

Supplement Material for “UnrealZoo: Enriching Photo-realistic Virtual Worlds for Embodied AI”

Contents

1. UE Environments	2
1.1. Comparison with other Simulators	2
1.2. Environments used in Visual Navigation	2
1.3. Environments used in Active Visual Tracking	2
1.4. Navigation Mesh	2
2. Exemplar Tasks	2
2.1. Visual Navigation	2
2.2. Active Visual Tracking	2
2.3. Task Configuration in JSON File	3
3. Implementation Details of Agents	3
3.1. Data Collection for Offline RL	3
3.2. RL-based Agents	3
3.3. VLM-based Agents	4
3.4. Human Benchmark for Navigation	4
4. Additional Results	6
4.1. Learning Curve	6
4.2. Testing in 16 Unseen Environments	6
4.3. Testing in Social Tracking Scenarios	7
4.4. Cross-platform Evaluation	7
5. Limitations and Discussions.	7

1. UE Environments

1.1. Comparison with other Simulators

To better explain Table 2, we list the description of each symbol about the scene types and playable entities in Table 3. Since photorealism mainly relies on the engine used, we visualize the snapshots rendered by different engines in Figure 1. Note that Google Maps are images captured in the real world, but can not simulate the dynamic of the scenes and interactions between objects. By utilizing advanced rendering and physics engines, Unreal Engine simulates large-scale photorealistic environments that are not only visually appealing but also capable of complex interactions between agents and objects. So we choose to build environments on Unreal Engine.

1.2. Environments used in Visual Navigation

We carefully selected two photo-realistic environments (**Roof** and **Factory**) for training and evaluating navigation in the wild, shown in Figure 2. The Roof environment features multiple levels connected by staircases and large pipelines scattered on the ground, providing an ideal setting for the agent to learn complex action combinations for transitioning between levels, such as jumping, climbing, and navigating around obstacles. The Factory environment, on the other hand, is characterized by compact boxes and narrow pathways, challenging the agent to determine the appropriate moments to jump over obstacles or crouch to navigate under them. These two environments offer diverse spatial structures, enabling agents to develop an understanding of multi-level transitions and precise obstacle avoidance.

1.3. Environments used in Active Visual Tracking

For training agents via offline reinforcement learning, we selected 8 distinct environments to collect demonstrations, as is shown in Figure 5. To comprehensively evaluate the generalization of the active visual tracking agents, we selected **16** distinct environments, categorized into Interior Scenes, Palaces, Wilds, and Modern Scenes. Each category presents unique challenges: 1) **Interior Scenes** feature complex indoor structures with frequent obstacles; 2) **Palaces** include multi-level structures and narrow pathways; 3) **Wilds** encompass irregular terrain and varying illumination; 4) **Modern Scenes** offer high-fidelity, real-world scenarios with modern buildings and objects. These diverse environments facilitate a thorough assessment of the agent’s generalization capabilities across varying complexities. The snapshot of each environment is shown in Figure 3.

1.4. Navigation Mesh

Based on NavMesh, we build an internal navigation system, allowing agents to autonomously navigate with the built-in AI controller in the Unreal Engine. This includes path-

finding and obstacle-avoidance capabilities, ensuring smooth and realistic movement throughout diverse terrains and structures. Moreover, in our City style map, we manually construct road segmentation, we manually segment the roads to distinguish between pedestrian and vehicle pathways. When agents use the navigation system for autonomous control, they will navigate the shortest path based on the priority of the different areas. Figure 4 shows an example of the rendered semantic segmentation for NavMesh in an urban city.

2. Exemplar Tasks

2.1. Visual Navigation

In this task, the agent is initialized at a random location in the environment at the beginning of each episode, while the target object’s location and category remain fixed throughout. The agent must rely on its first-person view observations and the relative spatial position of the target as input. The ultimate objective is to locate the target object within 2000 steps. Success is defined by the agent reducing the relative distance to less than 3 meters and aligning its orientation such that the relative rotation between the target and the agent is smaller than 30 degrees (in the front of the agent). This setup challenges the agent to optimize its movements and decision-making while adapting to the randomized starting conditions and dynamic environment. All methods in the task share the same discrete action space to control the movement, consisting of moving forward (+1 meter/s), moving backward (-1 meter/s), turning left (-15 degrees/s), turning right (+15 degrees/s), jumping (two continuous jumping actions trigger the climbing action), crouching, and holding position. This action space enables the agent to navigate and interact with complex 3D environments, making strategic decisions in real-time to reach the target object efficiently. The step reward for the agent is defined as:

$$r(t) = \tanh\left(\frac{dis2target(t-1) - dis2target(t)}{\max(dis2target(t-1), 300)} - \frac{|Ori|}{90^\circ}\right) \quad (1)$$

where $dis2target(t)$ is the Euclidean distance between the agent and the target at a given timestep t and $|Ori|$ is the absolute orientation error (in degrees) between the agent’s current heading and the direction toward the target, normalized by 90°




















