewards are also popular for learning exploration [32, 33]. We defer the reader to Ramakrishnan *et al*. [34] for a review of exploration objectives for embodied agents. For improving exploration in OBJECTNAV specifically, SemExp [16] made use of a modular policy for semantic mapping and path planning, Ye *et al*. [14] used time-decaying state visitation count reward, and Maksymets *et al*. [15] used area coverage reward.

Most relatedly, Chen *et al*. [35] used $\sim$700 human navigation trajectories from the EmbodiedQA dataset [8] (ignoring the questions) to learn task-independent exploration using imitation learning. We likewise train agents via imitation learning on human demonstrations, but rather than encouraging task-agnostic exploration, we consider human demonstrations to be a rich *task-specific* mix of exploration and efficient navigation, that simple architectures without explicit mapping and planning modules can be trained on.

## 3. Habitat-WebGL Infrastructure

To be able to train agents via imitation learning on human demonstrations, we first need a reliable pipeline to collect human demonstrations at scale. To this end, we develop a web-based setup to connect the Habitat simulator [18, 19] to AMT users, building on the work of Newman *et al*. [36].

**Interface**. Fig. 2 shows a screenshot of the interface an AMT user interacts with to complete a data collection task.

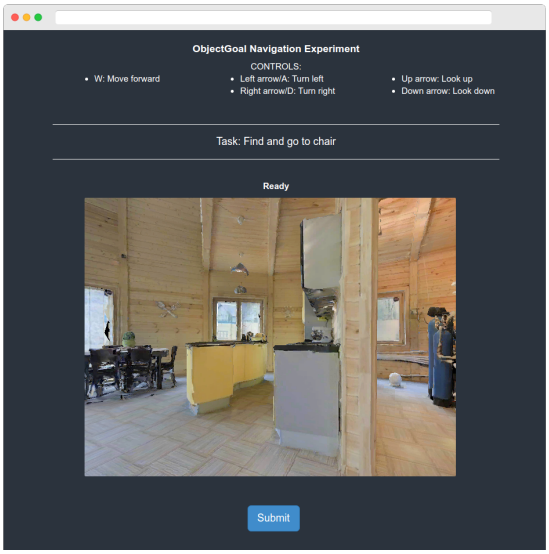



**Figure 2.** Screenshot of our Amazon Mechanical Turk interface for collecting OBJECTNAV demonstrations. Users are provided the agent's first-person view of the environment and an instruction such as "Find and go to chair". They can make the agent look around and move in the environment via keyboard controls, and can submit the task upon successful navigation by clicking 'Submit'.

This web application renders assets from Habitat-Sim running on the user's browser via WebGL. All data collection in this work was done in Matterport3D [37] scans, but any Habitat-compatible asset may be used in future. Users can see the agent's first-person RGB view, and can move around and grab / release objects using keyboard controls. On the task page, users are provided an instruction and details about keyboard controls to complete the task. For OBJECTNAV, we provide an instruction of the form *'Find and go to the* `<goal_object_category>`*'*. For tasks requiring interaction with objects (*e.g*. PICK&PLACE), we highlight the object under the user's gaze by drawing a 3D bounding box around it (pointed to by a crosshair as in video games). In our initial pilots, we found this to improve user experience when grabbing objects instead of users having to guess when objects are available to be picked up. When an object is successfully grabbed, it disappears from the first-person view and immediately appears in the 'inventory' area on the task interface. When a grabbed object is released, it is dropped at the center of the user's screen where the crosshair would be pointing to. If the crosshair points to a distance, the object is dropped on the floor from a height at a distance of $1m$ from the agent's location. Upon completion, users submit the task by clicking 'Submit'. At this point, the sequence of keyboard actions, agent, and object states are recorded in our backend server.

**Habitat simulator and PsiTurk**. Our Habitat-WebGL application is developed in Javascript, and allows us to access all C++ simulator APIs through Javascript bindings. This lets us use the full set of simulation features available in Habitat. To simulate physics, we use the physics APIs from

Habitat 2.0 [19], including rigid body dynamics support (C++ APIs exposed as Javascript bindings). Our interface executes actions entered by users every 50ms (rendering 20 frames per second) and then steps physics for 50ms in the simulator. All of our tasks on AMT are served using PsiTurk and an NGINX reverse proxy, and all data stored in a MySQL database. We use PsiTurk to manage the tasks as it provides us with useful helper functions to log task-related metadata, as well as launch and approve tasks.

**Validation**. To ensure data quality, every submitted AMT task goes through a set of validation checks. For OBJECTNAV, we use the same set of validation checks as the Habitat challenge evaluation setup, *i.e.* a task is considered successful only when the user has moved the agent to within $1m$ of the goal object. We do not limit the maximum number of steps to allow users on AMT to explore the environment. This captures key human exploration behavior necessary to succeed at these tasks. Similarly, for PICK&PLACE, a task is considered successful when the target object is placed on a receptacle object. Specifically, we check if the Euclidean distance between the centers of the target and receptable objects is less than $0.7m$, and that the target object is at a height greater than the receptacle center. We describe these tasks in detail in the following sections.

## 4. Tasks and Datasets

Using our web infrastructure, we collect demonstration datasets for two embodied tasks – OBJECTNAV [1, 2] and PICK&PLACE, an instantiation of object rearrangement [3].

### 4.1. ObjectGoal Navigation

In the ObjectGoal Navigation (OBJECTNAV) task, an agent is tasked with navigating to an instance of a specified object category (*e.g.* 'chair') in an unseen environment. The agent does not have access to a map of the environment and must navigate using an RGBD camera and a GPS+Compass sensor which provides location and orientation information relative to the start of the episode. The agent also receives the goal object category ID as input. The full action space is discrete and consists of MOVE_FORWARD ($0.25m$), TURN_LEFT ($30°$), TURN_RIGHT ($30°$), LOOK_UP ($30°$), LOOK_DOWN ($30°$), and STOP actions. For the episode to be considered successful, the agent must stop within $1m$ Euclidean distance of the goal object within a maximum of 500 steps and be able to turn to view the object from that end position [38].

**Human Demonstrations**. We collect human demonstrations on the 56 training scenes from Matterport3D [37] following the standard splits defined in [2, 37]. For each scene, we collect ∼42 demonstration episodes for each unique goal object category with a randomly set start location of the human demonstrator for each episode. This amounts to an average of 892 demonstrations per scene. Similar to when training artificial agents, humans can view first-person RGB on the task interface, but unlike artificial agents, humans do not get access to Depth and GPS+Compass. We assume humans are sufficiently proficient at inferring depth and odometry from vision alone. In total, so far, we have collected $50k$ OBJECTNAV demonstrations amounting to a total of ∼13.7 million steps of experience, each episode averaging 272 steps. We are in the process of scaling this up to $100k$ demonstration episodes (= 26.2 million steps).

**Shortest Path Demonstrations**. To compare against prior embodied datasets of shortest paths [5, 7, 8, 27] and to demonstrate the unique advantage of human demonstrations, we also generate a dataset of shortest paths. These demonstrations are generated by greedily fitting actions to follow the geodesic shortest path to the nearest navigable goal object viewpoint. Since shortest paths are (by design) shorter than human demonstrations (average 67 *vs.* 272 steps per demonstration), we compensate by generating a larger number of shortest paths to roughly match the steps with human demonstrations ($7.6M$ steps from $114k$ shortest paths *vs.* $13.7M$ steps from $50k$ human demonstrations).

**Analysis**. Table 8a reports statistics of our human and shortest path demonstration datasets. Recall that an episode is considered a failure if the target object is not found within 500 navigation steps. Under this definition, humans fail on 13.2% training set episodes; they fail on 0% episodes if we relax the step-limit. Surprisingly, SPL for humans is 39.9% for training split episodes, significantly lower than 94.9% for shortest paths underscoring the difficulty in searching for objects in in unseen environments.

We additionally report two metrics to demonstrate that the OBJECTNAV task requires significant exploration. Occupancy Coverage (OC) measures percentage of total area covered by the agent when navigating. To compute OC, we first divide the map into voxel grids of $2.5m \times 2.5m \times 2.5m$ and increment a counter for each visited voxel. Sight Coverage (SC) measures the percentage of total navigable area visible to the agent in its field of view (FOV) during an episode. To compute SC, we project a mask on the top-down map of the environment using the agent's FOV, that is iteratively updated at every step to update the area seen by the agent. OC and SC metrics for human demonstrations show that humans traverse 3-4x and observe 2x the area of the environment when performing this task compared to shortest paths.

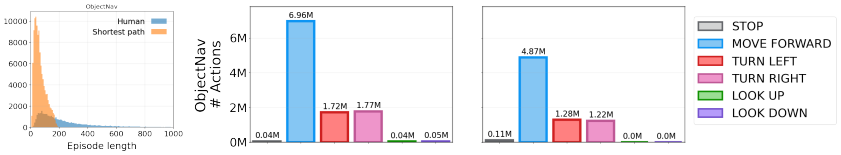
Fig. 8b,c show episode length and action histograms for human and shortest path demonstrations. Human demonstrations are longer (average ∼272 *vs.* ∼67 steps per demonstration) and have a slightly more uniform action distribution.

### 4.2. Object Rearrangement – PICK&PLACE

In the pick-and-place task (PICK&PLACE), an agent must follow an instruction of the form *'Place the* `<object>` *on the* `<receptacle>`', without being told the location of the `<object>` or `<receptacle>` in a new environment. The agent

| | | ObjectNav | | Pick-And-Place | |
|---|---|---|---|---|---|
| | | Human | Shortest Path | Human | Shortest Path |
| 1) | Total Episodes | 40,473 | 114,165 | 11,955 | 25,747 |
| 2) | Success | 86.8% | 100.0% | 86.3% | 100.0% |
| 3) | SPL | 39.9% | 94.9% | 21.2% | 90.9% |
| 4) | Occupancy coverage | 17.9% | 4.6% | 26.5% | 9.2% |
| 5) | Sight coverage | 67.7% | 33.2% | 70.3% | 42.5% |

(a)




(b) (c)

**Figure 3.** (a) Dataset statistics for human demonstrations *vs*. shortest paths for ObjectNav and Pick&Place. Coverage metrics are computed on subset of 1000 episodes. (b) Comparison of episode lengths and action histograms for human demonstrations *vs*. shortest paths. Human demonstrations are longer and have a more uniform action distribution than shortest paths.

must explore and navigate to the object, pick it up, explore and navigate to the receptacle, and place the previously picked-up object on it. Similar to ObjectNav, agents are not equipped with a map of the environment, and only have access to an RGBD camera and a GPS+compass sensor. At a high level, Pick&Place can be thought of as a natural extension of ObjectNav, performing it twice in the same episode – once to find the specified object and again to find the specified receptacle – delimited by grab and release actions. For object interaction, we use the 'magic pointer' abstraction defined in [3]. If the agent is not holding any object, the grab/release action will pick the object pointed to by its crosshair (at the center of its viewpoint) if within $1.5m$ of the object. If the agent is already holding an object, the grab/release action will drop the object at the crosshair location. If there is no drop-off point within $1.5m$ in the direction of the crosshair, the object will be dropped on the floor $1m$ in front of the agent. The full action space is discrete and consists of MOVE_FORWARD ($0.15m$), MOVE_BACKWARD ($0.15m$), TURN_LEFT ($5°$), TURN_RIGHT ($5°$), LOOK_UP ($5°$), LOOK_DOWN ($5°$), GRAB_RELEASE, NO_OP (step physics $50ms$), and STOP. For the episode to be considered successful, the agent must place the object on top of the receptacle – *i.e*. the object center should be at a height greater than the receptacle center, and within $0.7m$ of the receptacle object center – within 1500 steps. We picked this $0.7m$ threshold distance between the object and receptacle based on pilots on AMT. $0.7m$ was sufficiently strict for avoiding false positives in the collected demonstrations where users are able to submit the task without necessarily placing the object on top of the receptacle.

