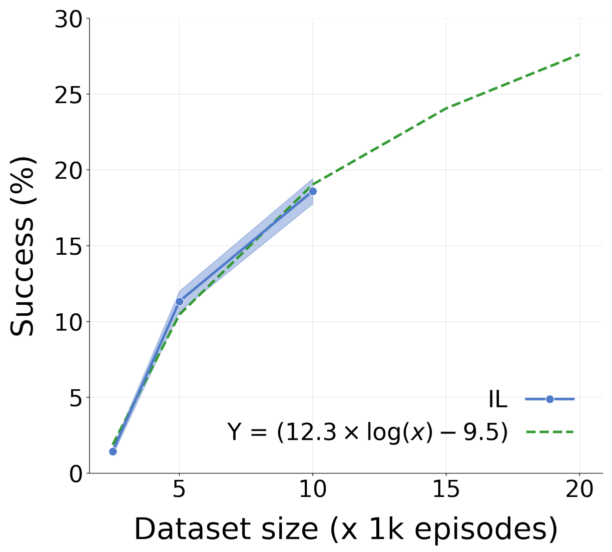


Figure 7. Dataset size vs performance plot for PICK&PLACE.

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 [42]. For depth, we use a ResNet50 that was pretrained on PointGoal navigation using DDPPO [13]. 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 [47]. 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.* [15]. 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 [15], 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.

$$\begin{aligned}
r_{\text{total}} &= r_{\text{success}} + r_{\text{slack}} + r_{\text{explore}} + r_{\text{grab\_release}} && \text{(2a)}\\
&\quad + r_{\text{drop\_penalty}} + r_{\text{seen}} && \text{(2b)}\\
r_{\text{success}} &= 5.5 \quad \text{on success} && \text{(2c)}\\
r_{\text{slack}} &= -10^{-4} \quad \text{per step} && \text{(2d)}\\
r_{\text{seen}} &= 1.5 \quad \text{First time object seen} && \text{(2e)}\\
r_{\text{drop\_penalty}} &= -3.5 \quad \text{Object dropped} > 2m \text{ away from receptacle} && \text{(2f)}\\
r_{\text{grab\_release}} &= 2.0 \quad \text{Grab / release success} && \text{(2g)}\\
r_{\text{explore}} &= 0.25 \times \frac{d^t}{v} \quad \text{Until object seen} && \text{(2h)}
\end{aligned}$$

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) Random	0.4%	0.4%
2) RGBD+RL [19]	8.2%	2.7%
3) RGBD+Semantics+RL [50]	15.9%	4.9%
4) Classical Map + FBE	40.3%	12.4%
5) Active Neural SLAM [51]	44.6%	14.5%
6) SemExp [48]	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.

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. 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 [41] 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, dining-table) from Matterport3D [39] 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 [48] (row 6 *vs*. row 7).

A.3. 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*. [28]. 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 [28] for all our time estimates, with the following rotation and translation equations:

$$y^{rotate} = 0.000358\phi^2 + 0.108\phi + 2.23 \tag{3}$$

$$y^{translate} = 4.2x + 0.362 \tag{4}$$

A.4. 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 $\sim$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 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. **??**) for human demonstrations *vs*. IL agents trained on human demonstrations *vs*. RL agents.

A.5. 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 $\sim$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*.

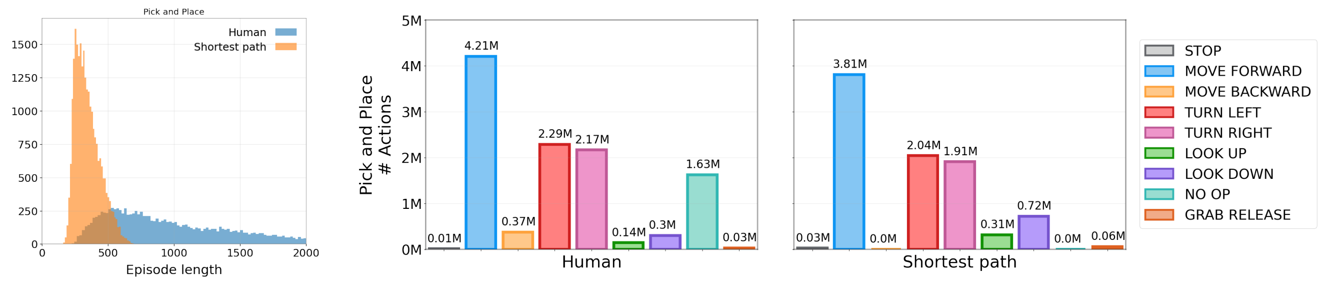


Figure 8. Comparison of episode lengths and action histograms for human demonstrations *vs*. shortest paths for PICK&PLACE. Human demonstrations are longer and have a more uniform action distribution than shortest paths.

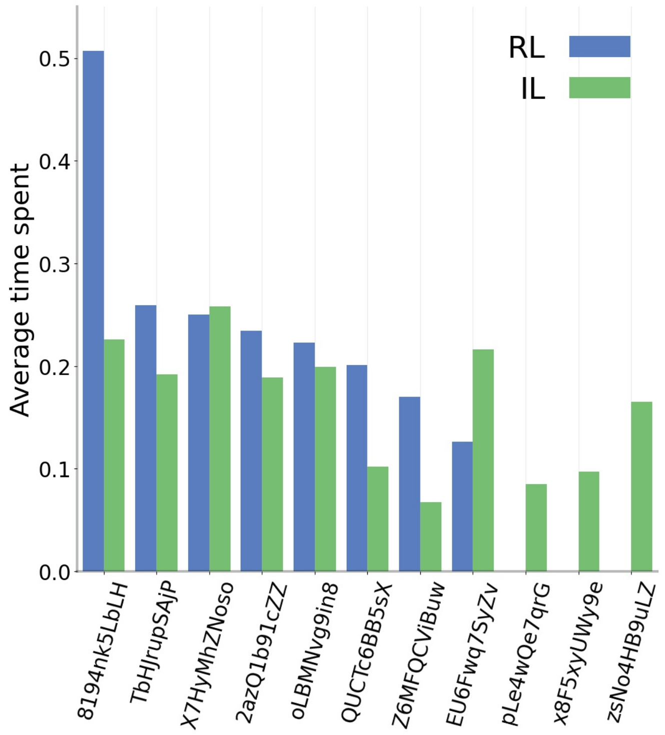


Figure 9. Per scene breakdown of GRTS for IL and RL agent on MP3D VAL split.

A.6. 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 environment, 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.

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.

