

APPENDIX

A IMPLEMENTATION DETAILS

We include the implementation details necessary for reproducing results for the tasks described in Section 4. Anonymous code for reproducing experimental results can be found here: <https://anonymous.4open.science/r/habitat3/README.md>

A.1 SOCIAL NAVIGATION

The robot uses neural network policies to find the humanoid, and the humanoid is scripted to navigate to random waypoints using a shortest path planner. We train all the end-to-end RL social navigation baselines using DD-PPO (Wijmans et al., 2019), distributing training across 4 NVIDIA A100 GPUs. Each GPU runs 24 parallel environments, and collects 128 steps for each update. We use a long short-term memory networks (LSTM) (Hochreiter & Schmidhuber, 1997) policy with ResNet18 as the visual backbone and two recurrent layers, resulting nearly 8517k parameters. We use a learning rate of 1×10^{-4} and the maximum gradient norm of 0.2. It takes about 200 million environment steps (roughly 4 days of training) to saturate. All baselines are trained with 3 different random seeds, and results are reported averaged across those seeds. Inspired by the reward design in training point-goal (Anderson et al., 2018), and object-goal policies (Batra et al., 2020), the social navigation reward is based on the distance to the humanoid at time t , and defined as follows:

$$r_t^{\text{distance}} = \begin{cases} \Delta(b_t, h_t) - \Delta(b_{t-1}, h_{t-1}), & \text{if } \Delta(b_t, h_t) \leq 1 \\ 2, & \text{if } 1 < \Delta(b_t, h_t) \leq 2 \\ \Delta(b_{t-1}, h_{t-1}) - \Delta(b_t, h_t), & \text{otherwise} \end{cases}$$

where $\Delta(b_t, h_t)$ is the geodesic distance between the robot location b_t and the humanoid location h_t at time t . The first condition encourages the robot to move away from the humanoid when it is closer than $1m$, ensuring that the agent maintains a safe distance from the humanoid. The second condition gives a constant reward for reaching within 1-2m, and the last condition rewards the robot to get closer to the humanoid.

To make sure the robot can face toward the humanoid, we further add an orientation reward when the robot is approaching the humanoid:

$$r_t^{\text{orientation}} = \begin{cases} (h_t - b_t) \cdot v_t^{\text{forward}}, & \text{if } \Delta(b_t, h_t) \leq 3 \\ 0, & \text{otherwise} \end{cases}$$

where v_t^{forward} is the robot normalized forward vector in the world frame, and the vector $(h_t - b_t)$ is also normalized.

During training, the episode terminates if there is a collision between the humanoid and the robot. The robot receives a bonus reward of +10 if the robot successfully maintains a safety distance between $1m$ and $2m$ to the humanoid and points to the humanoid for at least 400 simulation steps. The criteria for ‘facing the humanoid’ is computed by the dot product of the robot’s forward vector and the vector pointing from the robot to the humanoid, with the threshold of > 0.5 . We assume the robot has access to an arm depth camera (224×171 with the horizontal field of view (hFOV) of 55), an arm RGB camera (480×640 with hFOV of 47), a binary human detector (1-dim), and the relative pose of the humanoid in polar coordinate system (2-dim). In addition, a slack reward of -0.1 is given to encourage the agent to find the humanoid as soon as possible. In all episodes, to make sure that the robot learns to find the humanoid, the robot location is initialized at least $3m$ away from the humanoid. The final social navigation reward is as follows:

$$r_t^{\text{social-nav}} = 10\mathbb{1}^{\text{success}} + r_t^{\text{distance}} + 3r_t^{\text{orientation}} - 0.1.$$

During evaluation, the total episode length is 1500 steps and the episode terminates if there is a collision between the humanoid and the robot. In addition, the robot location is initialized at least $5m$ away from the humanoid, and the initial locations of the robot and the humanoid, and the humanoid path are fixed across the baselines. This ensures that we have a fair comparison across different baselines.

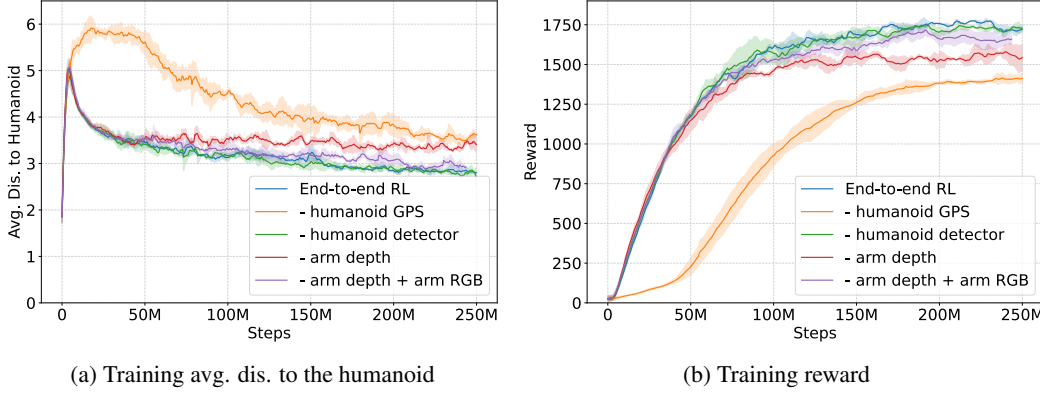


Figure 5: **Social Navigation training curves.** We plot the training average distance to the humanoid and reward for the social navigation baselines and ablations. We use 3 seeds for each model.

For the social navigation task, the output space of the policy is the linear and angular velocities with the range of -1 and $+1$ (2-dim), followed by scaling it to -10 and $+10$, which is equivalent to $2.5m/s$ in the real world. We use the same maximum linear and angular velocities of $2.5m/s$ (rad/s) for both humanoid and robot. Since the Spot robot has a long body shape and cannot be represented by a single cylinder, we use a 2-cylinder representation, placed in the center and the front of the robot for collision detection. This ensures that the robot arm camera does not penetrate walls or obstacles while allowing the robot to navigate in a cluttered scene.

Fig. 5 shows the average distance between the humanoid and the robot and reward learning curve over the number of simulation steps for the end-to-end RL policy and its ablations. We see that the agent is able to improve the reward while minimizing the distance to the humanoid for finding and following the humanoid over training.

Oracle for the Minimum Steps. To compute the Finding Success Weighted by Path Steps and Following rate, we need to measure the optimal finding time l . This measure is similar to the optimal path length in navigation tasks, but in this case the target to navigate to is dynamic. We thus define the optimal path length l as the minimum time that it would take for an agent to reach the humanoid if it knew the humanoid’s trajectory in advance. Formally, let h_i be the humanoid position at step i and r_i the minimum number of steps to go from the robot starting position to h_i , we define l as:

$$l = \arg \min_i (r_i < i), \quad (1)$$

measuring the earliest time where the robot will be able to find the humanoid. To compute this measure, we split the humanoid trajectory into equally spaced waypoints, we then use a path planner to measure the number of steps it would take to reach each waypoint from the robot starting position and take the earliest waypoint satisfying Eq. 1. Given this measure, the optimal following time corresponds to that of a robot which can find the humanoid in l and follow it until the end of the episode, i.e. $E - l$, with E being the episode length.

A.2 SOCIAL REARRANGEMENT

We train all the rearrangement baselines using DD-PPO (Wijmans et al., 2019), distributing training across 4 NVIDIA A100 GPUs. Each GPU runs 24 parallel environments, and collects 128 steps for each update. We train with Adam (Kingma & Ba, 2014) using a learning rate of $2.5e^{-4}$. We use 2 PPO minibatches and 1 epoch per update, an entropy loss of $1e^{-4}$, and clip the gradient norm to 0.2. All policies are trained for 100M environment steps. The policy uses a ResNet-18 (He et al., 2016) visual encoder to embed the 256×256 depth input image into a 512 dimension embedding. The visual embedding is then concatenated with the state sensor values and passed through a 2-layer LSTM network with hidden dimension 512. The LSTM output is then set to an action and value prediction network. All methods use the same reward function specified as $+10$ for succeeding in the overall task, $+5$ for completing any subgoal consisting of picking one of the target objects or placing an object at its goal, -0.005 penalty per simulator timestep to encourage faster completion, and a -5