**Human Demonstrations**. We collect human demonstrations for Pick&Place on 9 scenes from Matterport3D [37]. In each episode, objects and receptacles are instantiated by randomly sampling from 457 possible object-receptacle pairs. We initialize the object and receptacle at randomly sampled locations in the environment, and collect 3 demonstrations for each object-receptacle pair. The agent, object, and receptacle locations are randomized across all demonstration episodes (including the 3 we collect for each object-receptacle pair). In total, we have $457 \times 3$ unique object-receptacle-agent position initializations per scene, amounting to $457 \times 3 \times 9 = \sim 12k$ demonstrations, which is $\sim$11.5M steps in experience, each episode averaging 932 steps.

**Shortest Path Demonstrations**. Similar to ObjectNav, we generate shortest path demonstrations for Pick&Place. These demonstrations are generated by first using the geodesic shortest-path follower to the object, then using a heuristic action planner to face and pick up the object, then following the geodesic shortest-path to the receptacle, and again using a heuristic action planner to drop the object on the receptacle. We generated 25.7k shortest path demonstrations for Pick&Place, each averaging 342 steps, amounting to a total of $\sim$8.8 million steps of experience.

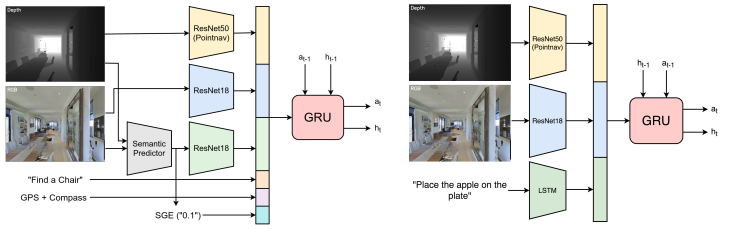
**Analysis**. Table 8a reports statistics for human and shortest path demonstrations. Similar to ObjectNav, humans have significantly lower SPL, and 2x higher occupancy and sight coverage compared to shortest paths, suggesting the need for exploration. Comparing episode lengths and action histograms (see appendix (Sec. 1.1) for figure), human demonstrations are longer and make use of all 9 actions. Interestingly, humans often use the MOVE_BACKWARD action to backtrack, which the shortest path agents do not use (by design), instead of turning $180°$ and moving forward. This behavior does not appear in ObjectNav shortest path demonstrations because there is just one target object, and so the geodesic shortest path would never involve backtracking or making $180°$ turns.

## 5. Imitation Learning from Demonstrations

We use behavior cloning to learn a policy from demonstrations. Let $\pi_\theta(a_t \mid o_t)$ denote a policy parametrized by $\theta$ that maps observations $o_t$ to a distribution over actions $a_t$. Let $\tau$ denote a trajectory consisting of state, observation, human action tuples: $\tau = \left(s_0, o_0, a_0, \ldots, s_T, o_T, a_T\right)$ and $\mathcal{T} = \left\{\tau^{(i)}\right\}_{i=1}^{N}$ denote a dataset of human demonstrations. The learning problem can be summarized as:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \sum_{(o_t, a_t) \in \tau^{(i)}} -\log\Big(\pi_\theta(a_t|o_t)\Big) \qquad (1)$$

**Inflection weighting** introduced in Wijmans *et al*. [9], adjusts the loss function to upweight timesteps where actions change (*i.e*. $a_{t-1} \neq a_t$). Specifically, the inflection weighting loss coefficient is computed as total no. of actions in




**(a)** OBJECTNAV architecture **(b)** PICK&PLACE architecture

| | Method | Success (↑) | SPL (↑) |
|---|---|---|---|
| 1) | RL (ExploreTillSeen) [15] | 20.0% | 6.5% |
| 2) | RL (ExploreTillSeen + THDA) [15] | 28.4% | 11.0% |
| 3) | RL (Red Rabbit) [14] | 34.6% | 7.9% |
| 4) | IL w/ Shortest Paths | 4.4% | 2.2% |
| 5) | IL w/ 40$k$ Human Demos (OBJECTNAV) | 31.6% | 8.5% |
| 6) | IL w/ 50$k$ Human Demos (+ THDA [15]) | 33.2% | 9.5% |
| 7) | Humans | 93.7% | 42.5% |

**(c)** OBJECTNAV results on MP3D-VAL

**Figure 4.** Our policy architectures for a) OBJECTNAV and b) PICK&PLACE. Both are simple CNN+RNN networks that embed and concatenate all sensory inputs, which are then fed into a GRU to predict actions. c) OBJECTNAV results on the MP3D VAL split [2, 37].

the dataset divided by the total no. of inflection points, and this coefficient is multiplied with the loss at each inflection timestep where $a_{t-1} \neq a_t$. This approach was found to be useful for tasks like navigation with long sequences of the same actions, *e.g.* several 'forward' actions when navigating corridors [9]. We use inflection weighting in all our experiments and found it to help over vanilla behavior cloning.

Our **base policy** is a simple CNN+RNN architecture. We first embed all sensory inputs using feed-forward modules. For RGB, we use a randomly initialized ResNet18 [39]. For depth, we use a ResNet50 that was pretrained on PointGoal navigation using DDPPO [12]. Then these RGB and depth features (and optionally other task-specific features) are concatenated and fed into a GRU [40] to predict a distribution over actions $a_{t+1}$. Task-specific architectural choices over this base policy are described in the next sections.

### 5.1. OBJECTNAV

Fig. 4a shows our OBJECTNAV architecture. Similar to Anand *et al.* [41], we feed in RGBD inputs of size $640 \times 480$ passed through a 2X2-AVGPOOL layer to reduce the resolution (performing low-pass filtering + downsampling). The agent also has a GPS+Compass sensor, which provides location and orientation relative to start of the episode. GPS+Compass inputs are pass through fully-connected layers to embed them to 32-d vectors. In addition to RGBD and GPS+Compass, following Ye *et al.* [14], we use two additional semantic features – semantic segmentation (SemSeg) of the input RGB and a 'Semantic Goal Exists' (SGE) scalar which is the fraction of the visual input occupied by the goal category. These semantic features are computed using a pretrained and frozen RedNet [42] that was pretrained on SUN RGB-D [43] and finetuned on 100$k$ randomly sampled front-facing views rendered in the Habitat simulator. Finally, we also feed in the object goal category embedded into a 32-d vector. All of these input features are concatenated to form an observation embedding, and fed into a 2-layer, 512-d GRU at every timestep. We train this policy for ∼400M steps (= ∼30 epochs on ∼50$k$ demonstration episodes). We evaluate checkpoints at every ∼15M steps for the last 50M steps of training, and report metrics for checkpoints with the highest success on the validation split.

### 5.2. PICK&PLACE

Fig. 4b shows our PICK&PLACE architecture. We feed in RGBD inputs of size $256 \times 256$. In addition to RGBD observations, the policy gets as input language instructions of the form *'Place the* `<object>` *on the* `<receptacle>`*'* encoded using a single-layer LSTM [44]. RGBD and instruction features are concatenated to form an observation embedding, which is fed into a 2-layer, 512-d GRU at every timestep. We train this policy for ∼90M steps (= ∼10 epochs on ∼9.5$k$ demonstration episodes). We evaluate checkpoints at every ∼10M steps during training, and report metrics for checkpoints with the highest success on the validation split.

## 6. Experiments & Results

### 6.1. OBJECTNAV

Table 4c reports results on the MP3D VAL split for several baselines. First, we compare our approach with two state-of-the-art RL approaches from prior work. Maksymets *et al.* [15] (row 1) train their policy using a reward structure that breaks OBJECTNAV into two subtasks – exploration and direct navigation to goal object once it is spotted. This agent gets a positive reward for maximizing area coverage until it sees the goal object. It then receives a navigation reward to minimize distance-to-object. This policy achieves 20.0% success and 6.5% SPL (row 1), which is 11.6% worse on success and 2.0% worse on SPL compared to behavior cloning on 40$k$ human demonstrations (row 5). [15] then combine this reward structure with Treasure Hunt Data Augmentation (THDA) – inserting arbitrary 3D target objects in the scene to augment the set of training episodes. With THDA, this achieves 28.4% success and 11.0% SPL (row 2), 3.2% worse and 2.5% better respectively than our approach (row 5). Ye *et al.* [14] (row 3) train their policy with a combination of exploration and distance-based navigation rewards, and their representations with several auxiliary tasks (*e.g.* inverse dynamics and predicting map coverage). This achieves 34.6% success and 7.9% SPL (row 3), which is 3.0% better on success and 0.6% worse on SPL than our approach (row 5). IL on a dataset of shortest paths achieves 4.4% success and 2.2% SPL (row 4), significantly worse than training on human demonstrations (31.6% success, 8.5%

SPL). Next, we also collected $10k$ human demonstrations on OBJECTNAV episodes generated in THDA fashion – *i.e.* asking humans to find randomly inserted objects. Notice that this involves pure exhaustive search, there are no semantic priors that humans can leverage in this setting. An IL agent trained on these $10k$ combined with the original $40k$ human demonstrations achieves 33.2% success and 9.5% SPL (row 6), *i.e.* adding these demos helps. This suggests that adding the human demonstrations with exhaustive search behavior are also helpful. TODO: Update IL on 40k results

Finally, we also benchmark human performance on the MP3D VAL split – 93.7% success, 42.5% SPL (row 7). Additional results for sensor ablations are in the appendix (Sec. 2).

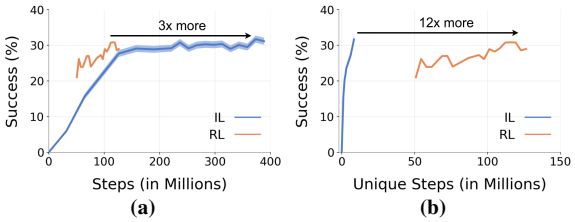
**Habitat ObjectNav Challenge Results**. Table 1 compares our results with prior approaches from the 2020 and 2021 Habitat Challenge leaderboards. Our approach achieves 25.6% success and 8.6% SPL (row 8), outperforming prior RL-trained counterparts – 1.1% better success, 2.2% better SPL than Red Rabbit (6-Act Base) [14] (row 5), and 4.5% better success, 0.2% worse SPL than ExploreTillSeen + THDA [15] (row 7) – with *just* $\sim 45k$ demonstrations.

| | Team / Method | Success (↑) | SPL (↑) |
|---|---|---|---|
| 1) | DD-PPO baseline [12, 14] | 6.2% | 2.1% |
| 2) | Active Exploration (Pre-explore) | 8.9% | 4.1% |
| 3) | SRCB-robot-sudoer | 14.4% | 7.5% |
| 4) | SemExp [45] | 17.9% | 7.1% |
| 5) | Red Rabbit (6-Act Base) [14] | 24.5% | 6.4% |
| 6) | Red Rabbit (6-Act Tether) [14] | 21.1% | 8.1% |
| 7) | ExploreTillSeen + THDA [15] | 21.1% | **8.8%** |
| 8) | IL w/ $50k$ Human Demos | **25.6%** | 8.6% |

**Table 1.** ObjectNav results on the Habitat ObjectNav Challenge TEST-STD split [46].

**Performance *vs.* Dataset size**. To investigate scaling behavior, we plot VAL success against the size of the human demonstrations dataset in Fig. 1b. We created splits of the human demonstrations' dataset of increasing sizes, from $4k$ to $50k$, and trained models with the same set of hyperparameters on each split. All hyperparameters were picked early in the course of the data collection (on the $4k$ and $12k$ subsplits) and fixed for later experiments. So VAL performance in the small-data regime may be an optimistic estimate and in the large data regime a pessimistic estimate. True scaling behavior may be even stronger. Increasing dataset size consistently improves performance and has not yet saturated, suggesting that simply collecting more demonstrations is likely to lead to further gains. We are collecting $50k$ more demonstrations to scale our dataset to $100k$ overall.

