

Supplementary Material for: AgentWorld: An Interactive Simulation Platform for Scene Construction and Mobile Robotic Manipulation

Anonymous Author(s)

Affiliation

Address

email

1 A Supplementary Video

2 The supplementary video provides dynamic visualizations of AgentWorld’s core capabilities, orga-
3 nized as follows:

4 • Scene Construction Demonstration (0:00-0:45):

- 5 – 0:00-0:13: Layout generation showcasing different room organization with stairs.
- 6 – 0:13-0:20: Semantic asset selection and placement demonstration.
- 7 – 0:20-0:29: Visual material configuration with walls, floors and assets with various mate-
8 rials.
- 9 – 0:29-0:45: Interactive physics simulation demonstrating inter-actable assets with physics
10 properties like collisions and joints, etc.

11 • Teleoperation System (0:45-2:04):

- 12 – 0:45-1:05: An example of VR-based arm and gripper teleoperation.
- 13 – 1:05-2:04: An example of dexterous hand control with manipulation of objects and artic-
14 ulated assets.

15 • Learning Results (2:04-2:56):

- 16 – 2:04-2:42: Imitation learning performance in simulation for multistage tasks.
- 17 – 2:42-2:56: A Sim-to-real transfer demonstration of simple pick & place task.

18 The video complements our paper by providing real-time demonstrations of AgentWorld’s interac-
19 tive features and experimental outcomes. Each segment highlights key functionalities of our simu-
20 lation platform.

21 B Training Details

22 B.1 Locomotion Policy for Humanoid Robots

23 We define a series of rewards to encourage humanoid robots to achieve walking, as shown in Table
24 B.1. Meanwhile, we define relevant termination conditions to end a certain current episode, as shown
25 in Table B.1. In addition, we use the foot trajectory generator[1] to generate a reference trajectory
26 for robot feet. The output of the locomotion policy network will be added as an increment to this
27 reference trajectory. In this way, we can narrow the exploration range of the action and accelerate the
28 training process. During the training process, we randomly set different joystick control commands
29 within a certain period, enabling the robot to encounter as many different control commands as

Reward	Description
track_angle_vel_z_exp	Encourages the robot to follow the rotation commands in the yaw direction.
track_linear_vel_xy_exp	Encourages the robot to follow the translation commands in the x and y directions.
linear_vel_z_l2	Penalizes the robot for fluctuations in the z - direction.
feet_air_time	Encourages the robot to lift its feet rather than drag them on the ground.
feet_slide	Penalizes the robot for foot sliding.
dof_acc_l2	Penalizes the robot’s joint accelerations to ensure smooth joint movements.
dof_torques_l2	Penalizes the robot’s joint torques to keep the joint torques as stable as possible.
dof_pos_limits	Penalizes the robot when its joints exceed the defined limits.
joint_deviation_arms	Penalizes the deviation of the robot’s arm joints from their default positions, guiding the policy to explore around the default positions.
joint_deviation_hip	Penalizes the deviation of the robot’s hip joints from their default positions, guiding the policy to explore around the default positions.
joint_deviation_torso	Penalizes the deviation of the robot’s torso joints from their default positions, guiding the policy to explore around the default positions.

Table 1: Rewards

Condition	Description
base_contact	If the robot’s body collides with the ground, it indicates that the robot has fallen.
time_out	Limits the maximum time for each training episode.

Table 2: Termination Conditions

possible. The goal of the training is to make the robot follow these commands to train the robot’s locomotion policy.

In practical applications, the robot’s upper limbs will grasp different objects, which means that the positions of the upper limbs can vary widely. Therefore, to ensure that our locomotion policy is robust enough, we have incorporated randomization of the arm positions during the training process. Based on the given default arm position, for every 150 frames, we superimpose the result of a uniform sampling as the arm target joint position. This approach enables the training to cover a wide range of different positions of the upper limbs.

B.2 Implementation of Imitation Learning Algorithms

Observation & Action Space. The input RGB observations maintain a consistent resolution of 480×640 . The robot’s proprioceptive state is defined as $s = [s_{qpos}, s_e, s_f]$, where s_{qpos} represents joint positions of the arms, s_e denotes end effector poses, and $s_f = (x, y, \theta)$ is the floating base’s plane location and yaw angle. As previously described, the algorithm outputs the action $a = [a_{qpos}, a_e, a_f, 0/1]$, where a_{qpos} and a_e specify arm joint and end effector actions, $a_f = (v_x, v_y, v_\theta)$ represents mobile base velocity commands, and 0/1 the binary value selects between locomotion and manipulation modes. The Unitree G1, H1, and Franka Emika Panda arm each possess 7 degrees of freedom (DOFs), while the dual-arm X-Trainer has 6 DOFs per arm. For end effectors, grippers are controlled via a single normalized value for opening distance, and the TRX-Hand5 features 18 actuated DOFs.

Algorithm Architecture. Our behavior cloning algorithm utilizes a ResNet-18 [2] image encoder for visual feature extraction. The flattened image features are concatenated with proprioception states and processed through a 5-layer MLP for action prediction. Layer normalization and tanh activation are applied to the output. For ACT, Diffusion Policy, and π_0 , we adopt the LeRobot [3] implementation with action chunk sizes of 20 for basic tasks and 40 for multistage tasks. All approaches incorporate *temporal ensemble* to enhance action smoothness.

Implementation Details. All algorithms are trained with batch size 16, except π_0 which uses 8. Training spans 50K steps for basic tasks and 150K for multistage tasks. We observe convergence across all methods with 80K-100K steps, with no subsequent performance increases. On NVIDIA L20 GPU, training durations are 6 hours for behavior cloning, 12 hours for ACT and Diffusion Policy, and 18 hours for π_0 .

C Dataset Details

Table C demonstrates detailed information of our datasets including concrete descriptions and number of trajectories of each task in every category.

Category	Task Description	Num of Trajectories
Pick & Place	Pick some objects and place them in the bowl.	10×10
	Pick some objects and place them into the opened microwave / fridge / stove.	30×10
	Pick some objects out from the bowl.	10×10
	Pick some objects out from the opened microwave / fridge / stove.	30×10
Open & Close	Open & Close doors of rooms.	10×20
	Open & Close doors of the furniture: fridges, microwaves, stoves.	30×20
Push & Pull	Pull out & Push in the drawers.	10×20
	Push the buttons on microwaves / stoves.	10×20
	Push some objects on the tables away from the robot.	10×10
Living Room	Pick up the books on the table and walk to the shelf and organize them.	10×50
	Pick up the pitcher and the cup and pour the drink.	5×30
	Pick up the trash on the table and walk to the trash bin to throw them away	10×50
Bedroom	Slide the objects on the bed away and put the pillow on the top of the bed	10×50
	Pick up the clothes rack and open the closets and hang them in.	10×50
	Pick up the alarm clock and push the button.	5×30
Kitchen	Pick up dishes on the table and store it into the fridge.	5×30
	Pick up the food and turn around to open the microwave and put the food in.	5×50
	Pick up the dishes on the rack and the sponge and clean the dishes.	5×50

Table 3: Dataset details for task descriptions and number of trajectories. The number of trajectories is demonstrated as number of assets \times number of recorded sequences.

63 **References**

- 64 [1] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion
65 over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.
- 66 [2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Pro-*
67 *ceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778,
68 2016.
- 69 [3] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, A. Zouitine, and T. Wolf. Lerobot: State-
70 of-the-art machine learning for real-world robotics in pytorch. [https://github.com/](https://github.com/huggingface/lerobot)
71 [huggingface/lerobot](https://github.com/huggingface/lerobot), 2024.