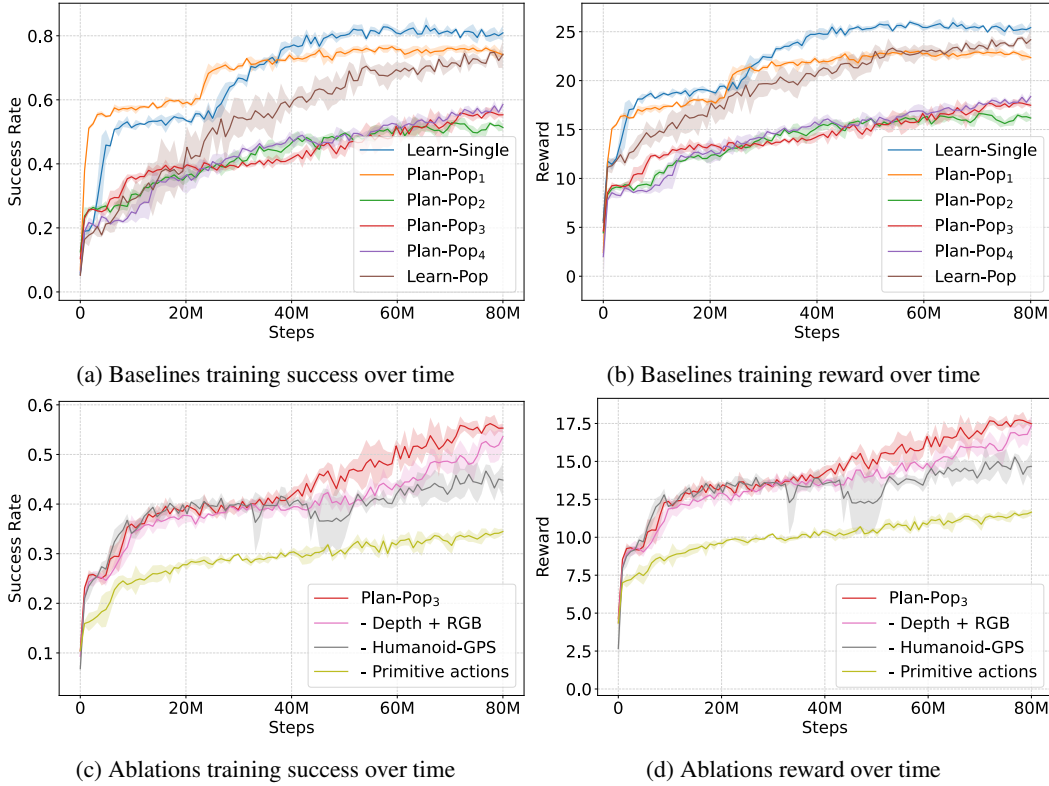


Figure 6: **Social Rearrangement training curves.** We plot the training success and reward for the social rearrangement baselines (**top**) and ablations (**bottom**). We use 3 seeds for each model.

penalty and episode termination if the agents collide. All baselines are trained with three different random seeds, and results are reported averaged across those seeds.

The final social rearrangement reward is as follows:

$$r_t^{\text{social-rearrange}} = 10 \cdot \mathbb{1}^{\text{success}} + 5 \cdot \mathbb{1}^{\text{subgoal}} - 5 \cdot \mathbb{1}^{\text{collision}} - 0.005.$$

Fig. 6 shows learning curves for all baselines and ablations on the social rearrangement task. We present the overall task success as well as the training reward for all approaches, averaged over 3 seeds. We observe that some baselines like Learn-Single and Plan-pop₁ are able to learn the task much faster than other baselines like Plan-pop_{2,3,4} due to a simpler training setting. Among the ablations, removing the sensors used in original training make learning slower, with primitive actions having the most effect.

Learned and Oracle skills. For ease of learning, we adopt a two-layer policy architecture for all baselines, where a learned high-level policy selects a low-level skill to execute based on observations. The action space of the learned high-level policy consists of discrete selections from all possible combinations of skills and objects/receptacles allowed at each step. We work with a known, fixed library of low-level skills that can accomplish instructions like “navigate to the fridge” or “pick an apple.” For the robot, we consider both learned skills and oracle skills that use privileged information from the environment. Additionally, we provide 4 ‘primitive’ actions to the high-level policy that move the robot forward/backward or turn it left/right by a fixed amount. During training, we use the oracle low-level skills to train the high-level policies due to faster training speed, but present evaluation results with both oracle and learned skills. For the humanoid, low-level skills are always oracle, as described in Section 3.1.

The oracle navigation skill has access to a map of the environment, and plans a shortest path to the agent’s destination. Manipulation oracle skills like “pick or place an apple” instantaneously attach

the apple to the gripper, or place it at the target location. If the high-level policy chooses to execute an infeasible low-level action, such as attempting to pick an object that is out of reach (based on predefined pre-conditions), the action results in a no-op with no changes to the environment. If the robot approaches the humanoid within a short distance ($< 1.5m$) while executing a skill, the current skill is aborted, and the high-level policy replans its next action. This results in reactive behaviors, like the high-level policy commanding the robot to move backwards to give way to the humanoid in narrow corridors, or choosing to pick the second object after realizing that the humanoid is picking the first.

When using learned skills, we use the same 2-layer policy architecture, except use learned navigation, and learned pick/place skills, which operate entirely using robot depth and onboard sensors. These skills do not use privileged information, and hence are more prone to failures in the diverse set of scenes considered in our tasks. Refer to Section B for more detail on training and design of low-level skills.

When evaluating social rearrangement with learned low-level skills, we keep the high-level policy frozen, after training it with oracle skills. Hence the high-level policy is not robust to low-level execution failures. As a result we observe a considerable drop in the overall performance, when using learned skills (Table 3). This performance can potentially be improved by training the high-level policy with learned low-level skills in-the-loop, or by fine-tuning in this setting. We leave this to future work.

B LOW-LEVEL SKILL TRAINING

We include the implementation details for training the low-level skills: navigation, pick, and place skills, which are coordinated by the high level policy described in Section 4.2.

Navigation Skill. Given the location of the target object, the robot uses neural network policies⁴ to find the object, similar to PointNav (Anderson et al., 2018). This navigation skill is different from the social navigation policy since the former does not require the robot to continuously follow the humanoid while avoiding collision. The robot has access to an arm depth camera (224×171 with hFOV of 55), and the relative pose of the target object in polar coordinate system (2-dim). Similar to the policy used in the social navigation task, the output space of the navigation skill is the linear and angular velocities with the range of -1 and $+1$ (2-dim), followed by scaling to -10 and $+10$. We also use the same 2-cylinder collision shape representation as the one in the social navigation task to ensure that the robot arm camera does not penetrate walls or obstacles while allowing the robot to navigate in a cluttered scene.

During training, the object navigation reward that encourages moving forward the target object r_t^{distance} at time t is defined as $\Delta(b_{t-1}, e_{t-1}) - \Delta(b_t, e_t)$, where $\Delta(b_t, e_t)$ is the shortest path distance between the robot location b_t and the object e_t at time t . To encourage the robot to orient itself toward the target object for improving grasping, it receives an additional ‘orientation’ reward $r_t^{\text{orientation}}$ when $\Delta(b_{t-1}, e_{t-1}) - \Delta(b_t, e_t) \leq 3$, in which the orientation reward penalizes the dot product of the robot forward vector and the robot to target object vector weighted by the scale of 5×10^{-2} . A navigation success reward of $+10$ is given if (1) the distance between the agent and the target object is less than $1.5m$, and (2) the dot product of the robot forward vector and the robot to target object vector > 0.5 . To reduce the collision between the robot and the scene, a penalty of -5×10^{-3} is given if there is a collision. Finally, a slack reward of -1×10^{-2} is given to encourage the robot to find the target object as soon as possible. The final navigation reward is as follows:

$$r_t^{\text{nav}} = 10\mathbb{1}^{\text{success}} + r_t^{\text{distance}} + 0.05r_t^{\text{orientation}} - 0.005\mathbb{1}_t^{\text{collision}} - 0.01.$$

During training, the episode terminates if the robot finds the object or reaches the maximum episode simulation step of 1500. In addition, the episode also terminates if the robot collides with a humanoid that walks randomly, similar to the setup in training social navigation policies. To make sure that the robot learns to navigate in complex scenes, the robot is placed at least $4m$ away from the target object location. We train the navigation skill using DD-PPO, distributing training across 4 NVIDIA A100 GPUs, and with learning rate of 1×10^{-4} and the maximum gradient norm of 0.2. Each GPU runs

⁴In this section, we use ‘policy’ and ‘skill’ interchangeably to refer to a controller parameterized by neural networks. In addition, we use ‘robot’ and ‘agent’ interchangeably to refer to a reinforcement learning agent.