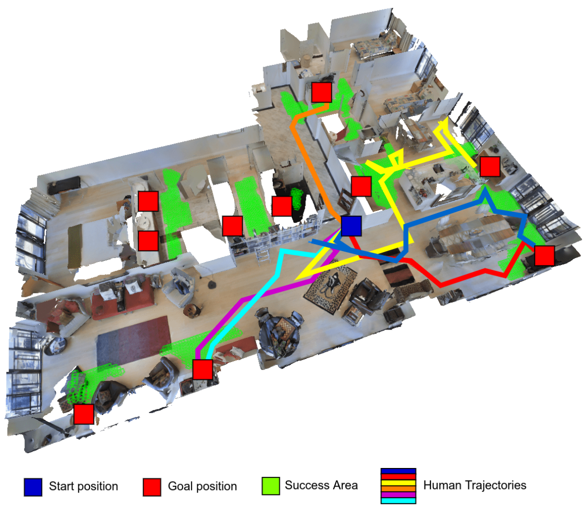


Figure 10. Visualizing multiple human demonstrations for OBJECTNAV all starting from the same start position and searching for 'cabinet'.

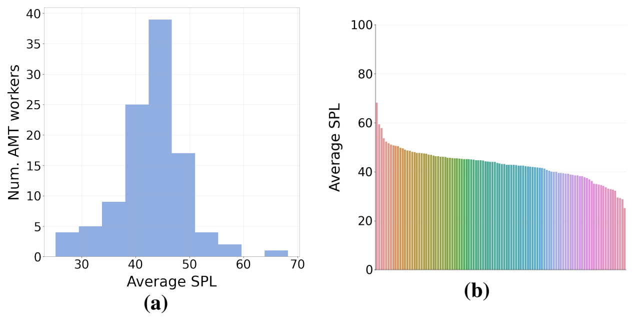
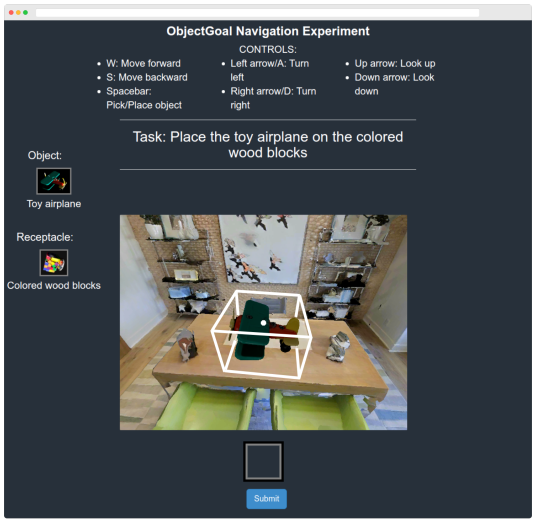


Figure 11. a) Histogram of average SPL for each AMT user for OBJECTNAV. b) Plot showing average SPL for AMT users for OBJECTNAV. These plots clearly demonstrate some humans are better at solving the OBJECTNAV task than others.


ObjectGoal Navigation Experiment

CONTROLS:

- W: Move forward
- S: Move backward
- Spacebar: Pick/Place object
- Left arrow/A: Turn left
- Right arrow/D: Turn right
- Up arrow: Look up
- Down arrow: Look down

Task: Place the toy airplane on the colored wood blocks

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.

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.

A.7. 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.

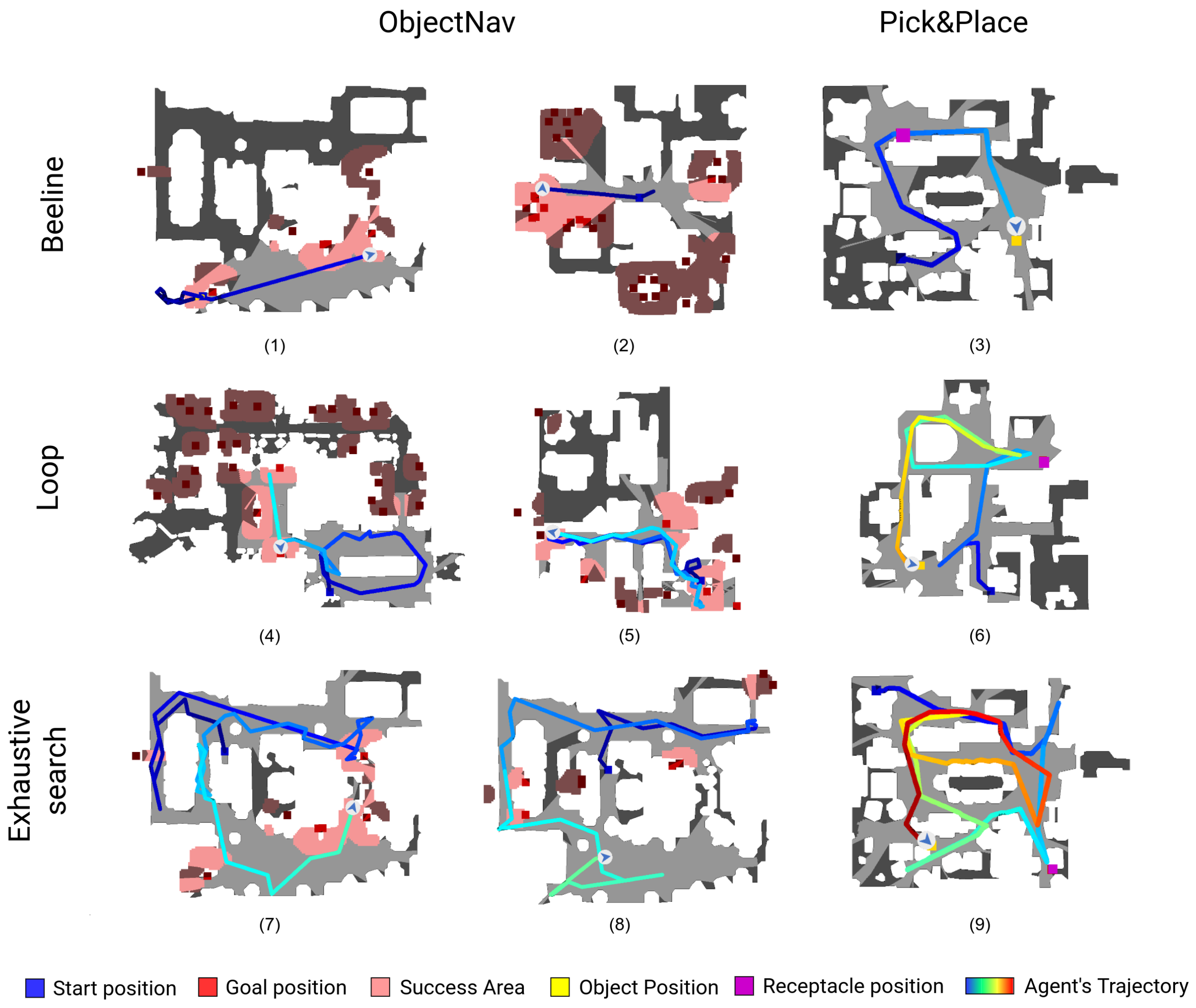


Figure 13. Visualizations of learnt agent behaviors for OBJECTNAV and PICK&PLACE. Best viewed in videos at sites.google.com/view/object-search-supp.