2.2. Active Visual Tracking

Referring to previous works [6], we use human characters as an agent player and a continuous action space for agents. The action space contains two variables: the angular velocity and the linear velocity. Angular velocity varies between $-30^\circ/s$ and $30^\circ/s$, while linear velocity ranges from -1 m/s to 1 m/s . In the agent-centric coordinate system, the reward

Table 1. We evaluate the simulation speed on an Nvidia GTX 4090 GPU, Intel i7-14700K CPU, and Windows OS by repeatedly calling UnrealCV+ API function 1000 times and recording the execution time. Multi-agent interaction performance is measured using the gym interface. The image resolution is set to 640×480. The table presents the FPS measurement results across environments of varying scales.

Env	Color Image	Object Mask	Surface Normal	Depth Image	2 Agents Interaction	6 Agents Interaction	10 Agents Interaction
FlexibleRoom (71 objects, 2440m ²)	85	164	137	100	94	40	27
BrassGarden (467 objects, 9900m ²)	107	214	173	123	102	48	31
Supermarket (2839 objects, 11700m ²)	99	173	167	117	70	33	19
SuburbNeighborhoodDay (2469 objects, 23100m ²)	79	139	112	92	53	27	17
GreekIsland (3174 objects, 448800m ²)	82	167	124	103	52	23	16
MedievalNatureEnvironment (8534 objects, 16km ²)	70	112	97	74	29	14	10

Table 2. The comparison with related photo-realistic virtual worlds for embodied AI.

Virtual Worlds	Scene: Categories	Scene: Scale Level	Scene: Style	Scene: Base Engine	Agent: Body	Agent: Nav. Sys.	Agent: Multi-agent
VirtualHome		Indoor	Modern, Western	Unity		✓	✓
AI2THOR		Indoor	Modern, Western	Unity		-	-
ThreeDWorld		Indoor, Building, Community	Modern, Western, Nature	Unity		-	✓
OmniGibson		Indoor	Modern, Western	Omniverse	-	-	-
Habitat 3.0		Indoor	Modern, Western	Habitat-Sim		✓	✓
CARLA		Community, Landscape	Modern, Western, Nature	UE 4/5		✓	✓
AirSim		Community, Landscape	Modern, Western, Nature	UE 4		-	✓
LEGENT		Indoor, Building	Modern, Western	Unity		✓	✓
V-IRL		Community, Landscape	Modern, Western, Nature	Google Map		✓	✓
UnrealZoo		Indoor, Building, Community, Landscape	Ancient, Modern, Sci-Fi Western, Eastern, Nature	UE 4/5		✓	✓

function is defined as:

$$r = 1 - \frac{|\rho - \rho^*|}{\rho_{max}} - \frac{|\theta - \theta^*|}{\theta_{max}} \quad (2)$$

where (ρ, θ) denotes the current target position relative to the tracker, $(\rho^*, \theta^*) = (2.5m, 0)$ represents the expected target position, i.e., the target should be 2.5m in front of the tracker. The error is normalized by the field of the view $(\rho_{max}, \theta_{max})$. During execution, an episode ends with a maximum length of 500 steps, applying the appropriate termination conditions. In the experiment, we adopt the original neural network structure and parameters, as listed in Table 6 and 7.

2.3. Task Configuration in JSON File

We provide an example of the task configuration JSON file in Figure 6. Using the JSON file, we can easily set the configuration of the binary, the continuous and discrete action space for each agent, the placement of the binding camera, choose the area to reset, and other hyper-parameters about the environments.

3. Implementation Details of Agents

3.1. Data Collection for Offline RL























To collect demonstration for offline reinforcement learning, we use state-based expert policy and the multi-level perturbation strategy [6] to automatically generate various imperfect

demonstrations as the offline dataset. For active visual tracking, we employ three distinct datasets for training agents via offline reinforcement learning (Offline RL) algorithms, referred to as *1 Env.*, *2 Envs.*, and *8 Envs.* The detailed composition of each dataset is depicted in Figure 5. For the *1 Env.* dataset, we use only the FlexibleRoom, an abstract environment enriched with diverse augmentation factors, to gather 100k steps of trajectory data. For 2 Envs., we collect 50k step trajectories from FlexibleRoom and an additional 50k steps from the Supermarket environment. The 8 Envs. dataset involves eight different environments, with 12.5k steps collected from each. Therefore, **the total amount of data in the three datasets is the same (100k) to ensure the fairness of the comparison.** These dataset configurations aim to highlight the critical role of environment diversity in enhancing the generalization capabilities of embodied AI agents.