24 parallel environments, and collects 128 steps for each update. We use a long short-term memory networks (LSTM) policy with ResNet-18 (He et al., 2016) as the visual backbone and two recurrent layers, resulting nearly 8517k parameters. It takes about 300 million simulation steps (roughly 6 days of training) to reach 90% navigation success rate using the above hardware setup.

Pick Skill. Given the location of the target object, the robot uses neural network policies to pick up an object by controlling the arm and moving the base (i.e., mobile pick, as defined in Gu et al. (2023)). The robot has access to (1) an arm depth camera (224×171 with hFOV of 55), (2) the relative pose of the target object in a Cartesian coordinate system (3-dim), (3) the arm joint angles (7-dim), (4) a binary holding detector if the robot is holding an object (1-dim), and (5) the relative pose of arm end-effector to the target resting location in a Cartesian coordinate system (3-dim). The output space of the pick skill is (1) the linear and angular base velocities with the range of -1 and $+1$ (2-dim), followed by scaling to -10 and $+10$, (2) the delta arm joint angles applied to the arm with the range of -1 and $+1$ (7-dim), followed by a scaling factor of 5×10^{-2} , and a binary command to snap/desnap the object to the end-effector (1-dim). The robot can only pick up the object if the distance between the end-effector and the object is less than $0.15m$ (we teleport the object to the end-effector to simulate the grasping).

During training, before picking up the object, the pick reward that encourages the arm to move toward the object r_t^{move} at time t is defined as $\Delta(c_{t-1}, e_{t-1}) - \Delta(c_t, e_t)$, where $\Delta(c_t, e_t)$ is the geodesic distance between the robot’s arm end-effector location c_t and the object e_t at time t . After picking up the object, the retract-arm reward that encourages the robot to retract the arm r_t^{retract} at time t is defined as $\Delta(c_{t-1}, q_{t-1}) - \Delta(c_t, q_t)$, where $\Delta(c_t, q_t)$ is the distance between the robot’s arm end-effector location c_t and the target end-effector resting location q_t at time t . The robot receives the success reward of $+2$ if the robot (1) picks up the right target object, and (2) $\Delta(c_t, q_t)$ is less than $0.15m$. Finally, a slack reward of -5×10^3 is given to encourage the robot to pick up the target object as soon as possible. The final pick reward is as follows:

$$r_t^{\text{pick}} = 2\mathbb{1}^{\text{success}} + r_t^{\text{move}} + r_t^{\text{retract}} - 0.005.$$

During training, the episode terminates if the robot (1) picks up the wrong object or drops the object, both with the penalty of -0.5 , (2) reaches maximum simulation steps of 1250, or (3) successfully picks up the right target object and retracts the arm, with the success reward of $+2$. To make sure that the robot learns to orient itself to pick up the object, the robot is placed at least $3m$ away from the target object location. We train the pick skill using DD-PPO, distributing training across 8 NVIDIA GPUs, and with learning rate of 3×10^{-4} and the maximum gradient norm of 0.2. Each GPU runs 18 parallel environments, and collects 128 steps for each update. We use a long short-term memory network (LSTM) policy with ResNet-18 as the visual backbone and two recurrent layers, resulting nearly 8540k parameters. It takes about 100 million simulation steps (roughly one day of training) to reach 90% pick success rate using the above hardware setup.

Place Skill. Given the location of the goal location, the robot uses neural network policies to place an object by controlling the arm and moving the base (i.e., mobile place Gu et al. (2023)). For consistency, the input space of the place skill has the exact same input space as the pick skill. The output space of the place skill also shares the same output space of the pick skill. The robot can only place the object if the distance between the end-effector and the target place location is less than $0.15m$ (we teleport the object from the end-effector to the target place location).

During training, before placing the object, the place reward that encourages the robot to move close to the target location r_t^{move} at time t is defined as $\Delta(c_{t-1}, e_{t-1}) - \Delta(c_t, e_t)$, where $\Delta(c_t, e_t)$ is the distance between the robot’s arm end-effector location c_t and the object e_t at time t . After placing the object, the retract-arm reward r_t^{retract} is the same as the one in pick skill to learn to reset the arm. In addition, the robot receives an addition bonus reward r_t^{bonus} of $+5$ if the robot places the object in the right location. Finally, the robot receives the success reward of $+10$ if (1) the robot places the object in the right location, and (2) $\Delta(c_t, q_t)$ is less than $0.15m$. A slack reward of -5×10^3 is given to encourage the robot to place the object as soon as possible. The final place reward is as follows:

$$r_t^{\text{place}} = 10\mathbb{1}^{\text{success}} + r_t^{\text{bonus}} + r_t^{\text{move}} + r_t^{\text{retract}} - 0.005.$$

During training, the episode terminates if the robot (1) places the object in the wrong location, (2) reaches maximum simulation steps of 1250, or (3) successfully places the right target object and

	S \uparrow	SPS \uparrow	F \uparrow	CR \downarrow	BYR	TD \downarrow	FD \downarrow
Heuristic Expert	1.00	0.97	0.51	0.52	0.24	2.56	1.72
End-to-end RL	0.97 \pm 0.00	0.65 \pm 0.00	0.44 \pm 0.01	0.51 \pm 0.03	0.19 \pm 0.02	3.43 \pm 0.07	1.70 \pm 0.04
- humanoid GPS	0.76 \pm 0.02	0.34 \pm 0.01	0.29 \pm 0.01	0.48 \pm 0.03	0.13 \pm 0.00	5.18 \pm 0.11	1.64 \pm 0.02
- humanoid detector	0.98 \pm 0.00	0.68 \pm 0.00	0.37 \pm 0.01	0.64 \pm 0.05	0.16 \pm 0.03	3.44 \pm 0.03	1.67 \pm 0.09
- arm depth	0.94 \pm 0.01	0.54 \pm 0.01	0.19 \pm 0.01	0.71 \pm 0.08	0.15 \pm 0.01	4.94 \pm 0.03	1.91 \pm 0.27
- arm depth + arm RGB	0.96 \pm 0.00	0.61 \pm 0.01	0.38 \pm 0.02	0.55 \pm 0.04	0.17 \pm 0.02	3.74 \pm 0.05	1.82 \pm 0.05

Table 2: **Social Navigation baseline results.** We report three additional metrics: (1) *Backup-Yield Rate (BYR)*, (2) *The Total Distance between the robot and the humanoid (TD)*, and (3) *The ‘Following’ Distance between the robot and the humanoid after the first encounter (FD)*.

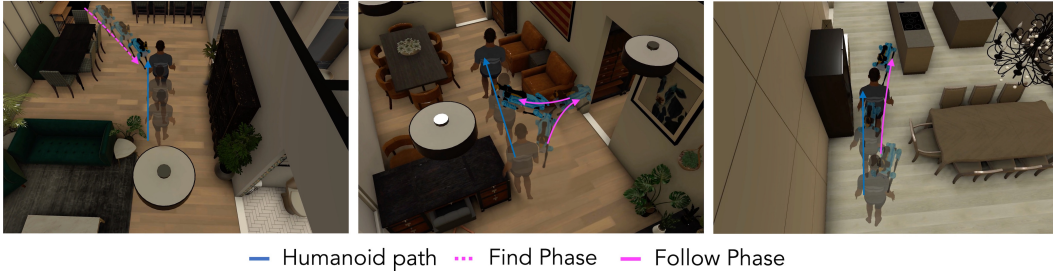


Figure 7: **Illustration of Social Navigation end-to-end RL policy’s behavior.** The robot finds the humanoid (**left**). The robot yields to the humanoid by doing a ‘three-point-turn’ motion (**center**). The robot yields to the humanoid by backing up (**right**).

retracts the arm, with the success reward of +10. To make sure that the robot learns to orient itself to place the object, the robot is placed at least 3m away from the target object location. Similar to the one in pick skill, we train the navigation skill using DD-PPO, distributing training across 8 NVIDIA GPUs, and with learning rate of 3×10^{-4} and the maximum gradient norm of 0.2. Each GPU runs 18 parallel environments, and collects 128 steps for each update. We use a long short-term memory network (LSTM) policy with ResNet-18 as the visual backbone and two recurrent layers, resulting nearly 8540k parameters. It takes about 50 million simulation steps (roughly half of a day of training) to reach 90% place success rate using the above hardware setup.

C DETAILED COMPARISON RESULTS

Here we provide additional detailed results comparing the performance of different baselines along additional metrics.

C.1 SOCIAL NAVIGATION

In this section, we provide a detailed analysis of the Social Navigation task, studying the behavior of the different baselines at evaluation time.