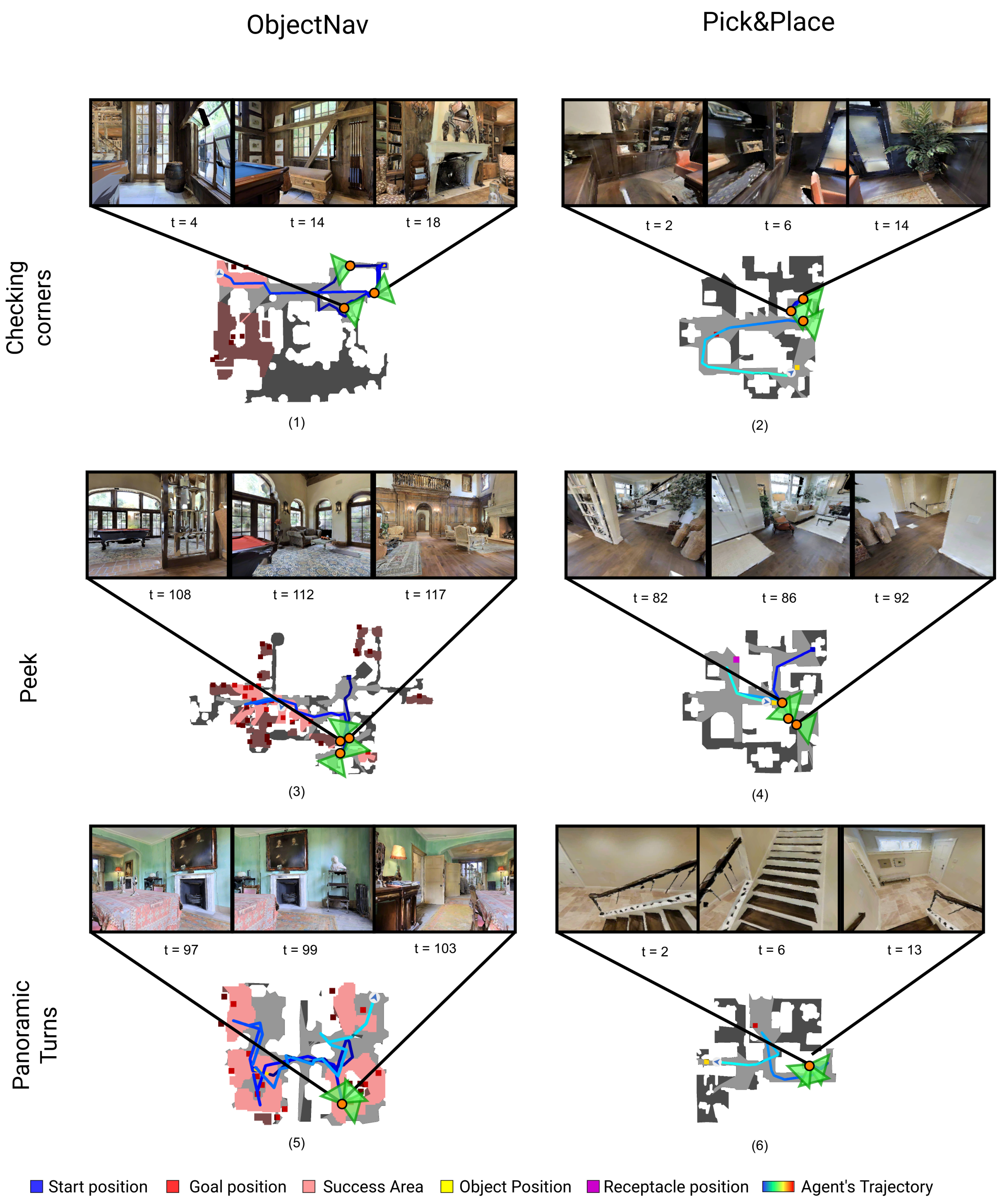
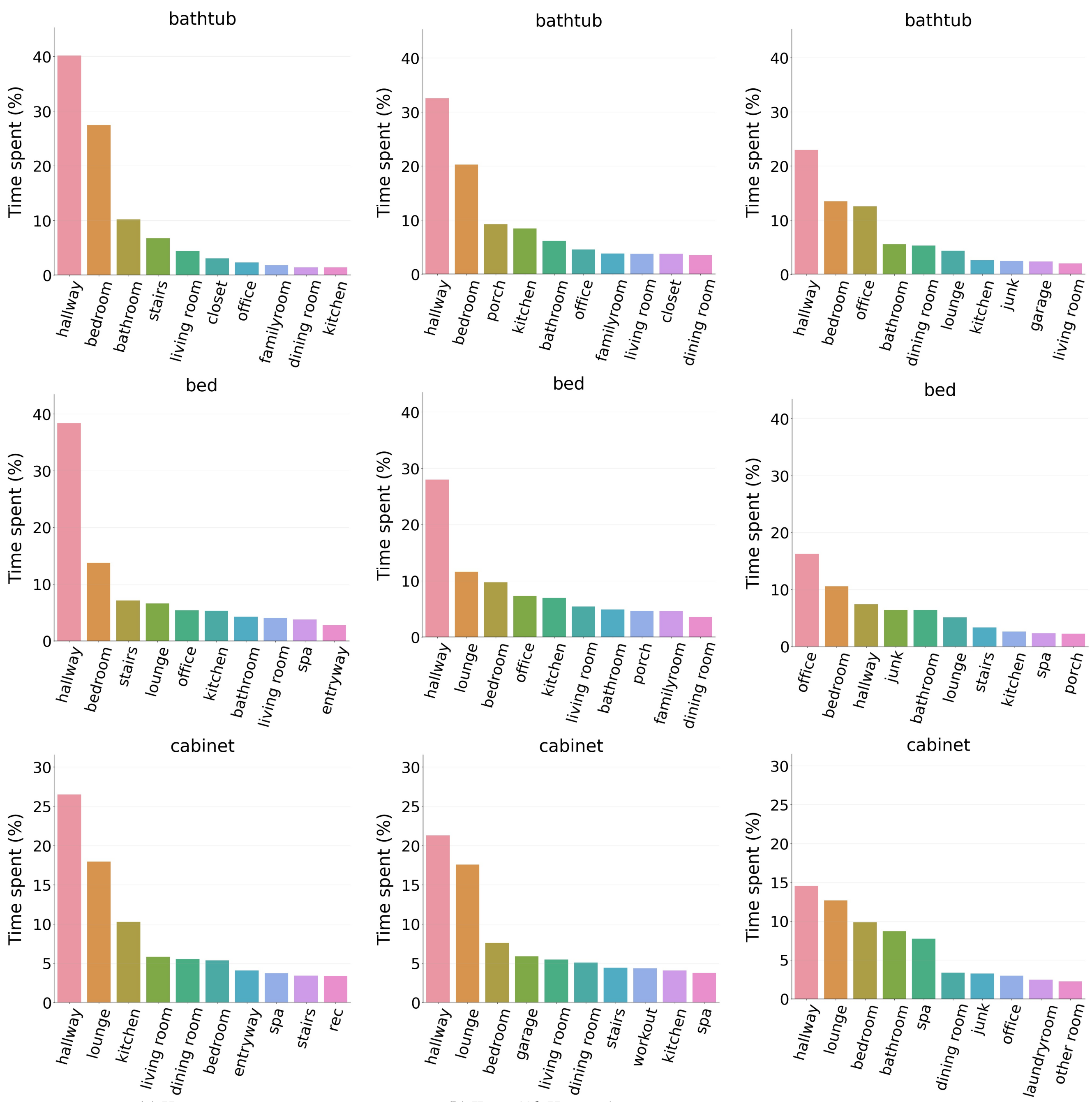
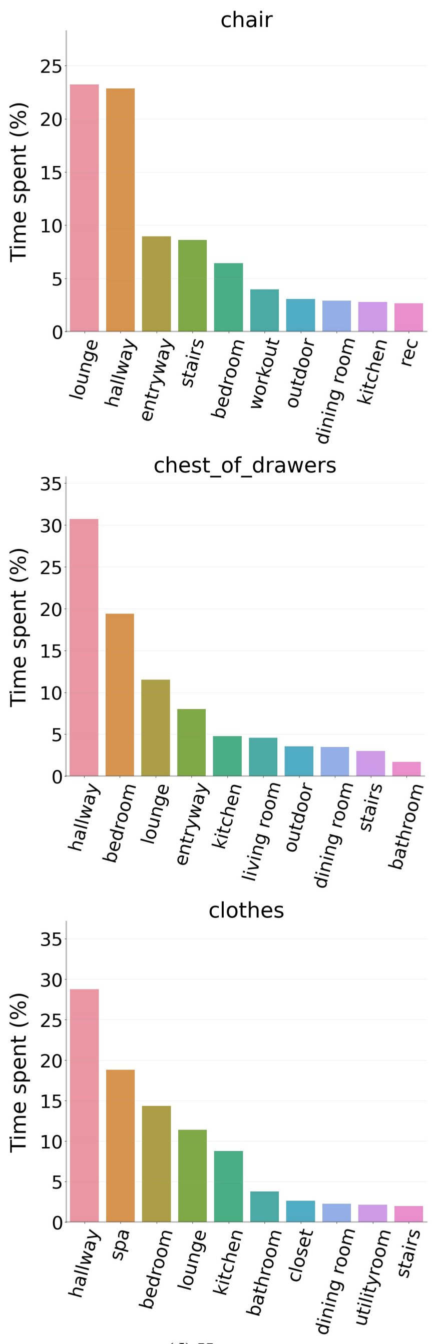