**Sample Efficiency**. Fig. 5 plots VAL success against no. of training steps of experience (in millions) in Fig. 5a and against *unique* steps of experience in Fig. 5b. Recall that




(a) (b)

**Figure 5.** Comparing RL and IL on (a) VAL success *vs.* no. of training steps, and (b) VAL success *vs.* no. of unique training steps. This distinguishes between an IL agent that learns from a static dataset *vs.* an RL agent that gathers unique trajectories on-the-fly.

| | | Method | VAL Success % (↑) | VAL SPL % (↑) | TEST Success % (↑) | TEST SPL % (↑) |
|---|---|---|---|---|---|---|
| New Inits. | 1) | IL w/ Shortest Paths | 1.9 | 1.8 | 1.7% | 1.6 |
| | 2) | IL w/ Human Demos | 17.6 ±0.8 | 9.7 ±0.3 | 17.5 | 9.8 |
| | 3) | Humans | 87.2 | 21.8% | 89.1 | 21.9 |
| New Instr. | 4) | IL w/ Shortest Paths | 1.3 | 1.2 | 1.1 | 1.0 |
| | 5) | IL w/ Human Demos | 15.9 ±0.2 | 8.4 ±0.4 | 15.1 | 8.3 |
| | 6) | Humans | 85.0 | 21.0 | 86.1 | 20.5 |
| New Envs. | 7) | IL w/ Shortest Paths | – | – | 0.2 | 0.3 |
| | 8) | IL w/ Human Demos | – | – | 8.3 | 4.1 |
| | 9) | Humans | – | – | 94.9 | 20.5 |

**Table 2.** Pick-and-place results on splits constructed with unseen initializations in seen environments (1-3), with unseen instructions (4-6), and with unseen environments (7-9).

IL involves $\sim 40$ epochs on a static dataset of $\sim 40k$ demos, while RL (from [14]) gathers unique agent-driven trajectories on-the-fly. Fig. 5a shows that IL behaves like supervised learning (as expected) with improvements coming from long training schedules; unfortunately, this means that wall-clock training times are not shorter. Fig. 5b shows that IL requires 12x fewer unique steps of experience to achieve comparable success and is thus much more sample-efficient.

**Zero-shot results on Gibson** [47] are in the appendix.

### 6.2. PICK&PLACE

**Results**. We report results in Table 2 across three evaluation splits. 1) New Initializations: new locations of objects and receptacles. This tests generalization to unseen locations in seen environments. 2) New Instructions: compositionally novel object-receptacle combinations of objects and receptacles individually seen during training. 3) New Environments: generalization to 2 scenes held out from training. Similar to OBJECTNAV and as described in Section 4, we also report results with shortest paths. Again, these paths are significantly shorter (average 342 *vs.* 932 steps per demonstration) and hence, we generate a larger dataset of $25.7k$ episodes roughly matching the cumulative steps of experience with human demonstrations ($8.8M$ shortest path steps *vs.* $11.5M$ human steps). Training on $9.5k$ human demonstrations achieves 17.5% success, 9.8% SPL on new object-receptacle initializations (row 2). Across splits, training on shortest paths hurts success by 8-16%. Going to new object-receptacle pairs, success drops by 2.4% (row 5 *vs.* 2), and then going to

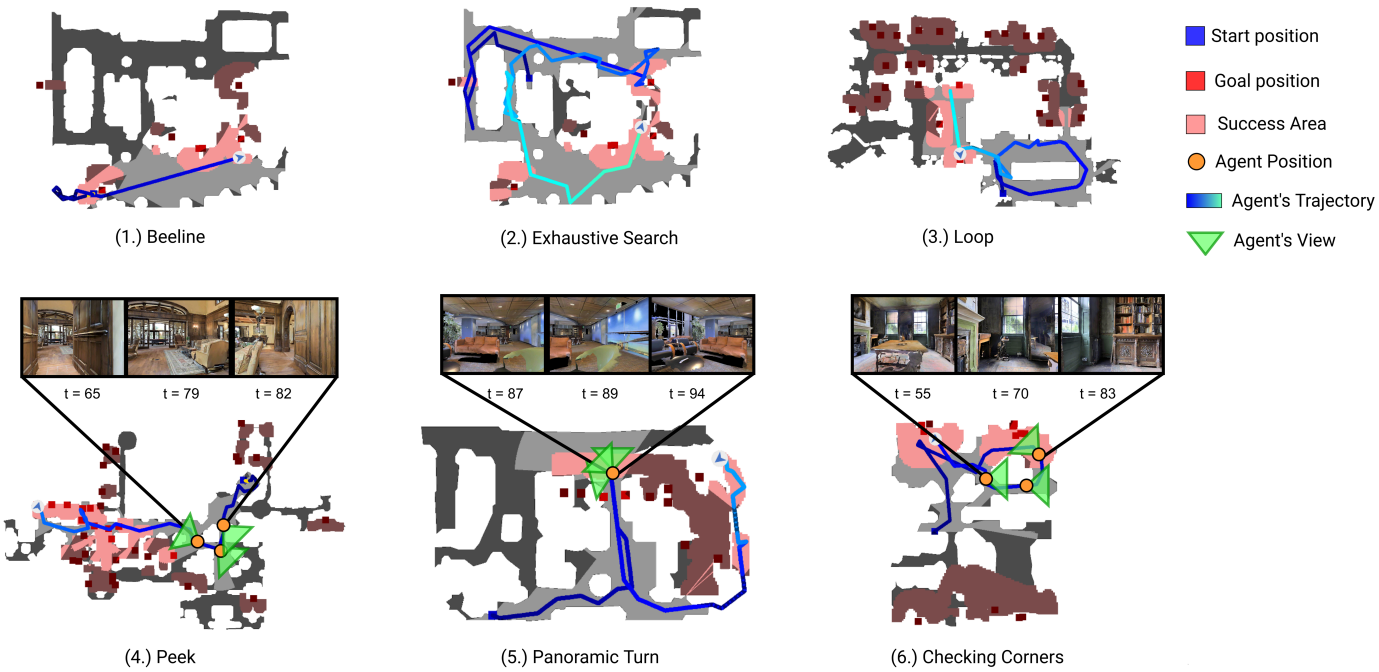



**Figure 6.** Visualizations of different learnt agent behaviors. Best viewed in videos at sites.google.com/view/object-search-supp.

new environments further hurts success by 6.8% (row 8 *vs.* 5). We also trained an RL policy with the exploration and distance-based rewards from [15], but it failed to get beyond 0% success on new object-receptacle intializations. See the appendix (Sec. 1.2) for training details.

**Performance *vs.* Dataset size**. Similar to OBJECTNAV, we trained policies on $2.5k$ to $9.5k$ subsets of our PICK&PLACE data, and found that performance continues to improve with more data. Figure in appendix (Sec. 1.3).

## 7. Characterizing Learnt Behaviors

To characterize the behaviors learnt by our best imitation-learned (IL) agents, we first sample 300 validation OBJECTNAV episodes for each method and manually categorize the behavior observed. A subset of observed behaviors are visualized in Fig. 13. Our agents demonstrate sophisticated object-search behaviors *e.g.* peeking into rooms to maximize sight coverage (SC), instead of occupancy coverage (OC), checking corners of rooms for small objects, beelining to goal object once seen, exhaustive search (ES), turning in place to get a panoramic view (PT), and looping back to recheck some areas. Amusingly, unlike shortest path / RL agents, these IL agents also stand idle and 'look around' *i.e.* turn in place, like humans. Table 3 quantifies these behaviors. See appendix (Sec. 5) for details on how these were computed. Agents trained with IL on human demonstrations have higher coverage (both occupancy and sight), peeking behavior, panoramic turns, beelines, and exhaustive search than RL. RL-trained agents achieve higher average Goal

| | Method | OC (%) | SC (%) | GRTS (%) | Peeks (%) | PT (%) | Beeline (%) | ES (%) |
|---|---|---|---|---|---|---|---|---|
| 1) | IL w/ shortest paths | 4.2±1.1 | 31.2±3.2 | 20.5±4.3 | 3.0±1.9 | 0.0±0.0 | 0.0±5.2 | 10.1±3.5 |
| 2) | IL w/ human demos | 21.4±1.8 | 72.1±3.5 | 22.4±4.1 | 19.6±4.4 | 4.3±2.3 | 10.3±5.1 | 55.3±5.6 |
| 3) | RL [14] | 14.6±1.6 | 66.6±5.1 | 27.7±8.5 | 9.7±5.5 | 0.0±0.0 | 0.1±2.2 | 49.0±7.0 |
| 4) | Humans | 15.4±1.6 | 70.3±3.4 | - | 13.8±3.9 | 5.1±2.4 | 23.6±4.8 | 52.1±5.6 |

**Table 3.** Quantifying semantic exploration behaviors for IL agents trained on shortest paths (row 1) and human demonstrations (row 2), the Red Rabbit RL agent [14] (row 3), and humans (row 4).

Room Time Spent (GRTS) – *i.e.* time spent in the room containing the target object – but also have significantly higher variance in GRTS across scenes compared to IL agents. See appendix (Sec. 5) for a per-scene breakdown of GRTS as well as histograms of time spent in each room (instead of just target room) when searching for a target object. We also discuss limitations of our approach in the appendix.

## 8. Conclusion

In this work, we developed the infrastructure to collect human demonstrations at scale and using this, trained imitation learning (IL) agents on $60k+$ human demonstrations for OBJECTNAV and PICK&PLACE. On OBJECTNAV, we find that IL using only $50k$ human demonstrations outperforms RL using $240k$ agent-gathered trajectories, and on PICK&PLACE, IL agents get to ∼18% success while RL fails to get beyond 0%. Qualitatively, we find that IL agents pick up on sophisticated object-search behavior implicitly captured in human demonstrations, much more prominently than RL agents. Overall, we believe our work makes a compelling case for investing in large-scale imitation learning of human demonstrations.

## References


[1] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun, "Minos: Multimodal indoor simulator for navigation in complex environments," *arXiv preprint arXiv:1712.03931*, 2017. 1, 2, 4

[2] P. Anderson, A. X. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018. 1, 2, 4, 6, 13

[3] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi, M. Savva, and H. Su, "Rearrangement: A Challenge for Embodied AI," *arXiv preprint arXiv:2011.01975*, 2020. 1, 4, 5

[4] L. Weihs, M. Deitke, A. Kembhavi, and R. Mottaghi, "Visual room rearrangement," in *CVPR*, 2021. 1

[5] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *CVPR*, 2018. 1, 3, 4

[6] J. Krantz, E. Wijmans, A. Majumdar, D. Batra, and S. Lee, "Beyond the nav-graph: Vision-and-language navigation in continuous environments," in *ECCV*, 2020. 1, 2

[7] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks," in *CVPR*, 2020. 1, 3, 4

[8] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied Question Answering," in *CVPR*, 2018. 1, 3, 4

[9] E. Wijmans, S. Datta, O. Maksymets, A. Das, G. Gkioxari, S. Lee, I. Essa, D. Parikh, and D. Batra, "Embodied Question Answering in Photorealistic Environments with Point Cloud Perception," in *CVPR*, 2019. 1, 5, 6

[10] A. Das, F. Carnevale, H. Merzic, L. Rimell, R. Schneider, J. Abramson, A. Hung, A. Ahuja, S. Clark, G. Wayne, and F. Hill, "Probing emergent semantics in predictive agents via question answering," in *International Conference on Machine Learning*, 2020. 1

[11] L. Yu, X. Chen, G. Gkioxari, M. Bansal, T. L. Berg, and D. Batra, "Multi-target embodied question answering," in *CVPR*, 2019. 1

[12] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames," in *ICLR*, 2020. 1, 6, 7, 12

[13] J. Ye, D. Batra, E. Wijmans, and A. Das, "Auxiliary tasks speed up learning pointgoal navigation," in *CoRL*, 2020. 1