C.1.1 ADDITIONAL METRICS

In Table 2, we further report three additional metrics: (1) *Backup-Yield Rate (BYR)*: How often did the robot do a backup or yield motion to avoid collision when the human is nearby? We define a ‘backup motion’ as a backward movement by the robot to avoid collision with the humanoid when the distance between them is less than 1.5 meters. Furthermore, we define a ‘yield motion’ as a robot’s motion aimed at avoiding collision with the humanoid when the distance between them is less than 1.5 meters, and the robot’s velocity is less than 0.1m/s; (2) *The Total Distance between the robot and the humanoid (TD)*: What was the L2 distance (in meter) between the robot and the humanoid over the total number of episode steps?, and (3) *The ‘Following’ Distance between the robot and the humanoid after the first encounter (FD)*: What was the L2 distance between the robot and the humanoid after the robot finds the humanoid? In summary, the backup-yield rate lets us quantitatively measure the frequency of backup and yield motions, and the two distance matrices

provide an observation of how close the robot and the humanoid are during the finding and following stages. Ideally, the FD should be between 1-2m, while policies with higher SPS have lower TD.

C.1.2 ADDITIONAL ANALYSIS

In this section, we provide additional analysis for the end-to-end RL policy and its ablations. Fig. 7 provides an example of how the robot moves to find the humanoid and produces a backup motion to yield to the humanoid by anticipating where the humanoid will walk next. For the ablation baseline without the humanoid GPS, we observe that it learns two types of finding humanoid strategies. The first strategy is that the robot randomly walks to increase the chances of finding the humanoid. The second strategy is that the robot keeps rotating until there is a humanoid in sight (i.e., scanning the environment), captured by the humanoid detector. As a result, compared to the method with the humanoid GPS, the one without the humanoid GPS needs more steps to find the humanoid (lower SPS). It also tends to lose track of the humanoid when the humanoid walks into another room, leading to the case that the humanoid is not in sight, and the robot needs to find the humanoid again (lower following rate).

For the ablation baseline without the humanoid detector, we observe that it has worse following performance (7% drop), and a higher collision rate (13% increase) than those of the one with the humanoid detector. This occurs because while the humanoid GPS offers a relative L2 distance to the humanoid, it does not account for scenarios where a wall or obstacle obstructs the robot’s view. As a result, the robot has no idea if there is a wall with a low L2 distance to the humanoid, leading to a lower following rate and higher collision rate. Providing the humanoid detector allows the agent to ‘see’ if there is a humanoid there, and thus follow the humanoid.

For the ablation baseline without the arm depth, we find that it has the highest collision rate (leading to the lowest following rate). This is because the arm depth provides useful information to know and record where the empty space/obstacles are (incorporated with the LSTM memory) when the robot needs to avoid collision. As a result, we find that the baselines with the arm depth or arm RGB tend to have a lower collision rate.

Finally, for the ablation baseline with the arm depth being replaced by the arm RGB, we find that it has a slightly higher collision rate than the one with the arm depth. This is because the arm depth provides information about the distance to obstacles, leading to a better moving strategy to avoid collisions.

Overall, we find that the end-to-end RL policy and its ablations have a comparable Backup-Yield Rate, suggesting that humanoid avoidance motion is learned through RL training. At the same time, the RL policy and its ablations can maintain a close distance to the humanoid (low Following Distance). But still, there is a performance gap between the RL policy and the Heuristic Expert in terms of SPS and the following rate. This leaves room for future improvement.

C.2 SOCIAL REARRANGEMENT

Here we present additional metrics and analysis for social rearrangement. We also describe the different ablations in more details.

Method	Train-pop-eval				ZSC-pop-eval			
	SR↑	RE↑	CR↓	RC↑	SR↑	RE↑	CR↓	RC↑
Learn-Single	98.50\pm0.48	159.2\pm1.0	0.12 \pm 0.12	0.49 \pm 0.00	50.94 \pm 39.55	106.02 \pm 34.32	0.25 \pm 0.33	0.45 \pm 0.02
Plan-Pop ₁	91.2 \pm 2.63	152.4 \pm 5.4	0.09\pm0.09	0.46 \pm 0.00	50.44 \pm 39.02	109.75 \pm 34.63	0.23 \pm 0.31	0.43 \pm 0.05
Plan-Pop ₂	66.89 \pm 1.47	110.06 \pm 6.83	0.10 \pm 0.10	0.50 \pm 0.00	70.23 \pm 7.02	102.13 \pm 11.10	0.15\pm0.17	0.52 \pm 0.05
Plan-Pop ₃	77.79 \pm 2.86	118.95 \pm 6.04	0.12 \pm 0.12	0.48 \pm 0.01	71.79 \pm 7.38	101.99 \pm 15.18	0.17 \pm 0.19	0.53 \pm 0.05
Plan-Pop ₄	72.42 \pm 1.32	105.49 \pm 1.7	0.09\pm0.09	0.55\pm0.00	71.32 \pm 6.47	103.53 \pm 9.8	0.16 \pm 0.17	0.53 \pm 0.04
Learn-Pop	92.20 \pm 2.21	135.32 \pm 3.43	0.15 \pm 0.15	0.50 \pm 0.00	48.52 \pm 35.51	99.80 \pm 31.02	0.26 \pm 0.33	0.46 \pm 0.02
+ learned skills	41.09 \pm 2.15	79.63 \pm 1.76	0.37 \pm 0.19	0.12 \pm 0.12	21.44 \pm 18.26	76.45 \pm 9.23	0.17 \pm 0.17	0.12 \pm 0.12
- depth + RGB	76.70 \pm 3.15	110.04 \pm 3.05	0.13 \pm 0.14	0.49 \pm 0.02	70.89 \pm 8.18	100.16 \pm 14.79	0.16 \pm 0.18	0.54 \pm 0.04
- Humanoid-GPS	76.45 \pm 1.85	108.96 \pm 2.66	0.18 \pm 0.18	0.49 \pm 0.01	68.70 \pm 6.75	98.58 \pm 10.32	0.22 \pm 0.24	0.53 \pm 0.05
- Primitive actions	85.71 \pm 1.58	124.36 \pm 3.79	0.32 \pm 0.32	0.55\pm0.00	76.80\pm9.66	111.97\pm10.91	0.33 \pm 0.34	0.58\pm0.04

Table 3: Social Rearrangement baseline results.

C.2.1 ADDITIONAL METRICS

Collision Rate (CR). Together with the success rate and relative efficiency, we are interested in measuring whether the social rearrangement agents can complete tasks safely, without colliding with the humanoid agent. Thus, we measure the collision rate (CR) in both the train-population and the zsc-population settings. Following Sec. 3.2, we define CR as the proportion of episodes containing collisions between the robot and the humanoid. We report the results in Table 3, along with the Success rate (SR) and Relative Efficiency (RE) metrics. We observe similar trends to the success rate measure, with Learn-Single, Plan-Pop₁, having low CR with the training population but high CR with ZSC population ($0.09 \rightarrow 0.23$ with train-pop for Plan-pop₁). Plan-Pop₄ obtains the best collision rate in the training population, with a rate of 9%, despite having a more diverse population than other baselines. This is because one of the agents in the training population for Plan-pop₄ stays in place does no part of the task, thus reducing the total number of collisions. In the ZSC setting, Plan-Pop₂ obtains the lowest collision rate, though the rates are comparable across Plan-Pop_{2,4}.

Ratio of Completion (RC). We also report, for all our baselines, the ratio of the task completed by the robot, which measures the proportion of objects that were rearranged by the robot agent. A value of 1.0 indicates that the task is completely done by the robot, whereas 0.0 indicates that the task was done by the humanoid alone. Values closer to 0.5 indicate that the task was split among the robot and humanoid, resulting in increased efficiency. We show the results in Table 3. Almost all baselines achieve a RC close to 0.5 with training population. Learn-Single achieves a RC close to 0.5 in the train population, showing that both agents learned to split the task to perform it efficiently. Plan-Pop₁ shows a slight decrease in RC, which is consistent with the drop in the SR, indicating that agents are still splitting the task evenly, while overall achieving lower success rate. Plan-pop₄ has the highest train-pop RC, because one of its training population agents complete no part of the task, requiring the robot to rearrange both objects. The results on ZSC population follow success rates, with Plan-Pop_{2,3,4} achieving higher RC, since the agents are trained to rearrange either object. In comparison, Learn-Single, Plan-pop₁ and Learn-Pop have slightly reduced RC due to inability to generalize to partners that rearrange different objects than their training population.