Figure 13. Visualizations of learnt agent behaviors for OBJECTNAV and PICK&PLACE. Best viewed in videos at ram81.github.io/projects/habitat-web.

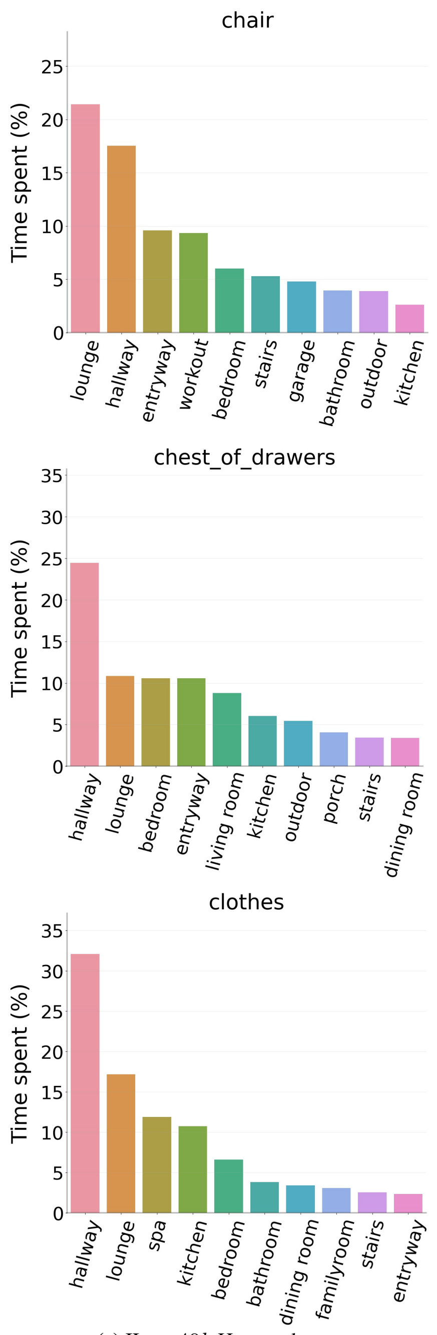


(a) Humans **(b)** IL on $40k$ Human demos **(c)** RL

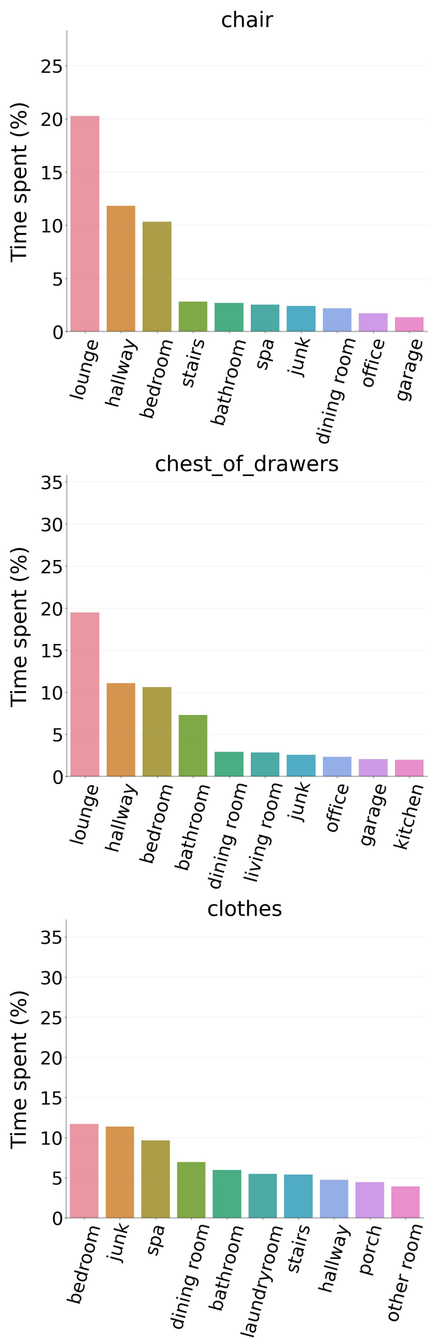
Figure 14. Comparison of per room time spent for all MP3D goal categories on VAL split for human demonstrations *vs*. IL agents trained on human demos *vs*. RL agents. The plot shows the top 10 rooms ordered by the maximum time spent in each room.