[14] J. Ye, D. Batra, A. Das, and E. Wijmans, "Auxiliary Tasks and Exploration Enable ObjectNav," in *ICCV*, 2021. 1, 2, 3, 6, 7, 8, 12

[15] O. Maksymets, V. Cartillier, A. Gokaslan, E. Wijmans, W. Galuba, S. Lee, and D. Batra, "THDA: Treasure Hunt Data Augmentation for Semantic Navigation," in *ICCV*, 2021. 1, 3, 6, 7, 8, 13

[16] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," in *NeurIPS*, 2020. 1, 3

[17] S. Schaal, "Learning from demonstration," in *NIPS*, 1996. 2

[18] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, *et al.*, "Habitat: A platform for embodied AI research," in *ICCV*, 2019. 2, 3, 13

[19] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra, "Habitat 2.0: Training home assistants to rearrange their habitat," in *NeurIPS*, 2021. 2, 3, 4

[20] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *ICRA*, 2017. 2

[21] S. Wani, S. Patel, U. Jain, A. X. Chang, and M. Savva, "MultiON: Benchmarking Semantic Map Memory using Multi-Object Navigation," in *NeurIPS*, 2020. 2

[22] A. Padmakumar, J. Thomason, A. Shrivastava, P. Lange, A. Narayan-Chen, S. Gella, R. Piramuthu, G. Tur, and D. Hakkani-Tur, "TEACh: Task-driven Embodied Agents that Chat," *arXiv preprint arXiv:2110.00534*, 2021. 2, 3

[23] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer, "Vision-and-dialog navigation," in *Conference on Robot Learning*, 2020. 2, 3

[24] M. Hahn, J. Krantz, D. Batra, D. Parikh, J. M. Rehg, S. Lee, and P. Anderson, "Where are you? localization from embodied dialog," *arXiv preprint arXiv:2011.08277*, 2020. 2, 3

[25] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," in *CoRL*, 2021. 2

[26] J. Krantz, A. Gokaslan, D. Batra, S. Lee, and O. Maksymets, "Waypoint models for instruction-guided navigation in continuous environments," in *ICCV*, 2021. 2, 13

[27] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "IQA: Visual Question Answering in Interactive Environments," in *CVPR*, 2018. 3, 4

[28] J. Schmidhuber, "A possibility for implementing curiosity and boredom in model-building neural controllers," in *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, 1991. 3

[29] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," *arXiv preprint arXiv:1507.00814*, 2015. 3

[30] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *ICML*, 2017. 3

[31] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, "Large-scale study of curiosity-driven learning," *arXiv preprint arXiv:1808.04355*, 2018. 3

[32] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," in *NeurIPS*, 2016. 3

[33] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "# exploration: A study of count-based exploration for deep reinforcement learning," in *NeurIPS*, 2017. 3

[34] S. K. Ramakrishnan, D. Jayaraman, and K. Grauman, "An exploration of embodied visual exploration," *arXiv preprint arXiv:2001.02192*, 2020. 3

[35] T. Chen, S. Gupta, and A. Gupta, "Learning exploration policies for navigation," in *ICLR*, 2019. 3

[36] B. Newman, K. Carlberg, and R. Desai, "Optimal assistance for object-rearrangement tasks in augmented reality," *arXiv preprint arXiv:2010.07358*, 2020. 3

[37] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3D: Learning from RGB-D Data in Indoor Environments," in *3DV*, 2017. MatterPort3D dataset license: http://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf. 3, 4, 5, 6, 13

[38] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans, "Objectnav revisited: On evaluation of embodied agents navigating to objects," *arXiv preprint arXiv:2006.13171*, 2020. 4

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016. 6, 12

[40] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *EMNLP*, 2014. 6

[41] A. Anand, E. Belilovsky, K. Kastner, H. Larochelle, and A. Courville, "Blindfold baselines for embodied qa," *arXiv preprint arXiv:1811.05013*, 2018. 6

[42] J. Jiang, L. Zheng, F. Luo, and Z. Zhang, "Rednet: Residual encoder-decoder network for indoor rgb-d semantic segmentation," 2018. 6

[43] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 567–576, 2015. 6

[44] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, 1997. 6, 12

[45] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," in *NeurIPS*, 2020. 7, 13

[46] H. Team, "Habitat challenge, 2020." https://aihabitat.org/challenge/2020, 2020. 7

[47] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *CVPR*, 2018. Gibson dataset license agreement available at storage.googleapis.com/gibson_material/Agreement GDS 06-04-18.pdf. 7, 13

[48] A. Mousavian, A.toshev, M. Fiser, J. Kosecka, A. Wahid, and J. Davidson, "Visual representations for semantic target driven navigation," in *International Conference on Robotics and Automation (ICRA)*, 2019. 13

[49] D. S. Chaplot, S. Gupta, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural mapping," in *ICLR*, 2020. 13

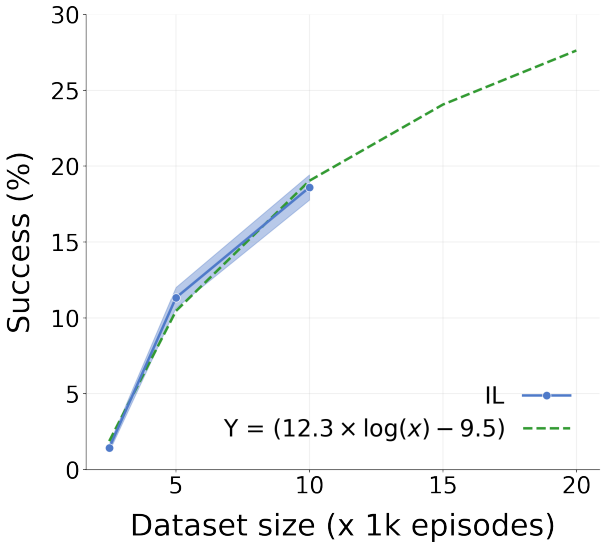



**Figure 7.** Dataset size vs performance plot for PICK&PLACE.

# A. Appendix

## A.1. Pick&Place

Recall that in the pick-and-place task (PICK&PLACE), an agent must follow an instruction of the form *'Place the* `<object>` *on the* `<receptacle>`*'*, without being told the location of the `<object>` or `<receptacle>` in a new environment. The agent must explore and navigate to the object, pick it up, explore and navigate to the receptacle, and place the previously picked-up object on it. In this section, we go over statistics of the human demonstrations dataset, how our PICK&PLACE imitation learning (IL) agents scale as a function of training dataset size, and details of our reinforcement learning baseline for PICK&PLACE.

### A.1.1 Dataset Stats

Fig. 8 compares the episode length and action histograms for human and shortest path demonstrations for PICK&PLACE. Human demonstrations are longer (average 932 vs 342 steps per demonstration) and have a more uniform action distribution compared to shortest paths. Human demonstrations also make use of all 9 actions whereas shortest path demonstrations use only 6 actions. Notice, humans also tend to stand idle and do nothing (50ms of idle time is translated to a NO_OP action). They likely use this time to strategize their next set of actions to explore the environment, which is not the case in shortest path demonstrations (by design).

### A.1.2 RL Baseline

Similar to the imitation learning baseline, our base policy is a simple CNN+RNN architecture. We first embed all sensory inputs using feed-forward modules. For RGB, we use a randomly initialized ResNet18 [39]. For depth, we use a ResNet50 that was pretrained on PointGoal navigation using DDPPO [12]. In addition to RGBD observations, the policy gets as input language instructions of the form *'Place the* `<object>` *on the* `<receptacle>`*'* encoded using a single-layer LSTM [44]. RGBD and instruction features are concatenated to form an observation embedding, which is fed into a 2-layer, 512-d GRU at every timestep. We train this policy for $\sim$100M steps on $\sim 9.5k$ episodes.

**Rewards**. The agent receives a sparse success reward $r_{success}$, a slack reward $r_{slack}$ to motivate faster goal-seeking, an exploration reward $r_{explore}$, an object seen reward $r_{seen}$, a grab/release success reward $r_{grab\_release}$, and a drop penalty reward $r_{drop\_penalty}$ to penalize dropping the object far from the receptacle. For incentivizing exploration, we use a visitation-based coverage reward from Ye *et al*. [14]. We first divide the map into a voxel grid of $2.5m \times 2.5m \times 2.5m$ voxels and reward the agent for visiting each voxel. Similar to [14], we smooth $r_{explore}$ by decaying it by number of steps the agent has spent in the voxel (visit count $v$). To ensure that the agent prioritizes PICK&PLACE (and not just exploration), we decay $r_{explore}$ based on episode timestep $t$ with a decay constant of $d = 0.995$. The agent is provided a reward for exploration until it sees the object. Once it sees the object, it receives a significant positive reward $r_{seen}$, and then the reward switches to a path-efficiency based navigation reward. In addition, the agent also receives a significant positive reward when it successfully grabs a object or releases the object close to the receptacle.

$$r_{\text{total}} = r_{\text{success}} + r_{\text{slack}} + r_{\text{explore}} + r_{\text{grab\_release}} \quad (2a)$$

$$+ r_{\text{drop\_penalty}} + r_{\text{seen}} \quad (2b)$$

$$r_{\text{success}} = 5.5 \quad \text{on success} \quad (2c)$$

$$r_{\text{slack}} = -10^{-4} \quad \text{per step} \quad (2d)$$

$$r_{\text{seen}} = 1.5 \quad \text{First time object seen} \quad (2e)$$

$$r_{\text{drop\_penalty}} = -3.5 \quad \text{Object dropped} > 2m \text{ away from receptacle} \quad (2f)$$

$$r_{\text{grab\_release}} = 2.0 \quad \text{Grab / release success} \quad (2g)$$

$$r_{\text{explore}} = 0.25 \times \frac{d^t}{v} \quad \text{Until object seen} \quad (2h)$$

**Results**. A policy trained with this reward for 100M steps fails to get beyond 0% success on the PICK&PLACE task. The agent learns to pick up the object at the start of training if it sees the object while navigating but it fails to search for the receptacle and place the object on top of receptacle. Overall, throughout training, the agent doesn't solve the task

| Method | Success (↑) | SPL (↑) |
|---|---|---|
| 1) IL wo/ Vision | 0.0% | 0.0% |
| 2) IL wo/ Semantic Input | 22.7% | 6.1% |
| 3) IL w/ Shortest Paths | 4.4% | 2.2% |
| 4) IL w/ $40k$ Human Demos (OBJECTNAV) | 31.6% | 8.5% |
| 5) IL w/ $50k$ Human Demos (+ THDA [15]) | 33.2% | 9.5% |
| 6) Humans | 93.7% | 42.5% |

**Table 4.** ObjectNav ablation results on the MP3D VAL split [2, 37].

successfully even once demonstrating the difficulty of the task and inadequacy of the above reward structure.

#### A.1.3 Performance *vs*. Dataset Size

Fig. 7 plots VAL success of our IL agent *vs*. the size of the PICK&PLACE human demonstrations dataset. We trained policies on $2.5k$ to $9.5k$ subsets of the data. Performance continues to improve with more data and has not saturated.

### A.2. ObjectNav Sensor Ablations

Table 4 reports results on the MP3D VAL split for various ablations of our approach. First, without any visual input (row 1), *i.e*. no RGBD and semantic inputs, the agent fails to learn anything (0% success, 0% SPL). Second, without SemSeg and SGE features (and keeping only RGB and Depth features) to the policy, performance drops by 8.9% success and 2.4% SPL (row 2 vs 4). Third, we trained a policy on a dataset of shortest paths instead of human demonstrations. Note that since shortest paths are shorter than human demonstrations (average 67 *vs*. 272 steps per demo), we compensate by generating a larger number of shortest paths to roughly match the steps of experience ($7.6M$ steps from $114k$ shortest paths *vs*. $10.6M$ steps from $40k$ human demonstrations). The policy trained on shortest paths achieves 4.4% success and 2.2% SPL (row 3), significantly worse than training on human demonstrations (31.6% success, 8.5% SPL). Finally, to get a sense for human performance, we also collected demonstrations on the MP3D VAL split. Humans achieve 93.7% success, 42.5% SPL (row 5).