C.2.2 ADDITIONAL ABLATION RESULTS AND ANALYSIS

In the main paper we presented ablation experiments where we: (1) replaced oracle skills with learned skills, (2) replaced depth arm camera with RGB, (3) Removed the humanoid GPS. Here, we present an additional ablation, where we remove some primitive actions like move backwards, forwards, turn left or right from the action space of the high-level policy. This measures the effect of these primitive navigation actions in the Social Rearrangement task, called - *Primitive actions*. The robot agent is trained in the same setting as Plan-Pop₃, but without the four low level navigation actions. We report the results in Table 3. As we can see, the collision rate (CR) increases significantly, showing that the primitive navigation actions are essential to reduce collisions between both agents. At the same time, removing the navigation actions improves the success rate and RE both in the Train-Pop and ZSC-pop settings. This is because the agent does not spend time making way for the humanoid, which allows to complete the task in a lower number of steps. Despite being faster at completion, the high CR makes this baseline non-ideal as it reduces the “safety” of human or humanoid collaborators.

C.2.3 ZERO-SHOT POPULATION DETAILS

Here we describe the ZSC population used in our experiments in detail, and their effect in the behavior of the trained agents in the zero-shot coordination setting.

The 10 ZSC population collaborators used in ZSC eval are created as follows: 3 are trained checkpoints from the training of Learn-Single, 3 are trained checkpoints from the training run of Learn-Pop and 4 are planner-based humanoids, where 1 picks up both objects, 2 pick up one of the two, and 1 stays still. For the learned checkpoints, we only use the learned policy for the humanoid, and discard the learned robot policy. As a result, most baselines have seen about 1/3 of the ZSC population during training, and need to generalize to 2/3 of the population. In general, the learned ZSC evaluation agents tend to focus on rearranging one of the 2 objects in the environment, while the planner-based agents follow their scripted behavior.

We measure the trained agents’ performance when evaluated with different agents from the ZSC-pop-eval population. In Figs. 8a, 8b, 8c and 8d, we show the success rate (a), efficiency rate (b), collision

Training agents	Learn-Single	32.59 \pm 46.07	65.92 \pm 46.61	65.16 \pm 46.08	26.81 \pm 31.90	62.52 \pm 44.17	31.59 \pm 44.51	64.06 \pm 45.30	47.87 \pm 0.42	64.33 \pm 0.54	48.58 \pm 0.55	50.94 \pm 39.55
	Plan-Pop ₁	0.08 \pm 0.04	96.59 \pm 1.46	95.86 \pm 2.52	4.19 \pm 0.26	92.79 \pm 1.81	0.09 \pm 0.03	94.63 \pm 2.51	46.96 \pm 1.04	64.49 \pm 0.81	48.68 \pm 0.55	54.44 \pm 39.02
	Plan-Pop ₂	79.27 \pm 5.66	72.62 \pm 5.94	70.41 \pm 5.31	59.57 \pm 4.51	69.04 \pm 5.15	72.97 \pm 5.46	65.79 \pm 4.56	67.26 \pm 1.63	77.06 \pm 0.83	68.40 \pm 2.88	70.23 \pm 7.02
	Plan-Pop ₃	73.21 \pm 1.93	78.07 \pm 6.33	76.36 \pm 7.18	59.63 \pm 3.59	72.19 \pm 8.60	69.96 \pm 3.09	71.60 \pm 7.75	69.81 \pm 3.56	78.22 \pm 2.72	68.90 \pm 2.49	71.79 \pm 7.38
	Plan-Pop ₄	76.48 \pm 6.24	72.39 \pm 7.67	71.50 \pm 6.55	63.60 \pm 1.86	68.14 \pm 5.04	73.03 \pm 4.55	66.63 \pm 6.05	68.59 \pm 1.80	77.72 \pm 0.83	75.14 \pm 1.71	71.32 \pm 6.47
	Learn-Pop	67.37 \pm 41.80	38.26 \pm 42.58	37.05 \pm 41.84	49.79 \pm 27.16	34.38 \pm 40.10	63.59 \pm 39.93	34.41 \pm 43.34	47.30 \pm 1.77	64.48 \pm 1.19	48.61 \pm 0.66	48.52 \pm 35.51
		ZSC-Agents										
		Learner-Partner ₁ Learner-Partner ₂ Learner-Partner ₃ Learner-Partner ₄ Learner-Partner ₅ Learner-Partner ₆ Plan-Partner ₁ Plan-Partner ₂ Plan-Partner ₃ Plan-Partner ₄ Averaged										

(a) Success rate across different ZSC-agents

Training agents	Learn-Single	97.45 \pm 41.89	127.85 \pm 42.44	125.49 \pm 40.77	77.64 \pm 14.52	105.51 \pm 26.66	91.14 \pm 32.97	129.58 \pm 43.66	100.78 \pm 0.21	108.92 \pm 0.65	95.81 \pm 0.88	106.02 \pm 34.33
	Plan-Pop ₁	67.82 \pm 0.02	153.88 \pm 3.66	150.57 \pm 5.67	67.46 \pm 0.07	123.58 \pm 1.75	67.82 \pm 0.02	160.24 \pm 3.54	100.20 \pm 0.95	109.48 \pm 0.90	96.43 \pm 0.61	109.75 \pm 34.63
	Plan-Pop ₂	98.11 \pm 5.31	114.15 \pm 3.39	110.16 \pm 2.95	79.37 \pm 1.58	98.36 \pm 1.17	90.03 \pm 4.00	113.26 \pm 3.25	105.47 \pm 1.78	112.28 \pm 1.41	100.07 \pm 1.65	102.13 \pm 11.10
	Plan-Pop ₃	97.24 \pm 12.42	113.12 \pm 16.07	110.93 \pm 16.49	81.00 \pm 6.22	97.56 \pm 11.95	91.46 \pm 10.41	113.57 \pm 15.70	105.17 \pm 3.49	111.39 \pm 2.10	98.49 \pm 1.38	101.99 \pm 15.18
	Plan-Pop ₄	105.28 \pm 7.56	109.88 \pm 6.09	107.67 \pm 6.76	84.90 \pm 3.94	95.31 \pm 6.77	96.67 \pm 7.51	109.59 \pm 5.50	107.43 \pm 1.17	113.29 \pm 1.58	105.34 \pm 2.15	103.54 \pm 9.84
	Learn-Pop	124.02 \pm 39.60	97.57 \pm 41.77	96.08 \pm 39.82	85.14 \pm 12.23	84.79 \pm 24.35	110.03 \pm 29.80	98.70 \pm 42.75	97.75 \pm 2.00	108.59 \pm 1.99	95.34 \pm 0.77	99.80 \pm 31.01
		ZSC-Agents										
		Learner-Partner ₁ Learner-Partner ₂ Learner-Partner ₃ Learner-Partner ₄ Learner-Partner ₅ Learner-Partner ₆ Plan-Partner ₁ Plan-Partner ₂ Plan-Partner ₃ Plan-Partner ₄ Averaged										

(b) Efficiency rate ZSC-agents

Training agents	Learn-Single	0.73 \pm 0.35	0.48 \pm 0.33	0.49 \pm 0.33	0.74 \pm 0.20	0.53 \pm 0.29	0.74 \pm 0.33	0.29 \pm 0.11	0.34 \pm 0.01	0.38 \pm 0.02	0.29 \pm 0.01	0.25 \pm 0.33
	Plan-Pop ₁	0.94 \pm 0.02	0.23 \pm 0.01	0.24 \pm 0.00	0.86 \pm 0.01	0.30 \pm 0.00	0.94 \pm 0.02	0.19 \pm 0.00	0.31 \pm 0.01	0.35 \pm 0.02	0.26 \pm 0.01	0.23 \pm 0.31
	Plan-Pop ₂	0.36 \pm 0.02	0.32 \pm 0.04	0.33 \pm 0.04	0.46 \pm 0.02	0.36 \pm 0.04	0.41 \pm 0.03	0.17 \pm 0.00	0.21 \pm 0.00	0.24 \pm 0.01	0.18 \pm 0.01	0.15 \pm 0.17
	Plan-Pop ₃	0.40 \pm 0.07	0.35 \pm 0.08	0.34 \pm 0.06	0.52 \pm 0.07	0.39 \pm 0.06	0.44 \pm 0.08	0.20 \pm 0.05	0.22 \pm 0.02	0.25 \pm 0.02	0.19 \pm 0.01	0.17 \pm 0.19
	Plan-Pop ₄	0.33 \pm 0.03	0.34 \pm 0.04	0.35 \pm 0.03	0.47 \pm 0.05	0.40 \pm 0.04	0.38 \pm 0.04	0.20 \pm 0.04	0.21 \pm 0.03	0.23 \pm 0.02	0.19 \pm 0.02	0.16 \pm 0.17
	Learn-Pop	0.50 \pm 0.34	0.71 \pm 0.31	0.74 \pm 0.31	0.59 \pm 0.22	0.74 \pm 0.29	0.52 \pm 0.32	0.40 \pm 0.13	0.32 \pm 0.03	0.39 \pm 0.01	0.30 \pm 0.01	0.26 \pm 0.33
		ZSC-Agents										
		Learner-Partner ₁ Learner-Partner ₂ Learner-Partner ₃ Learner-Partner ₄ Learner-Partner ₅ Learner-Partner ₆ Plan-Partner ₁ Plan-Partner ₂ Plan-Partner ₃ Plan-Partner ₄ Averaged										