(d) Humans

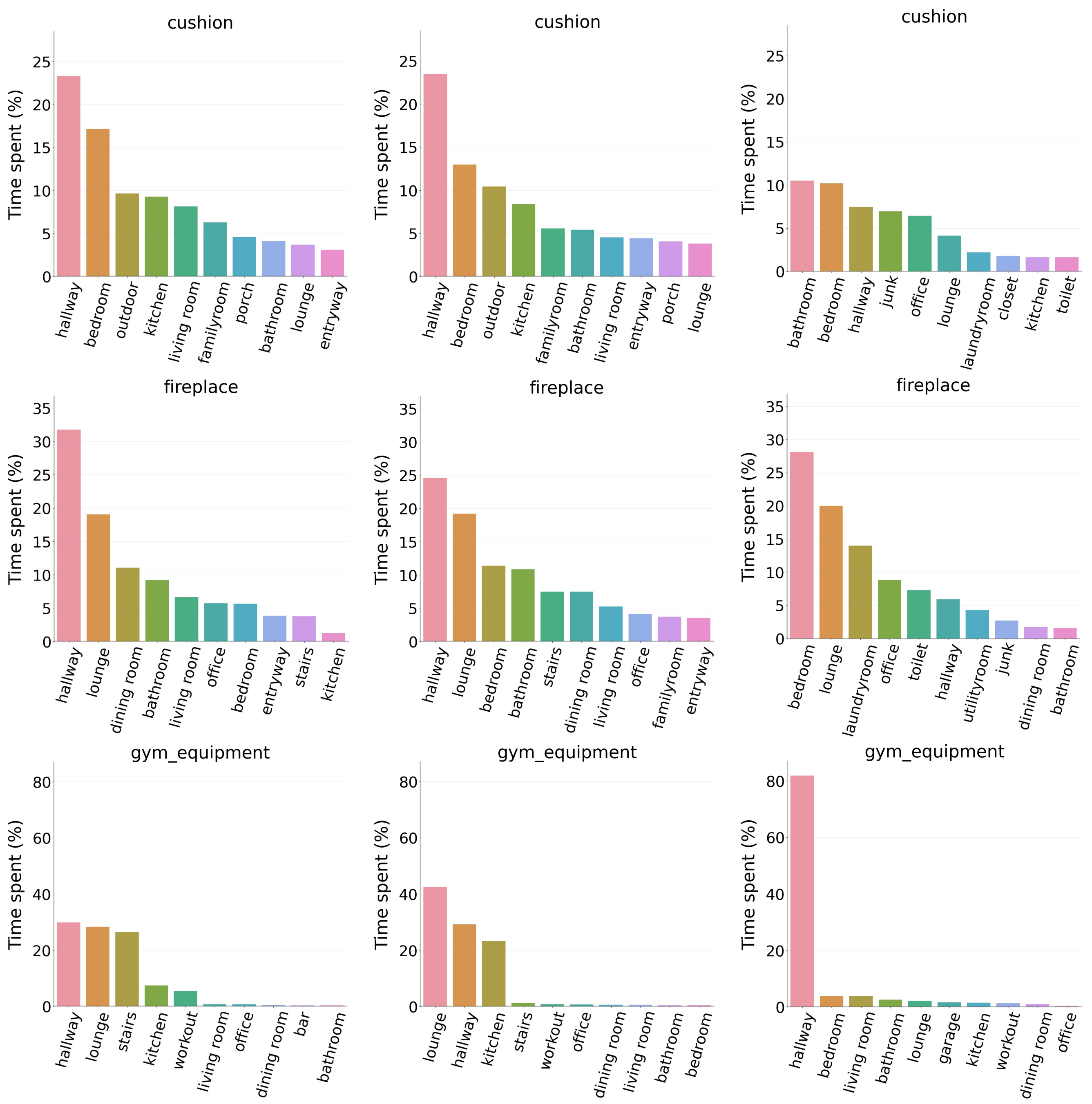


(e) IL on $40k$ Human demos



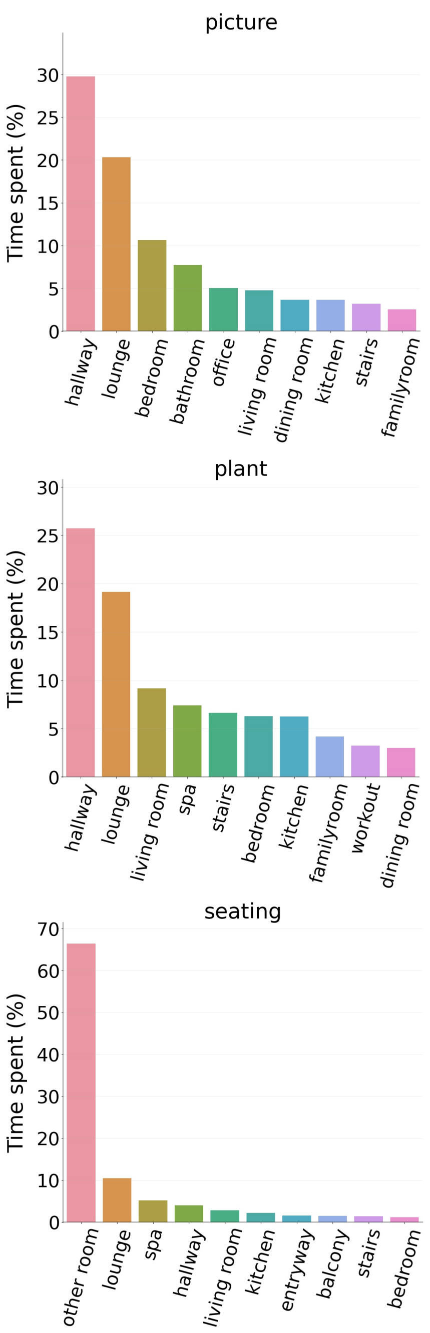
(f) RL

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.

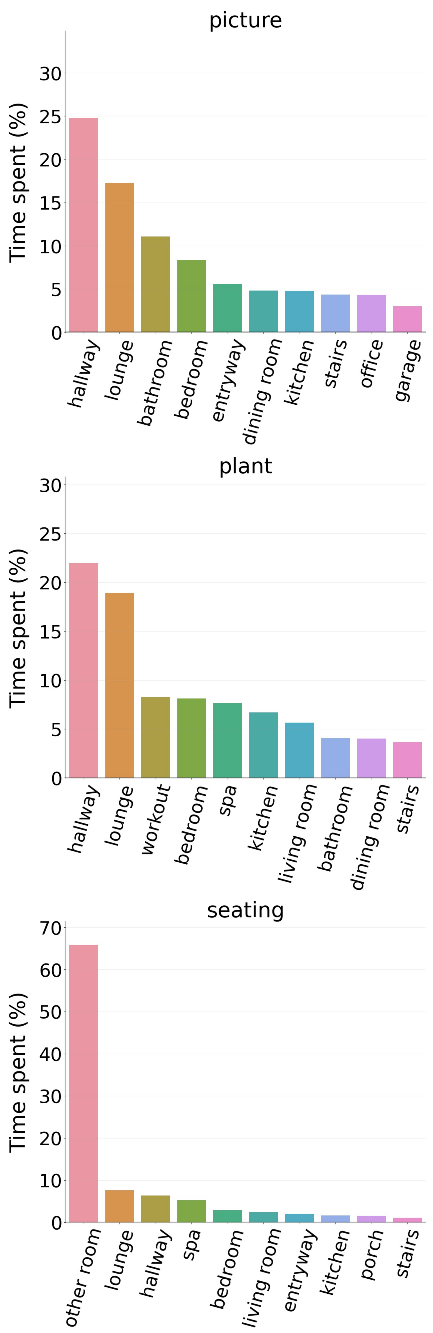


(g) Humans **(h)** IL on $40k$ Human demos **(i)** RL