### A.3. Zero-shot OBJECTNAV results on Gibson

To test generalization of the IL agents trained on human demonstrations, we report zero-shot results by transferring our policy trained on $40k$ human demonstrations to the Gibson dataset [47] VAL split in Table 5. To enable zero-shot transfer of semantic features, we remap the common goal categories (chair, couch, potted plant, bed, toilet, TV, diningtable) from Matterport3D [37] to Gibson goal category IDs. Our IL agent achieves 16.6% success and 2.5% SPL (row 7) with no finetuning on Gibson dataset. Comparing our zero-shot results to approaches trained on Gibson, our IL agent is 0.5% better on success and 2.4% worse on SPL than an RL baseline that takes RGBD + Semantics as input (row 3 *vs*. row 7), and it is 38.0% worse on success and 17.4% worse on SPL than SemExp [45] (row 6 *vs*. row 7).

| Method | Success (↑) | SPL (↑) |
|---|---|---|
| 1) Random | 0.4% | 0.4% |
| 2) RGBD+RL [18] | 8.2% | 2.7% |
| 3) RGBD+Semantics+RL [48] | 15.9% | 4.9% |
| 4) Classical Map + FBE | 40.3% | 12.4% |
| 5) Active Neural SLAM [49] | 44.6% | 14.5% |
| 6) SemExp [45] | 54.4% | 19.9% |
| 7) **IL w/ $40k$ Human Demos (Zero-Shot)** | 16.6% | 2.5% |

**Table 5.** ObjectNav results on the Gibson VAL split.

### A.4. Estimating time using a LoCoBot motion model

To estimate the time a robot would take to execute the collected human trajectories in the real world, we use the LoCoBot motion model from Krantz *et al*. [26]. This model consists of a rotation function that maps turn angle to time and a translation function that maps straight-line distance to time. For estimating time required for grab/release actions, we replace them with $0.15m$ forward steps and use the straight-line distance translation function. We use the MOVEBASE controller from [26] for all our time estimates, with the following rotation and translation equations:

$$y^{rotate} = 0.000358\phi^2 + 0.108\phi + 2.23 \quad (3)$$

$$y^{translate} = 4.2x + 0.362 \quad (4)$$

### A.5. Characterizing Learnt Behaviors

In this section, we describe the metrics used to characterize the exploration behavior exhibited by these agents in Sec. 7 in the main paper. These include 1) Occupancy Coverage (OC) and 2) Sight Coverage (SC) introduced in Sec. 4 in the main paper, as well as 3) Goal Room Time Spent (GRTS) – the number of steps as a fraction of total episode length an agent takes within the room bounding box containing the target object, 4) Peeks – check if the agent steps back into the last visited room after taking just ∼10 steps in another room, 5) Panoramic Turn (PT) – whether the agent stands at one place and turns left and right to get sweeping views, 6) Beeline – if the agent takes 10 continuous forward actions before reaching the goal in the last 15 steps, 7) Exhaustive Search (ES) – $\geq$ 75% sight coverage. To compute these metrics, we use the semantic annotations in Matterport3D. These annotations provide 3D bounding box coordinates for each room category in an environment. We use these bounding box coordinates to track the rooms an agent visits during an episode. GRTS gives us a measure of how often the agent ends up reaching goal object room but doesn't successfully locate the object. A higher GTRS suggests that the agent

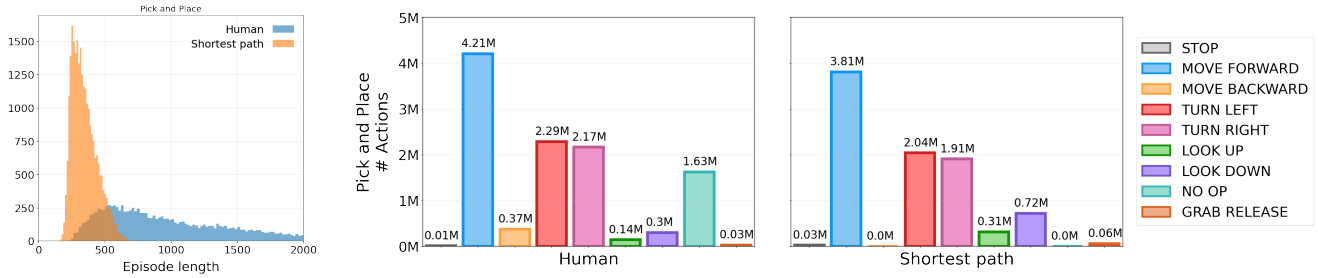



**Figure 8.** Comparison of episode lengths and action histograms for human demonstrations *vs*. shortest paths for PICK&PLACE. Human demonstrations are longer and have a more uniform action distribution than shortest paths.

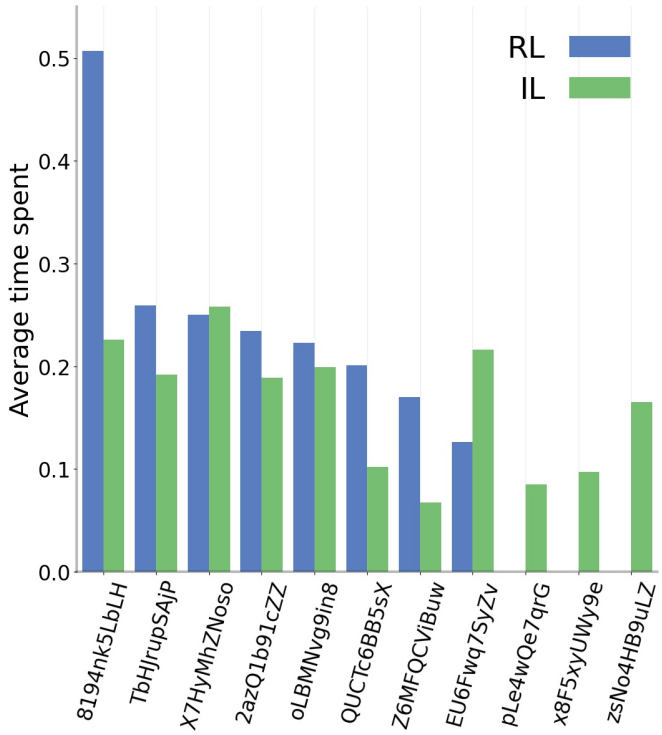



**Figure 9.** Per scene breakdown of GRTS for IL and RL agent on MP3D VAL split.

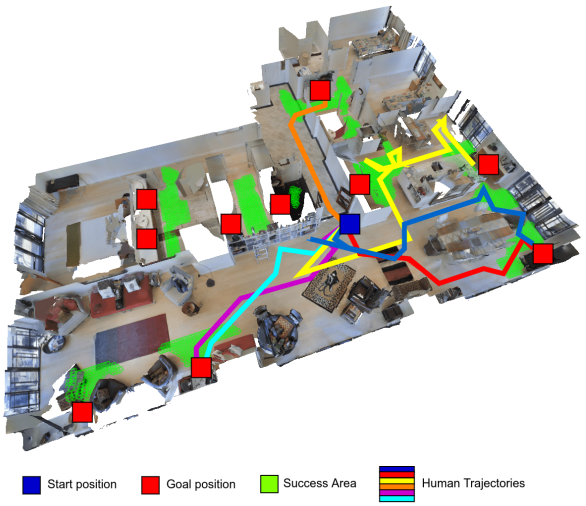



**Figure 10.** Visualizing multiple human demonstrations for OBJECTNAV all starting from the same start position and searching for 'cabinet'.

is at least good at reaching semantically meaningful locations in search of the goal object. We find that RL agents have higher average GRTS but also significantly higher variance in GRTS across scenes while our IL agents have lower average GRTS but more consistently spend time in the target room (see Fig. 9). To evaluate not just the final room the agent ends up at, but all the rooms it visits through the course of an episode, we also plot distributions of the time spent per room category for each goal object (see Fig. 14) for human demonstrations *vs*. IL agents trained on human demonstrations *vs*. RL agents.

### A.6. Inter-human Variance in OBJECTNAV

To get a sense for the variance in OBJECTNAV human demonstrations, we collected 20 unique human-provided trajectories for the same initial location and target object ('cabinet'). This is visualized in Fig. 10. We see that there is quite a bit of diversity in navigation trajectories across humans. They often navigate to different instances of the goal object category 'cabinet', and even when multiple humans go to the same object instance, the routes taken are different (red *vs*. blue trajectory).

We also plot the average SPL per AMT user in our dataset in Fig. 11. We find that human performance has a lot of variability, ranging from 25.2% to 68.2% (Fig. 11a). The SPL range that has the most humans is ∼50%. The best-performing human annotator achieves an SPL of 68.2% averaged over 6 episodes (Fig. 11b), which is particularly close to shortest paths and arguably *super-human*.

### A.7. AMT Interface

Fig. 12 shows a screenshot of our AMT interface for collecting PICK&PLACE demonstrations. For the PICK&PLACE task, we provide humans with an instruction of the form '*Place the* `<object>` *on the* `<receptacle>`', without being told the location of the `<object>` or `<receptacle>` in a new envi-

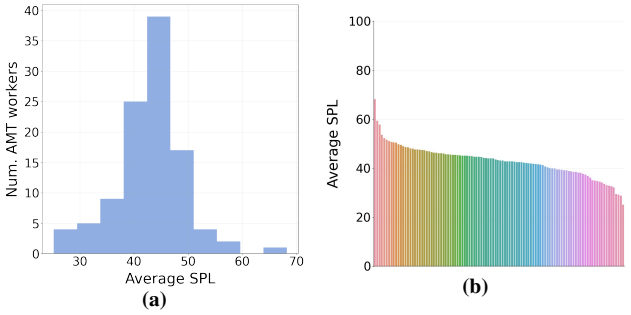



**Figure 11.** a) Histogram of average SPL for each AMT user for OBJECTNAV. b) Plot showing average SPL for AMT users for OBJECTNAV. These plots clearly demonstrate some humans are better at solving the OBJECTNAV task than others.

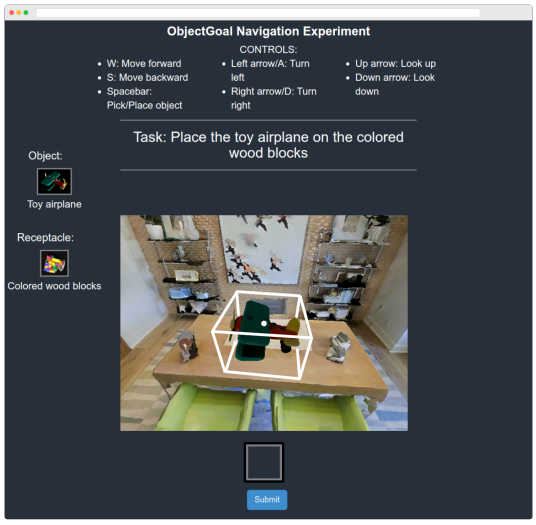



**Figure 12.** Screenshot of our Amazon Mechanical Turk interface for collecting PICK&PLACE demonstrations. Users are provided the agent's first-person view of the environment and an instruction such as "Pick the toy airplane and place it on the the colored wood blocks". They can make the agent look around and move in the environment via keyboard controls, and can submit the task upon successful completion by clicking the 'Submit' button.

ronment, and they can see agent's first-person view of the environment. They can make the agent move, look around, and interact with the environment using keyboard controls. Once the AMT user completes the task they can submit the task by clicking the 'Submit' button. We then run task-specific validation checks to ensure only successful tasks get submitted.