(c) Collision rate ZSC-agents

Training agents	Learn-Single	0.45 \pm 0.02	0.47 \pm 0.02	0.47 \pm 0.03	0.45 \pm 0.02	0.46 \pm 0.04	0.44 \pm 0.03	0.47 \pm 0.01	0.47 \pm 0.00	0.44 \pm 0.00	0.44 \pm 0.00	0.46 \pm 0.03
	Plan-Pop ₁	0.38 \pm 0.06	0.48 \pm 0.00	0.48 \pm 0.01	0.41 \pm 0.04	0.48 \pm 0.00	0.37 \pm 0.06	0.47 \pm 0.01	0.44 \pm 0.03	0.41 \pm 0.03	0.42 \pm 0.02	0.44 \pm 0.05
	Plan-Pop ₂	0.59 \pm 0.02	0.48 \pm 0.02	0.48 \pm 0.02	0.58 \pm 0.01	0.48 \pm 0.02	0.57 \pm 0.02	0.46 \pm 0.01	0.51 \pm 0.01	0.48 \pm 0.01	0.54 \pm 0.01	0.52 \pm 0.05
	Plan-Pop ₃	0.55 \pm 0.05	0.52 \pm 0.04	0.52 \pm 0.03	0.58 \pm 0.02	0.51 \pm 0.02	0.55 \pm 0.04	0.50 \pm 0.04	0.52 \pm 0.01	0.49 \pm 0.00	0.53 \pm 0.01	0.53 \pm 0.04
	Plan-Pop ₄	0.55 \pm 0.06	0.52 \pm 0.06	0.52 \pm 0.06	0.58 \pm 0.03	0.51 \pm 0.05	0.55 \pm 0.05	0.50 \pm 0.07	0.53 \pm 0.01	0.49 \pm 0.00	0.57 \pm 0.01	0.53 \pm 0.05
	Learn-Pop	0.48 \pm 0.01	0.47 \pm 0.02	0.46 \pm 0.02	0.50 \pm 0.01	0.44 \pm 0.04	0.48 \pm 0.02	0.46 \pm 0.02	0.45 \pm 0.02	0.44 \pm 0.01	0.45 \pm 0.01	0.46 \pm 0.03
		ZSC-Agents										
		Learner-Partner ₁ Learner-Partner ₂ Learner-Partner ₃ Learner-Partner ₄ Learner-Partner ₅ Learner-Partner ₆ Plan-Partner ₁ Plan-Partner ₂ Plan-Partner ₃ Plan-Partner ₄ Averaged										

(d) Ratio of Completion for ZSC-agents

Figure 8: **Zero-shot coordination.** We report the performance of the baseline agents in the zero-shot coordination setting. Each row corresponds to one of the baselines and the columns represent the different types of zero-shot coordination agents.



Figure 9: **Robot Embodiment.** Spot robot in the simulation environment is designed to minimize the embodiment gaps to the robot in the physical world.

rate (c) and ratio of completion (d) of the different baseline trained agents under this setting. Each row corresponds to one of the baselines, trained and averaged across three seeds, and each column corresponds to one of the 10 ZSC evaluation agents. The last column corresponds to the ZSC-pop-val results in Fig. 4.

The Learn-Single agent overfits to one type of partner, learning to split the task by focusing on rearranging a single object and letting the other agent rearrange the remaining one. When evaluated with a ZSC partner, it will either be highly successful and efficient, if the evaluation partner was trained to pick the opposite object, or exhibit low performance if the evaluation agent is focused on the same object. For this reason the success rate in the first row of Fig. 8a is close to 33% and 66% for the Learn-Single agent, corresponding to one or two of the 3 training seeds matching the ZSC-partner.

Plan-Pop₁ exhibits a similar behavior to the Learn-Single baseline, but in this case, the agent is trained with a planner that always focuses on the same object, which makes Plan-Pop₁ focus on the opposite objects across all the training seeds. As a result, the agent has a success rate close to 100% or 0% for the different ZSC evaluation agents. As expected, it also exhibits a high success rate when partnering with the Plan₁ agent. Because of this, the agent also exhibits a significant increase in relative efficiency when partnering with certain agents. As a result it is the method with highest relative efficiency, when averaged across ZSC-agents, despite exhibiting high variance.

Plan-Pop₂ is trained with a population of agents arranging one of the two objects at random and therefore it cannot specialize anymore by always focusing on the same object. As a result, the agent shows much lower variance across the members of the ZSC-population, resulting in a higher success rate. At the same time, it needs to adapt to the behavior of the ZSC partnering agent, which results in lower peak efficiency than Plan-Pop₁ or Learn-Single, resulting on a lower relative efficiency on average. The average collision rate is also reduced relative to the previous baselines, with a lower collision rate with planning-based partners than the learning-based ones. Interestingly, this trend of lower CR with ZSC plan-agents holds across all baselines. We believe this is because ZSC plan-agents stop after finishing their portion of the task, while ZSC learned-agents continue to move in the environment (since our reward function does not penalize this). As a result, the chances of colliding with ZSC plan-partners is higher than with ZSC learned partners.

Plan-Pop_{3,4} exhibit very similar performance across the different agents in the ZSC population. Their success rate shows a slight improvement with respect to Plan-Pop₂, whereas the relative efficiency decreases slightly for Plan-Pop₃ and increases slightly for Plan-Pop₄, though not significant. In general, adding an extra agents to Plan-Pop₄ that remains still does not seem to change the performance. To improve performance of learned coordination robot policies, we might need to incorporate other types of diversities, like humanoid speed, instead of just which object they rearrange.

C.3 QUALITATIVE EXAMPLE

We also show in Fig. 10 an example episode of social rearrangement. We use Plan-Pop₃ as the baseline policy with learned low-level skills. In this example, the task is split amongst both agents, with each rearranging one of the goal objects. Frames 1,3 show the robot and humanoid picking

up objects. 4,5 show them placing each object in its desired location. Frame 2 shows a scenario where the humanoid and robot cross paths, and the robot backs up to let the humanoid pass before continuing, avoiding a collision.

D ROBOT

We use the Boston Dynamics (BD) [Spot](#) robot as the robot agent (Fig. 9) due to its robust hardware for real world deployment of trained policies in the future. Spot is a quadruped robot with five pairs of depth and RGB cameras (front-left, front-right, left, right, back), and a 7 degree of freedom arm with one pair of depth and RGB cameras mounted on the gripper. The arm depth camera has a 224×171 resolution and hfov of 55. Spot robots are capable of grasping rigid objects, climbing up/down stairs, and outdoor/indoor navigation given users’ control input using BD’s Spot control APIs. Its versatility and robustness make it an ideal mobile platform for studying sensing, manipulation, and human-robot interaction. On the simulation side, the Spot robot simulation moves its base by commanding linear and angular velocities (2D, continuous) and can move backwards if turning or moving forwards results in collision. The action space of the robot is continuous forward and angular velocity in the local frame of the robot (2-dim) for navigation and delta joint angles (7-dim) for the arm.

E SCENES

We incorporate scene assets from the Habitat Synthetic Scenes Dataset (HSSD-200) ([Khanna et al., 2023](#)) in Habitat 3.0. HSSD is a dataset of 211 high-quality 3D homes (scenes) containing over 18k individual models of real-world objects. An example scene is shown in Fig. 11. For our experiments, we use a subset of 59 scenes and limit our objects to the YCB dataset ([Calli et al., 2017](#)) for simplicity. Specifically, we use 37 scenes from training, sampling 1000 episodes per scene, 12 for validation, with 100 episodes per scene and 10 for test, with 15 episodes in each scene. Note that the Habitat3.0 framework will be released with the complete HSSD dataset.