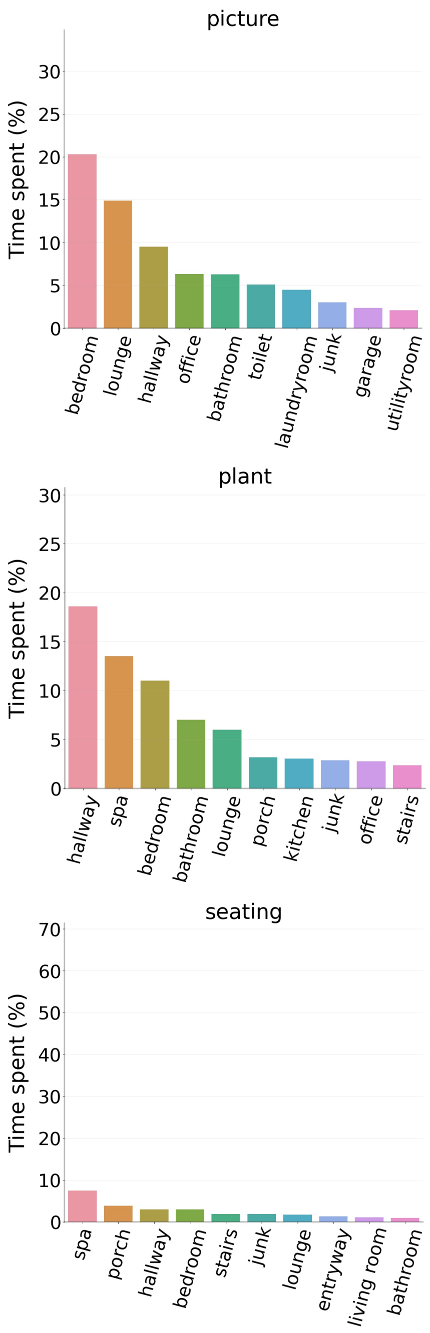
Figure 14. Comparison of per room time spent for all MP3D goal categories on VAL split for human demonstrations *vs*. IL agents trained on human demos *vs*. RL agents. The plot shows the top 10 rooms ordered by the maximum time spent in each room.