## A.8. Limitations

Our approach is fundamentally limited by the limitations of imitation learning as our approach uses vanilla behavior cloning with inflection weighting. Additionally, these agents trained on human demonstrations exhibit some common failure cases. Some examples of common failure cases are – reaching close to the goal object but not within goal radius and ending episode early, trying to move straight when agent is colliding and getting stuck, looping around multiple instances of the goal object and as a result, exceeding maximum episode steps, and exploring the environment and not finding the goal object. Our approach is also limited by the amount of human demonstrations we can gather and the agent architecture being trained on this dataset. Currently, we use a vanilla CNN+RNN architecture to learn imitation learning policies but we can build better architecture which make full use of the rich semantic information these human demonstrations have.

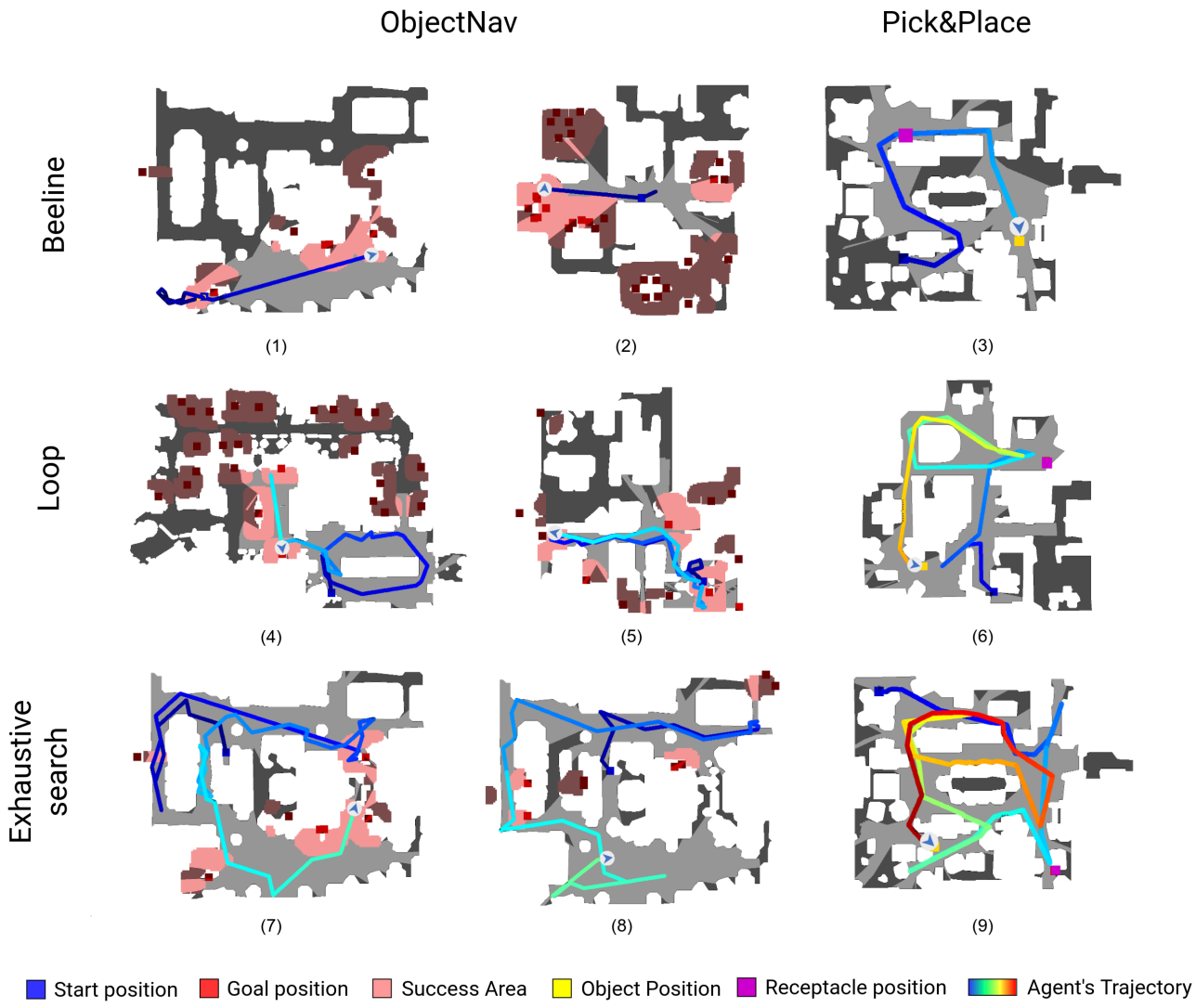



**Figure 13.** Visualizations of learnt agent behaviors for OBJECTNAV and PICK&PLACE. Best viewed in videos at sites.google.com/view/object-search-supp.

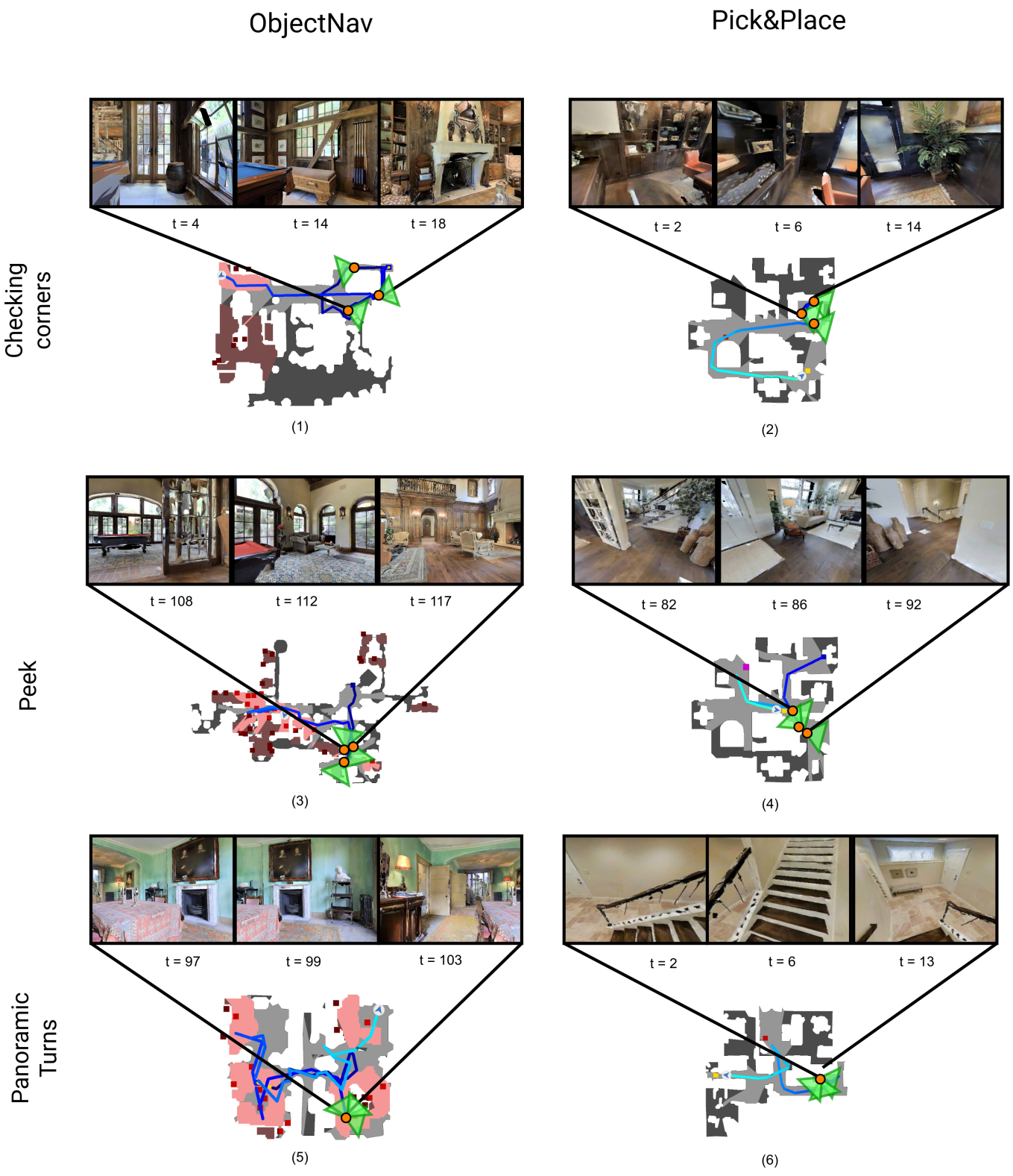
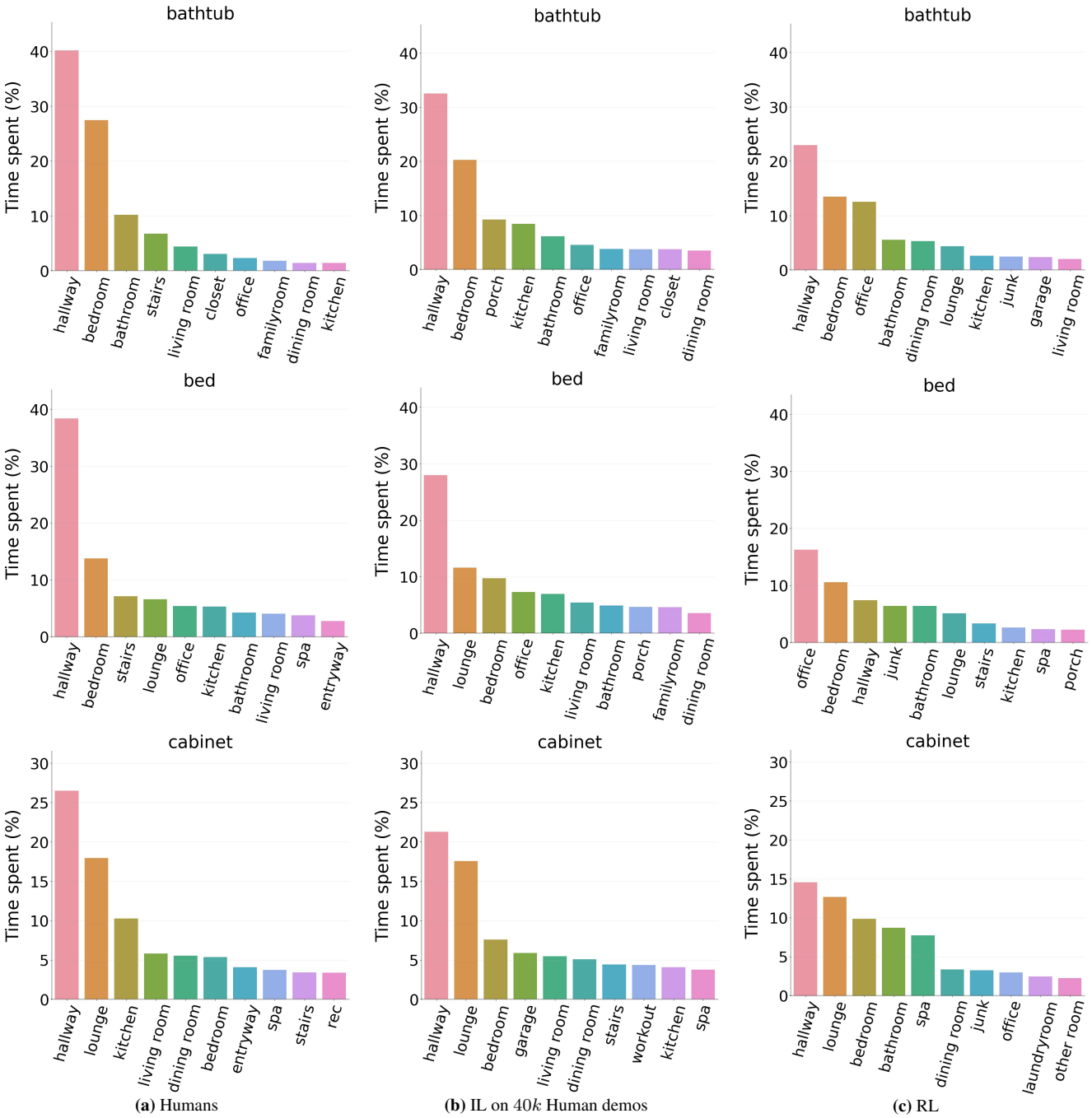
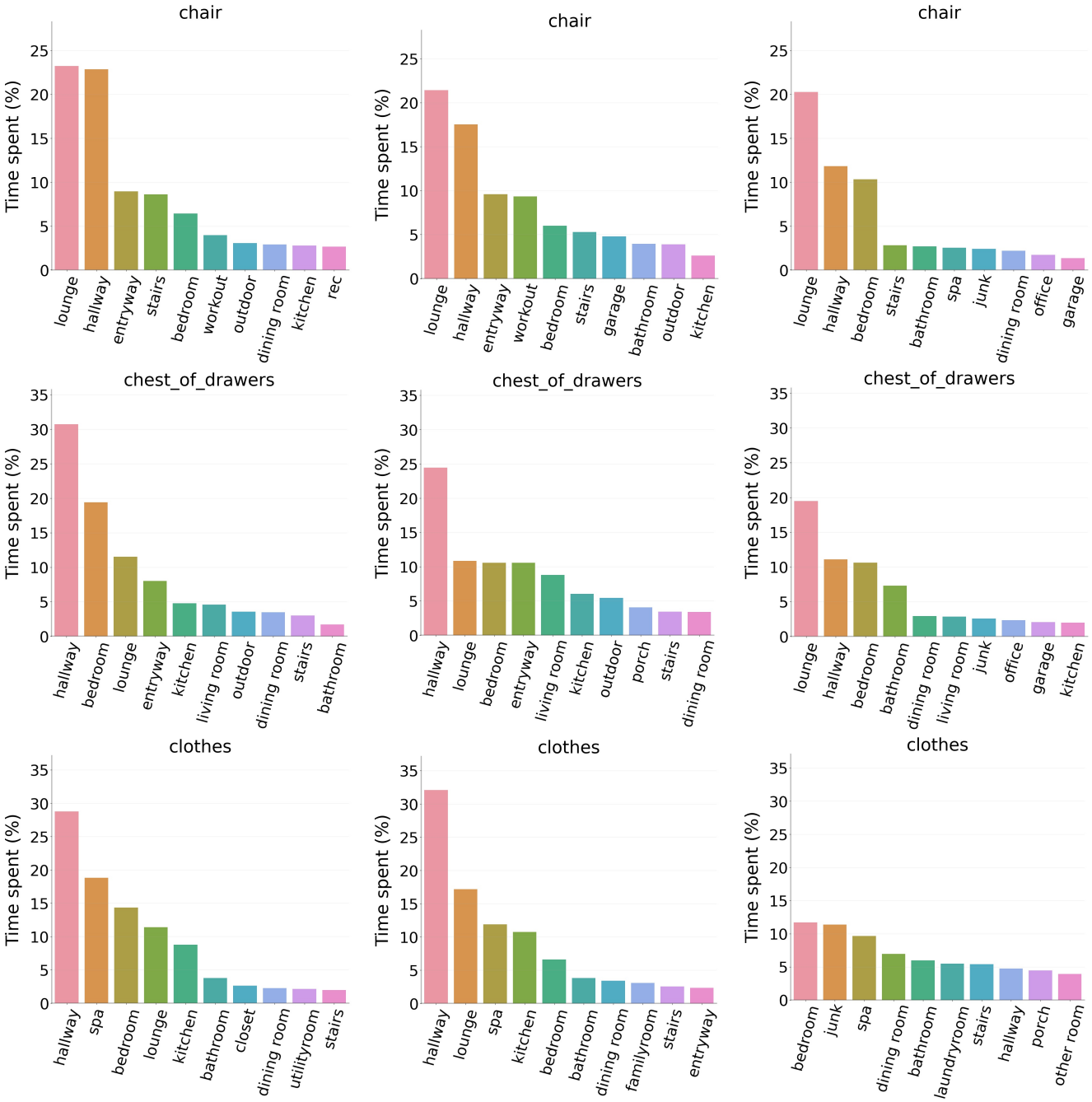