F SIMULATOR DETAILS

F.1 BENCHMARKING

We benchmark here Habitat 3.0 speed under varying scene sizes, different number of objects in the scene, and the type and number of agents. All tests are conducted on a single Nvidia V100 GPU. We report results on a single environment (Fig. 12, left) and 16 parallel Habitat 3.0 environments (Fig 12, right). For each measure we sample random actions for 300 steps per agent and report average and standard error results across 10 runs. All the solid blue curves correspond to small scenes sizes and two objects in the environment. For each agent, we always render a single depth image. Our small scene is $68.56m^2$ in size and has 1 bedroom and 1 bathroom, the medium scene is $136.11m^2$ in size with 3 bedrooms and 2 bathrooms, and the large scene is $846.15m^2$ in size with 4 bedrooms, 4 bathrooms, and a den and office space.

On a single environment, we obtain performances on the range of 140 to 250 FPS, depending on the setting. As we can see, varying the number of objects has no significant effect in the performance speed, while we notice significant differences when changing the scene size (245 ± 19 FPS down to 154 ± 14). Switching from a single spot agent to two spot agents drops the performance from 245 ± 19 to 150 ± 1 , having a similar effect to varying the scene size. The humanoid is also slower than the Spot robot (245 ± 19 vs 155 ± 26), due to the higher number of joints in the skeletal model. However, the difference between robot and humanoid agents, becomes much less pronounced when switching to the two agent setting, obtaining comparable performances between the Robot-Robot and Human-Robot settings (150 ± 1 vs. 136 ± 1). Since humanoid-robot simulation is the primary use-case of Habitat3.0, this is a positive signal, that shows that adding a humanoid over a robot does not decrease simulation speed. We don’t notice a difference in performance between representing the humanoid as a skeleton or applying linear blend skinning, implying that the visual realism of skinned humanoids has very little effect on our simulation speed. We also don’t notice significant differences in performance between the picking or navigation actions.

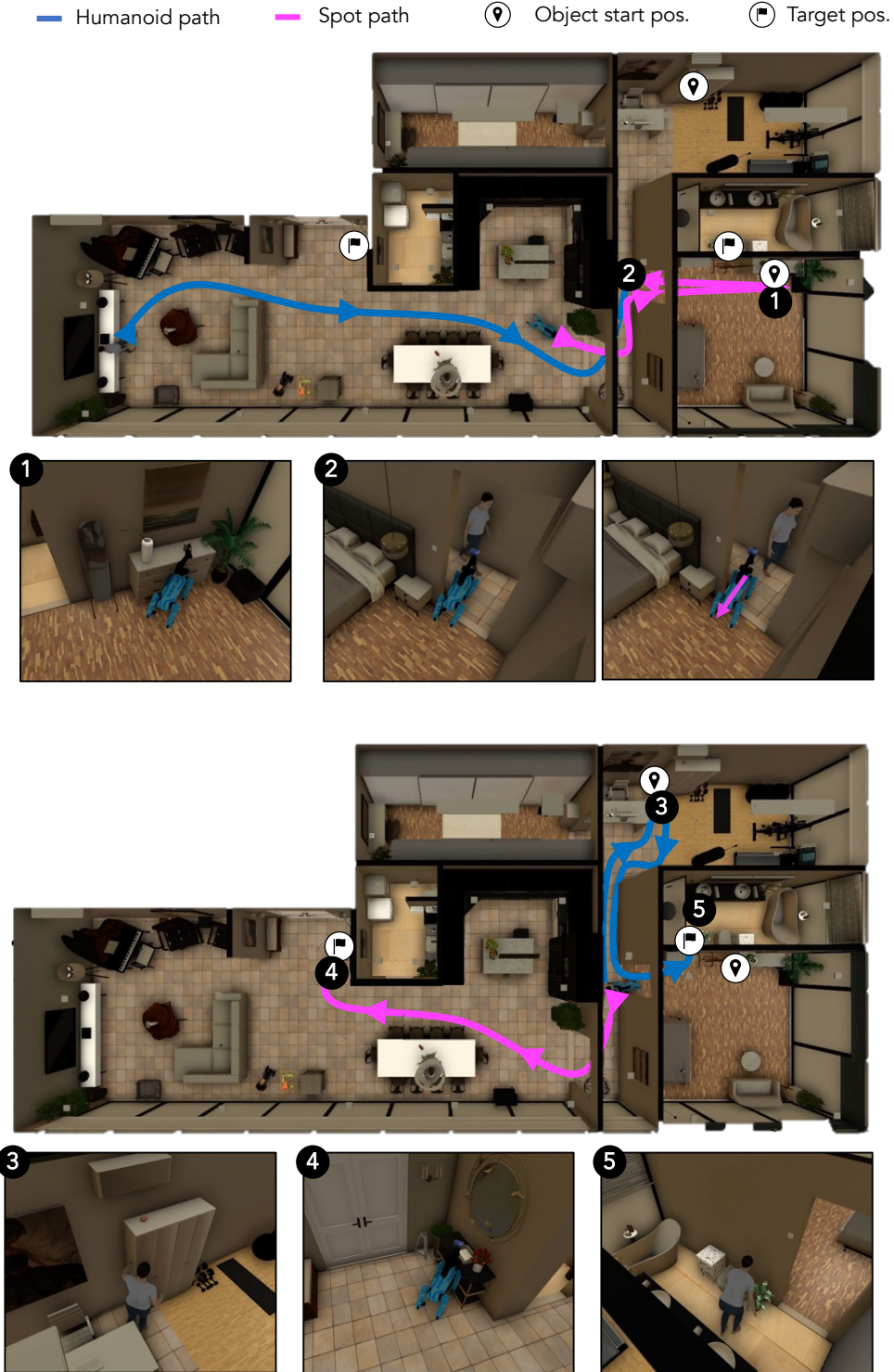


Figure 10: **Social Rearrangement Example:** We show an example behavior from the Plan-Pop₃ baseline with learned low-level skills. Here, the agents split the rearrangement task, with each picking one of the objects (frames 1, 3) and placing them in their target location (frames 4, 5). In frame 2, we observe an emergent collaborative behavior: the robot moves backward to let the humanoid pass through the hallway, before continuing with its task, avoiding a collision.



Figure 11: **Example Scene.** Habitat 3.0 is based on HSSD (Khanna et al., 2023) scenes. An example scene which includes various types of objects and scene clutter has been shown.

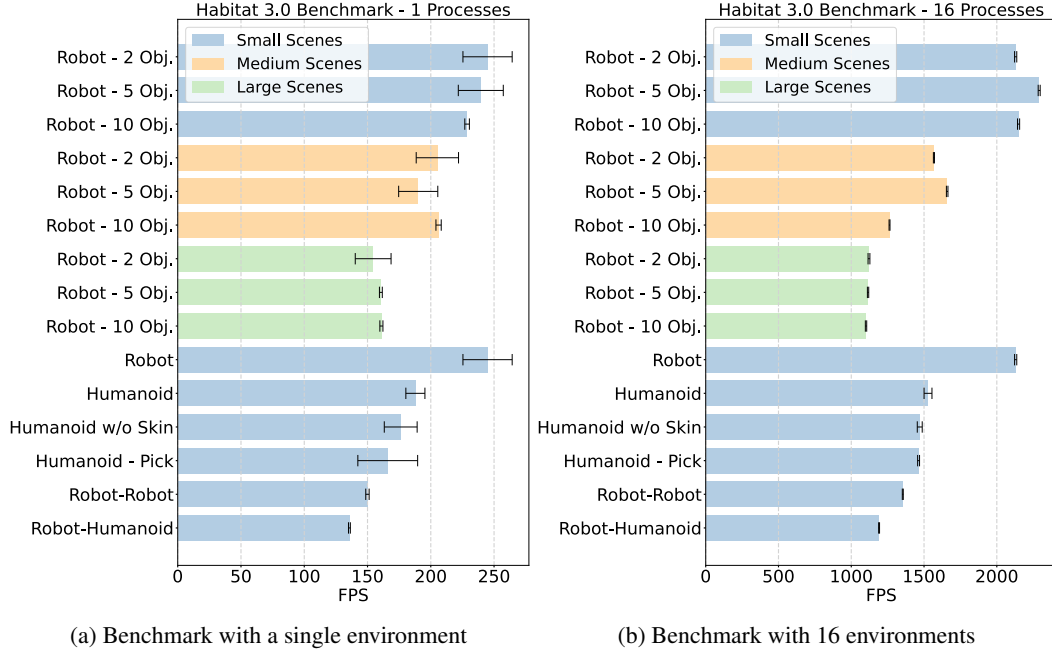


Figure 12: **Benchmark results in Habitat 3.0.** We study the effect of varying scene size, number of objects, type of agents and single or multi-agent.