(j) Humans

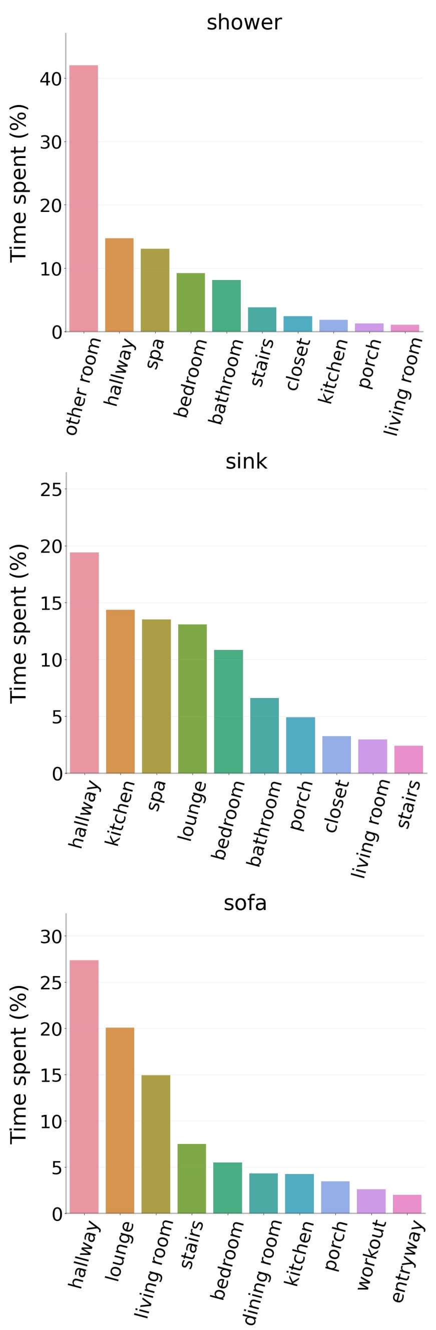


(k) IL on $40k$ Human demos

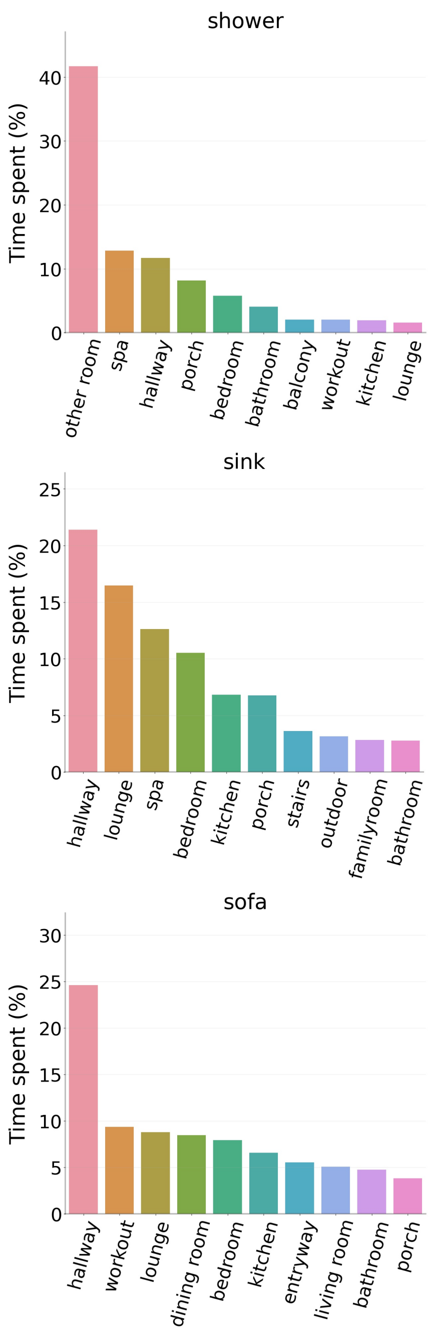


(l) RL

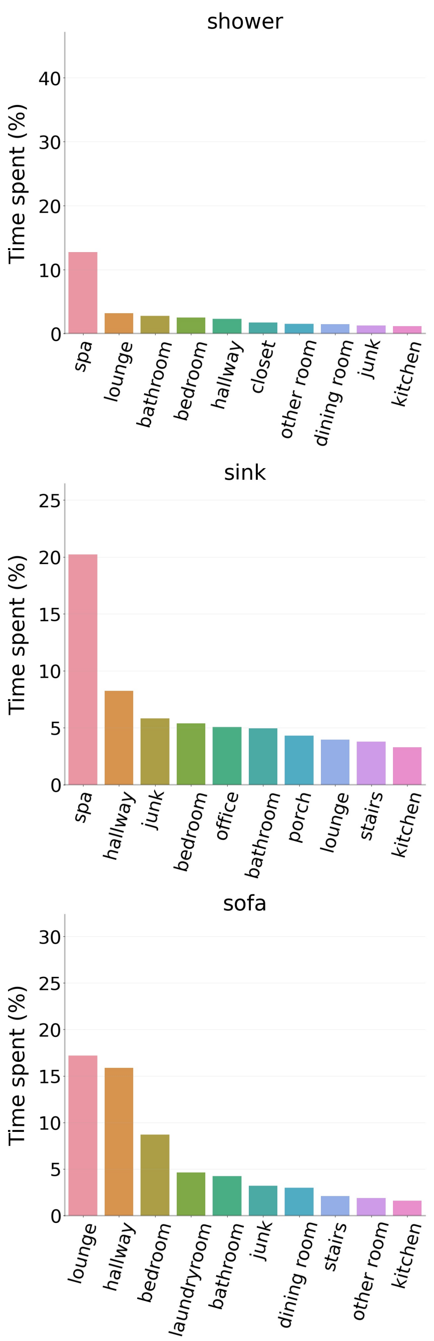
Figure 14. Comparison of per room time spent for all MP3D goal categories on VAL split for human demonstrations *vs*. IL agents trained on human demos *vs*. RL agents. The plot shows the top 10 rooms ordered by the maximum time spent in each room.