**Figure 13.** Visualizations of learnt agent behaviors for OBJECTNAV and PICK&PLACE. Best viewed in videos at sites.google.com/view/object-search-supp.



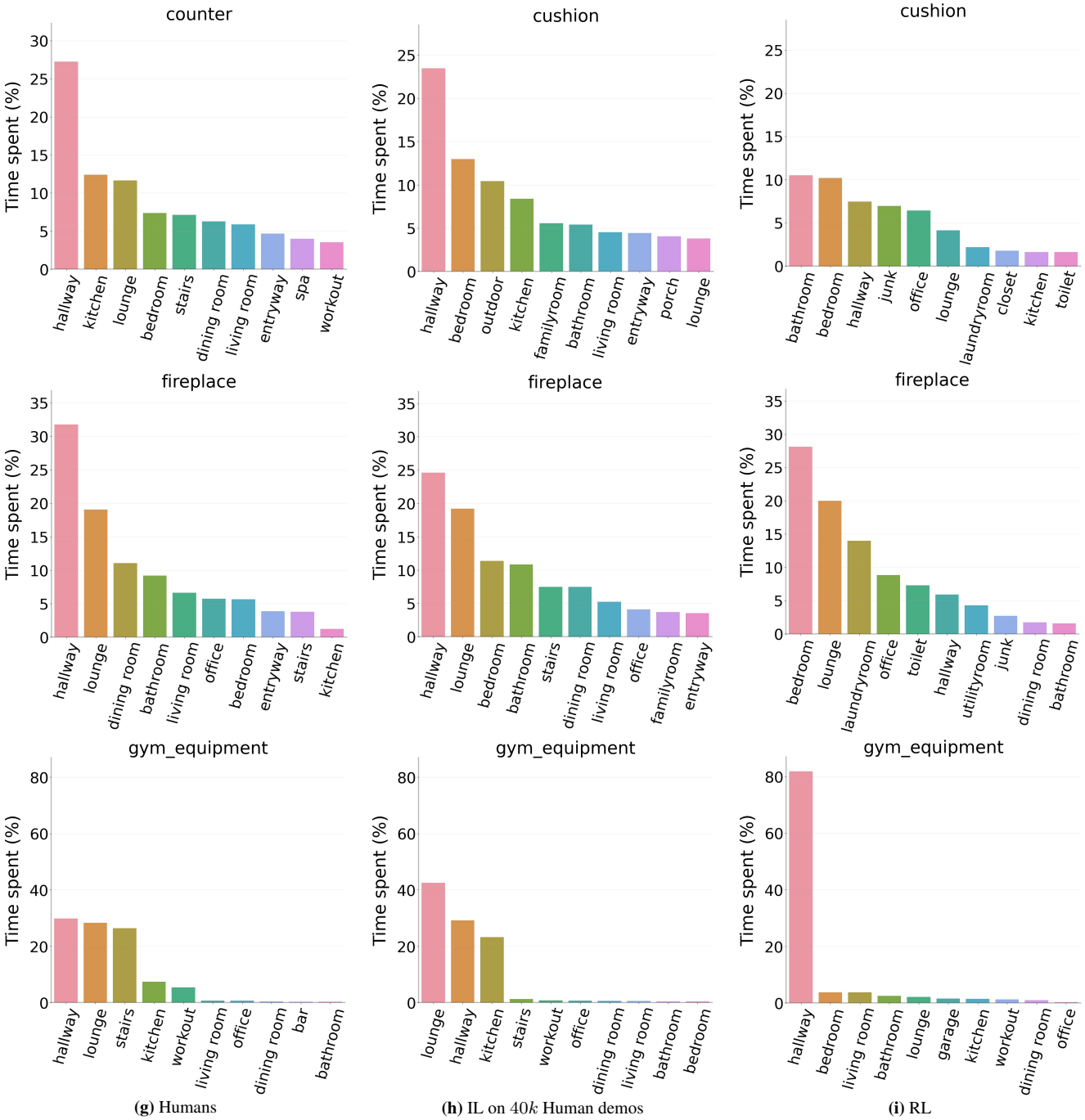

**(a)** Humans **(b)** IL on $40k$ Human demos **(c)** RL

**Figure 14.** Comparison of per room time spent for all MP3D goal categories on VAL split for human demonstrations *vs*. IL agents trained on human demos *vs*. RL agents. The plot shows the top 10 rooms ordered by the maximum time spent in each room.



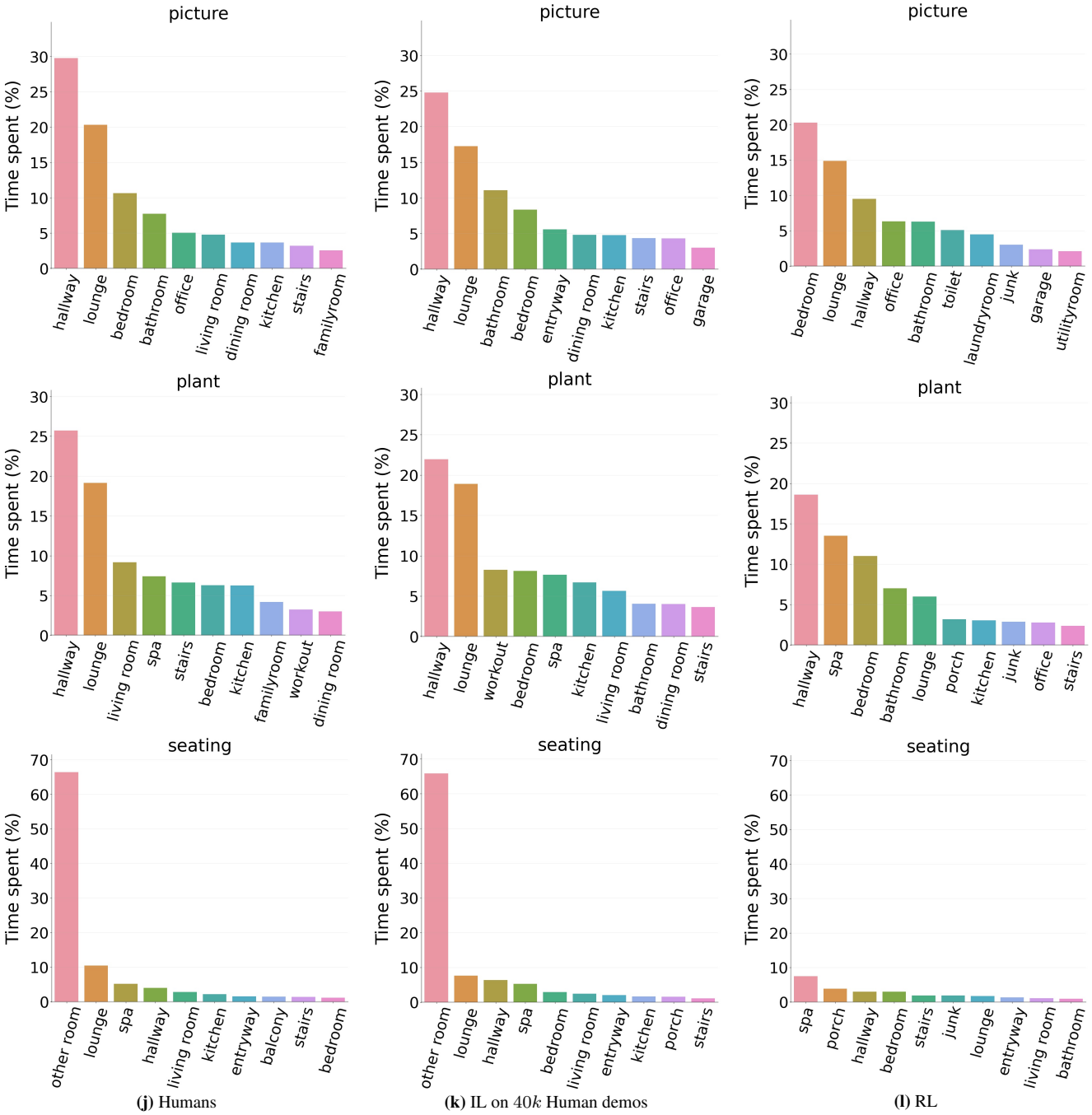

**(d)** Humans **(e)** IL on 40$k$ Human demos **(f)** RL

**Figure 14.** Comparison of per room time spent for all MP3D goal categories on VAL split for human demonstrations *vs*. IL agents trained on human demos *vs*. RL agents. The plot shows the top 10 rooms ordered by the maximum time spent in each room.