We observe similar trends when switching to 16 environments, obtaining promising scaling results, with performances in the range of 1100 to 2290 FPS. Note that these 16 environments are still on a single GPU, and Habitat 3.0 natively supports environment parallelization. As a result, the most common simulation speed that users will experience when training policies is between 1100 to 2290 FPS depending on the scenario.

F.2 COMPARISON WITH EXISTING SIMULATORS

Habitat3.0 is designed to support efficient simulation of tasks with humans and robots in indoor environments. In Tab. 4 we provide a comparison of Habitat3.0 with some existing related simulators. Habitat3.0 provides support for both real robot models and humanoids, enabling to train policies for both types of agents. Humanoids in Habitat3.0 are based on the SMPL-X model, as opposed to Authored Designs (AD). This enables to scale up the number of possible body models, and provide motions coming from motion captured data or motion generation models. The humanoids can also be controlled at different levels, including joints (Jt), inverse kinematics (IK) or high level commands, such as walk or pick up (HL). A key feature in our platform is the HITL interface, which allows to control the humanoid via a mouse-keyboard interface (KB) or VR (VR). Furthermore, we include a large number of authored multi-room scenes enabling training policies in a wide diversity of realistic indoor environments, and provides a significant increase in simulation speed, compared to other simulators with robot and humanoid support. Note that the speed is influenced by several factors, including rendering fidelity, physics simulation, resolution, hardware capabilities, and more (refer to

Section F.1 for detailed benchmarking). As such, speed numbers between different simulators are not directly comparable and serve as rough reference points. We take the reported numbers in individual papers for single environment settings.

	Robot	Humanoid		HITL	Type of	#Authored	Speed
	Support	Support	Body Model	Control	Simulation	Multi-Agent	(steps/s)
VirtualHome-Social	-	✓	AD	IK, HL	☞	PP	50
VRKitchen	-	✓	AD	Jt, IK	☞	RBF, PP	16
SAPIEN	✓	-	-	-	-	RBF	-
AI2-THOR	✓	-	-	-	-	RBF, PP	120
Habitat 2.0	✓	-	-	-	-	RBF	105
TDW	✓	✓	AD	IK, HL	☞	RBF, PS	15
SEAN 2.0	✓	✓	AD	HL	☞☞	RBF	3
iGibson	✓	✓	AD	R	☞☞	RBF	50
Habitat 3.0	✓	✓	SMPL-X	Jt, IK, HL	☞☞	RBF	200

Table 4: **Comparison with related simulators.** We compare Habitat3.0 with other related simulators for Embodied AI supporting simulation of real robots or humanoids. Speeds were taken directly from respective publications or obtained via direct personal correspondence with the authors when not publicly available (indicated by †).

Body Model: Authored Design (AD), SMPL-X (Pavlakos et al., 2019).

Control: Joints (Jt), Inverse Kinematics (IK), High Level Commands (HL), Rigid Transforms (R).

HITL Interface: Mouse and Keyboard (☞), Virtual Reality (☞).

Type of Simulation: Rigid Body Physics (RBF), Pre-post-conditions (PP), Particle Simulation (PS).

Speed: The reported numbers correspond to benchmarks conducted by different teams using different hardware configurations to simulate diverse capabilities. Thus, these should be considered only as qualitative comparisons representing what a user should expect to experience on a single instance of the simulator (without parallelization).

G HUMAN-IN-THE-LOOP (HITL) EVALUATION

We test the ability of trained robotic agents to coordinate with real humans via our human-in-the-loop (HITL) infrastructure across 30 participants. Our user study consisted of 3 conditions: doing the task alone (*solo*), paired with a robot operating with *Learn-Single*, or paired with a *Plan-Pop₃* agent. We have a brief training step, where the users get accustomed to the tool, followed by solving each condition in a random order. However, as the users interact more with the tool, they get better at the overall task. We account for this learning effect by leveraging latin-square counter-balancing (Bradley, 1958) for the ordering of these conditions.

Each participant performs the task for 10 episodes per condition in one of the test scenes. We measure the collision rate (CR), task completion steps (TS) and Relative Efficiency (RE) across all episodes, and report them in Table 1. CR is the ratio of episodes that contain collisions with the robot. SR, RE are the same as defined in Sec 4.2. RE still captures the relative efficiency of task completion when the task is done with a robot and requires the computation of task completion steps with the robot relative to the human doing the task alone. However, unlike the automated evaluation (Train-pop-eval) and (ZSC-pop-eval), we cannot compute the RE per episode in HITL evaluation. Since humans exhibit learning effect when asked to do an episode multiple times, we cannot ask the human to do the same episode with and without the robot, preventing the computation of RE per episode. Instead, we fit a generalized linear mixed-effect model (GLMM) (Kaptein, 2016) with a Poisson distribution, for TS as the dependent variable and method/condition as the independent variable while controlling for variations from participants and scenes using random intercepts (Bates et al., 2015). We then use the ratio between the estimated means of TS for Learn-Single and Plan-Pop conditions w.r.t the solo condition to compute their average RE respectively. For CR, we fit a logistic regression model with a binomial distribution, again controlling for the random effects of participants and scenes. We use lme4 package (Bates et al., 2015) in R version of 4.3.1 to fit the GLMMs. Table 1 shows the results over three conditions across the 30 users. Fig. 13 shows the TS over all successful episodes and CR over all episodes across the three conditions in HITL.

After fitting a GLMM model with Poisson distribution for TS, we also compute post hoc pairwise comparisons between the three conditions. We find that the difference in estimated mean of TS

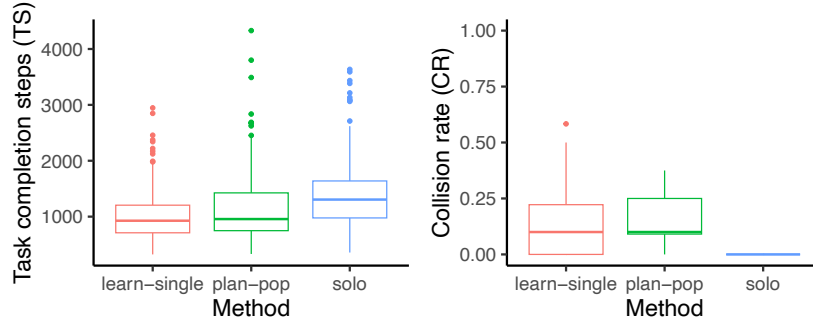


Figure 13: TS and CR across all participants in the user study.

Method	Estimated mean difference in TS	p-value
Plan-pop - Learn-Single	78.45	0.0533
Solo - Learn-Single	316.58	<.0001
Solo - Plan-pop	238.12	<.0001

Table 5: Post hoc pairwise comparison between the three conditions of the user study for TS.

between Learn-Single and Plan-pop is not significant, while that between both these conditions and the solo condition is significant (Table 5). To the best of our knowledge, this is the first analysis of performance of learned collaboration agents when paired with real human partners in a realistic, everyday rearrangement task. The indication that the robot agents indeed make humans more efficient is very promising, with huge significance for the assistive-robotics community.

H LIMITATIONS

In this section, we address the limitations of our work, focusing on our three primary contributions: **Human simulation.** While our framework can accommodate various actions, our experiments utilize only walking and reaching behaviors. We implement the reaching behavior by obtaining static poses representing distinct reaching positions and interpolating between them to generate motion. For the walking motion, our approach for navigating across waypoints is suitable for path planners, but shows visual artifacts when the humanoid is rotating in place. While this approach suits the behaviors we consider, it may not be suitable for more complex motions, such as opening a cabinet or sitting down. This aspect will be explored in our future work. Additionally, we employ fixed linear blend skinning to represent the human, with static weight assignments for various poses. Consequently, certain pose configurations may exhibit skinning artifacts, as mentioned in the main paper.

Human-in-the-loop tool. There is a gap between the observations a human acquires through the HITL tool and those accessible to our humanoid in automated evaluations. Although we provide visual cues to bridge this gap (such as markers that indicate the direction from the human to the target objects), they still represent a different observation space between the two domains.

Tasks. Our current focus is on zero-shot coordination without communication, but introducing communication between the robot and human could potentially enhance task efficiency. Furthermore, our models currently benefit from access to ground truth information regarding the initial and final object locations in the rearrangement task. A future challenge lies in integrating search capabilities into the rearrangement task. Additionally, we make the assumption that objects are initially placed on open receptacles. Handling rearrangements that involve searching for and interacting with articulated objects, like drawers, poses a greater challenge.