(m) Humans

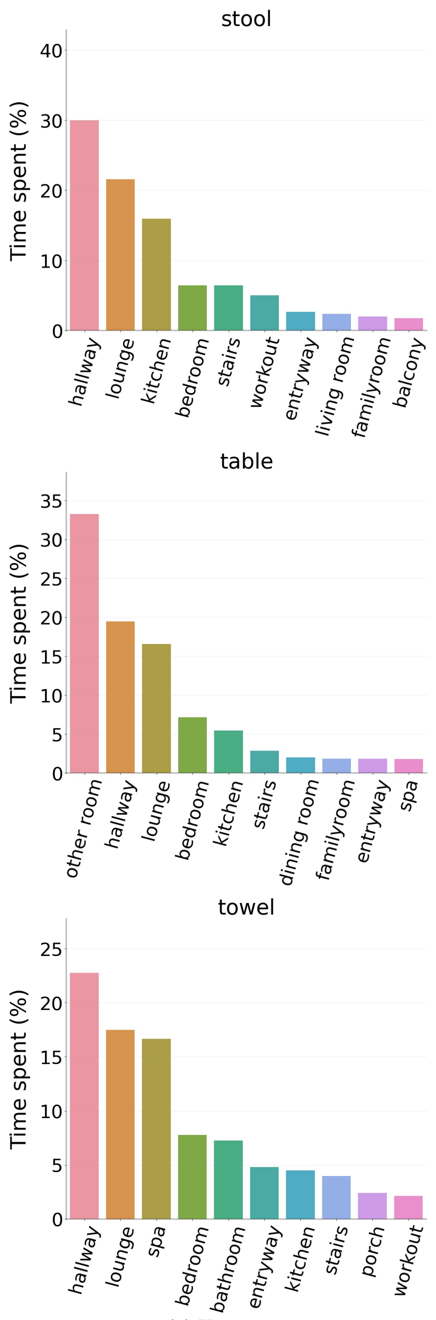


(n) IL on $40k$ Human demos

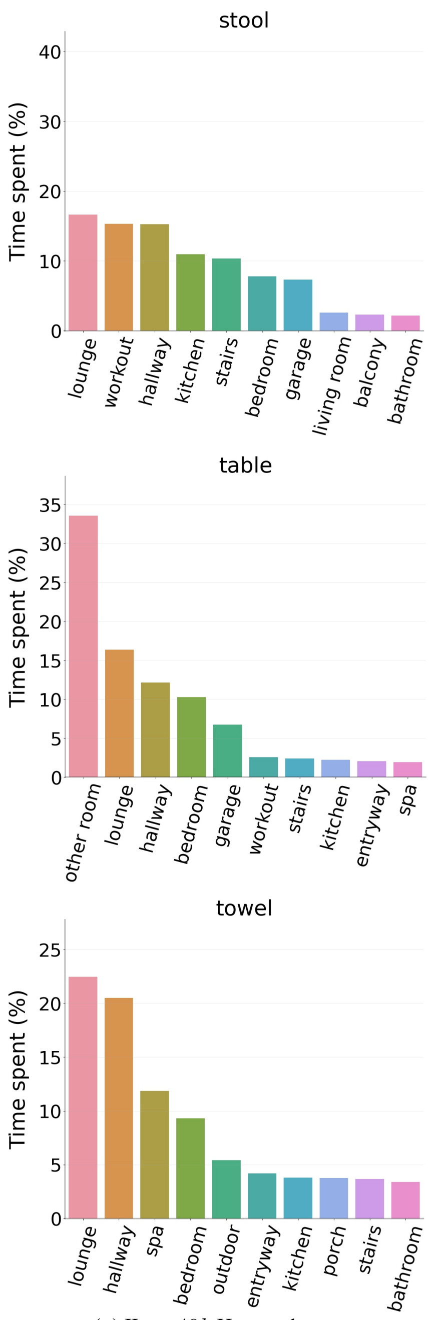


(o) RL

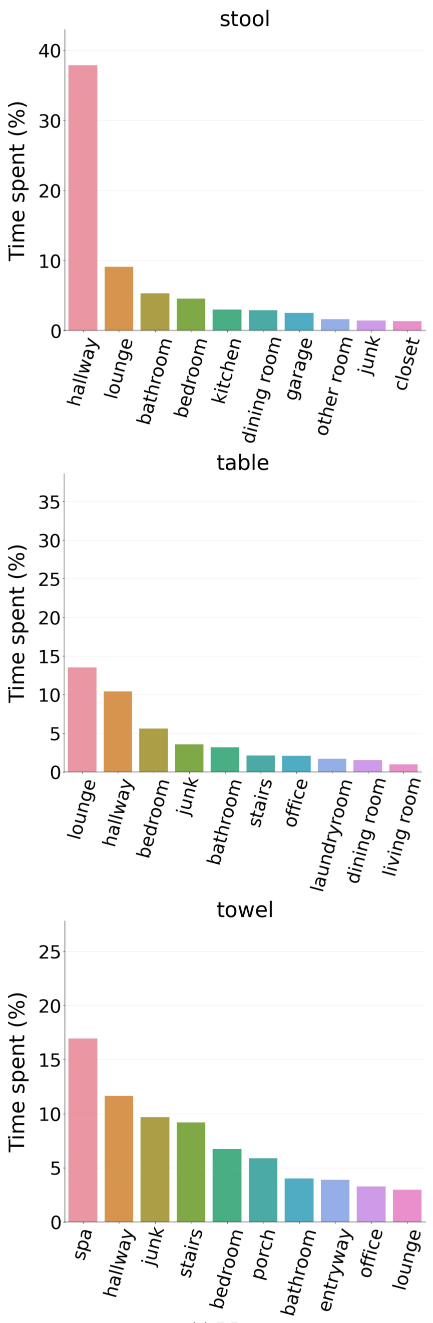
Figure 14. Comparison of per room time spent for all MP3D goal categories on VAL split for human demonstrations *vs*. IL agents trained on human demos *vs*. RL agents. The plot shows the top 10 rooms ordered by the maximum time spent in each room.



(p) Humans

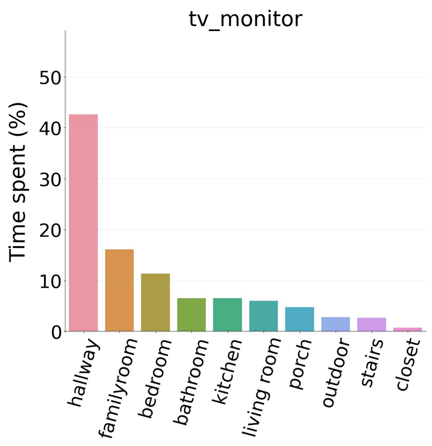


(q) IL on $40k$ Human demos

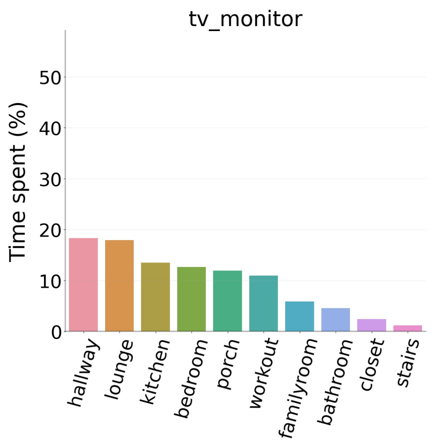


(r) RL

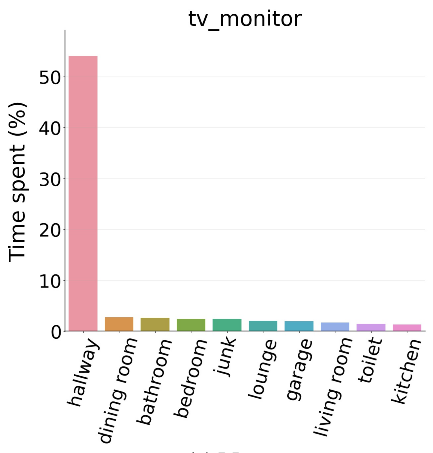
Figure 14. Comparison of per room time spent for all MP3D goal categories on VAL split for human demonstrations *vs*. IL agents trained on human demos *vs*. RL agents. The plot shows the top 10 rooms ordered by the maximum time spent in each room.



(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.