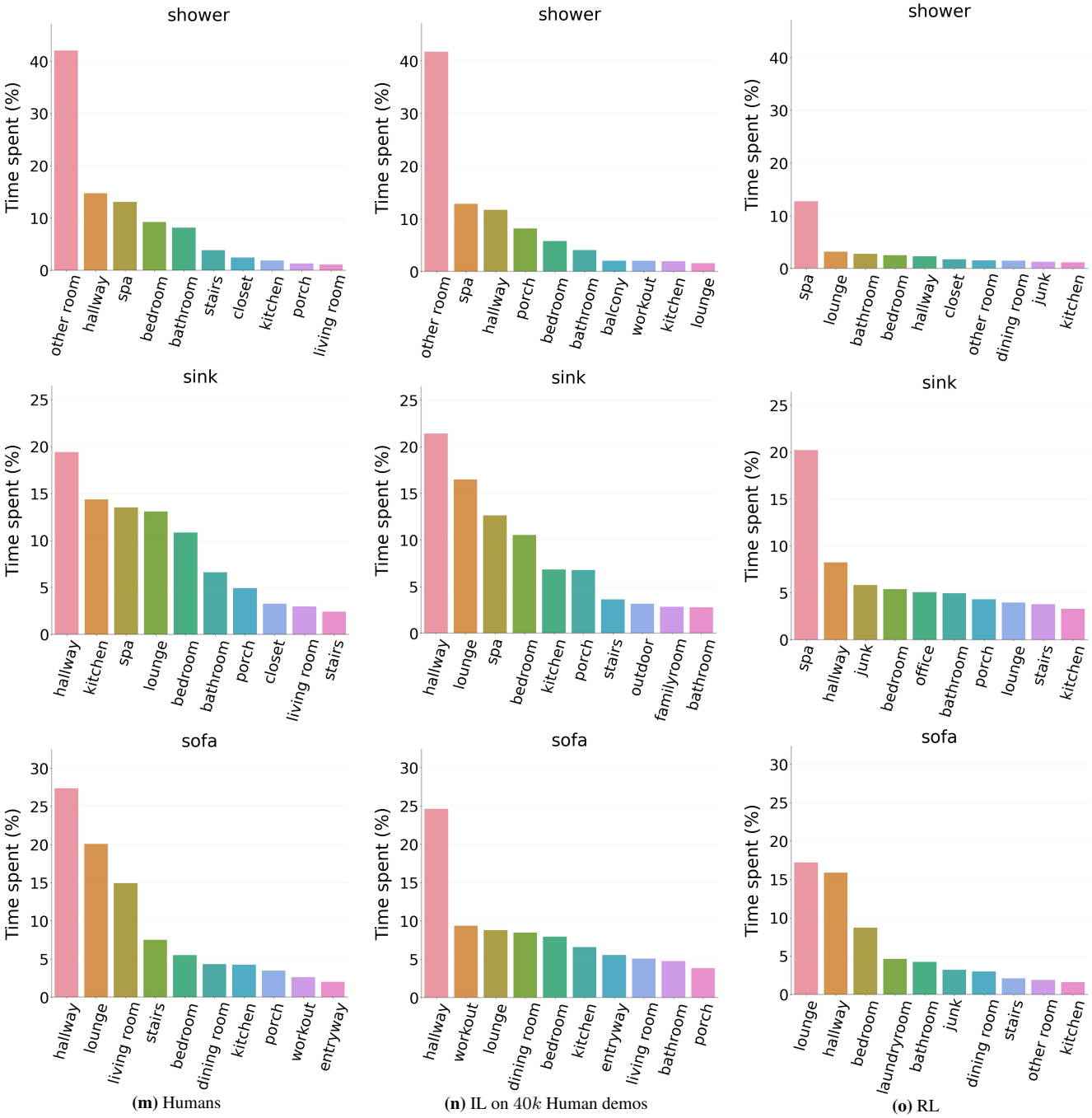

**(g)** Humans **(h)** IL on 40$k$ Human demos **(i)** RL

**Figure 14.** Comparison of per room time spent for all MP3D goal categories on VAL split for human demonstrations *vs*. IL agents trained on human demos *vs*. RL agents. The plot shows the top 10 rooms ordered by the maximum time spent in each room.



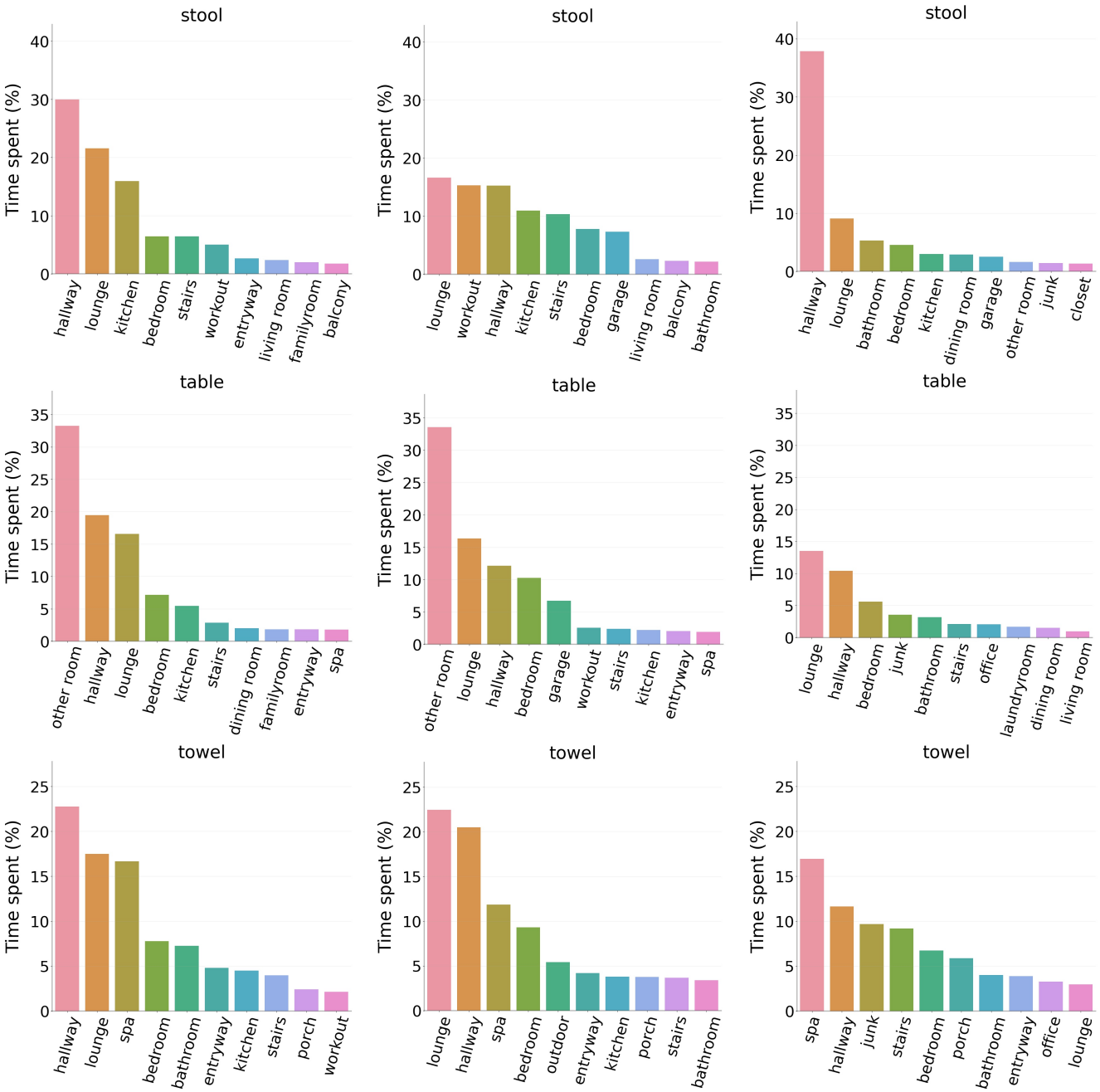

**(j)** Humans **(k)** IL on $40k$ Human demos **(l)** RL

**Figure 14.** Comparison of per room time spent for all MP3D goal categories on VAL split for human demonstrations *vs*. IL agents trained on human demos *vs*. RL agents. The plot shows the top 10 rooms ordered by the maximum time spent in each room.



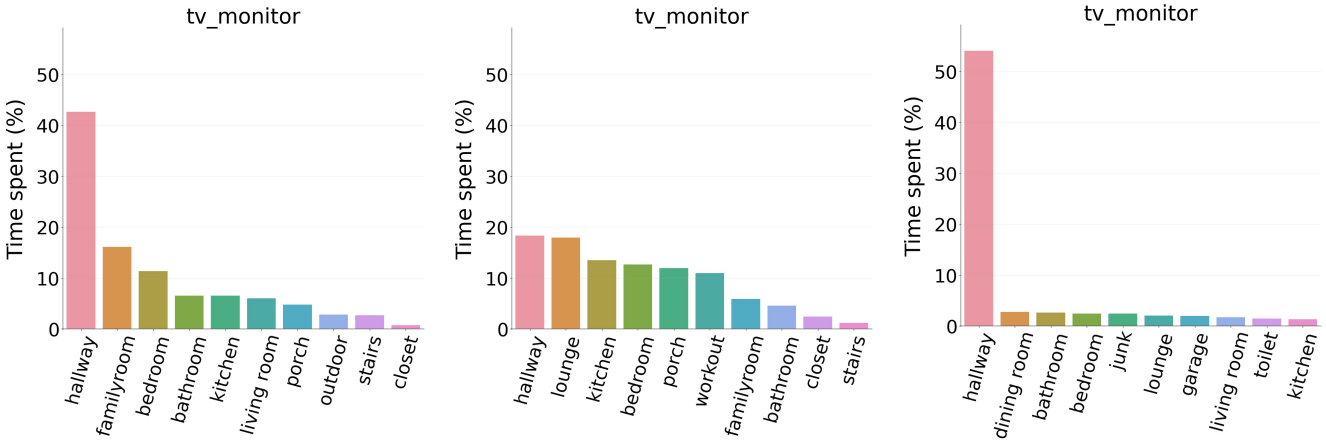

**(m)** Humans **(n)** IL on $40k$ Human demos **(o)** RL

**Figure 14.** Comparison of per room time spent for all MP3D goal categories on VAL split for human demonstrations *vs*. IL agents trained on human demos *vs*. RL agents. The plot shows the top 10 rooms ordered by the maximum time spent in each room.




**(p)** Humans **(q)** IL on $40k$ Human demos **(r)** RL

**Figure 14.** Comparison of per room time spent for all MP3D goal categories on VAL split for human demonstrations *vs*. IL agents trained on human demos *vs*. RL agents. The plot shows the top 10 rooms ordered by the maximum time spent in each room.




**(s)** Humans **(t)** IL on $40k$ Human demos **(u)** RL

**Figure 14.** Comparison of per room time spent for all MP3D goal categories on VAL split for human demonstrations *vs*. IL agents trained on human demos *vs*. RL agents. The plot shows the top 10 rooms ordered by the maximum time spent in each room.