3.2. RL-based Agents

Learning to navigate with online reinforcement learning. For navigation, we construct an RL-based end-to-end model, using A3C [3] to accelerate online reinforcement learning in a distributed manner. The model’s structure is as follows: a mask encoder extracts spatial visual features from the segmentation mask, which are then passed to a temporal encoder to capture latent temporal information. Finally, the spatiotemporal features, concatenated with the target’s relative spatial position, are fed into the actor-critic network to optimize the actor layer for action prediction. The detailed

Table 3. The description of symbols used in Table 2.

Symbol	Description
	Interior house with furnishings
	Residential community with multiple buildings
	High-fidelity large-scale urban environments
	Exterior scenes with roads
	Natural scenes with forests or grasslands
	Large-scale natural landscape, including lakes, mountains, desert
	An island landscape
	Castle-style historic buildings
	Asian temple architecture features stairs, lofts, and shrines.
	Industrial areas with internal roads and factory facilities
	Educational settings, including classrooms and gymnasiums.
	Sports venue scenes, such as swimming pool, sport stadium.
	Supermarket contains a wide range of daily essentials and produce.
	Typical urban public transportation hubs, such as train and gas stations.
	Detailed hospital interior scenes.
	Human characters with detailed features such as hair textures, clothing, and actions
	Mobile robot
	Driveable car
	Animals include common animal species such as cats, dogs, horses, pigs, etc.
	Driveable motorbike
	Drones
	Virtual camera that has no physical entity and is movable

network structure and parameters used in the experiment are listed in Table 4 and 5. Here, we provide the training curves in *Roof* and *Factory* environments, depicted in Figure 10. In the *Factory*, we set the number of workers to 4, while in the *Roof*, the number of workers is set to 6. It can be observed that, for Online RL, the number of workers and the complexity of environments have a significant impact on training efficiency. Looking forward, we anticipate that offline-based algorithms can effectively address the challenges of training efficiency and generalization.

Learning to track with offline reinforcement learning. For the tracking task, we adopt an offline reinforcement learning (Offline RL) approach to enhance training efficiency and improve the agent’s generalization to unknown environments. Specifically, we build an end-to-end model trained using offline data and the conservative Q-learning (CQL) strategy [2]. We adopt the same model structure from the latest visual tracking agent [6], consisting of a Mask Encoder, a Temporal Encoder, and an Actor-Critic network. Detailed model structures and training parameters are summarized in Table 6 and 7. Additionally, we provide the model’s loss curves under different dataset setups, as shown in Figure 9. The model achieves near-convergence within two hours

across all dataset setups. To ensure the loss curves stabilize fully, we continued training for an additional three hours, during which no significant further decrease in the loss was observed. A comprehensive evaluation of the model’s performance is presented in Tables 8 and 9, highlighting its strong generalization to unseen environments and robustness to dynamic disturbances. The training efficiency, generalization capability, and robustness achieved by offline RL further reinforce our belief that offline RL methods will become a mainstream approach for rapid prototyping and iteration in embodied intelligence systems.

3.3. VLM-based Agents

We built agents with a reasoning framework based on the Large Vision-Language Model. We employ OpenAI GPT-4o as the base model. System prompt used in the navigation task, as shown in Figure 8 and system prompt used in the tracking task, as shown in Figure 7.

3.4. Human Benchmark for Navigation

In the navigation task, we incorporated human evaluation as a baseline for comparison to demonstrate the existing gap between the current method and optimal navigation perfor-



Figure 1. Comparison of the visual realism of different engines: we show the snapshots captured from different engines to compare the photo-realism of different environments for an intuitive feeling. Note that Google Maps capture and reconstruct the images from the real world, but can not simulate the dynamic of the scenes and interactions between agents and objects.

Table 4. Details the neural network structure of RL-based agent for navigation task, where 5×5 -32S1 means 32 filters of size 5×5 and stride 1, FC256 indicates the fully connected layer with output dimension 256, and LSTM128 indicates that all the sizes in the LSTM unit are 128.

Module	Mask Encoder							
Layer#	CNN	Pool	CNN	Pool	CNN	Pool	CNN	Pool
Parameters	5×5 -32S1	2-S2	5×5 -32S1	2-S2	4×4 -64S1	2-S2	3×3 -64S1	2-S2
Module	Temporal Encoder		Actor			Critic		
Layer#	FC	LSTM	FC			FC		
Parameters	256	128	2			2		



Figure 2. Two photo-realistic environments used for visual navigation.

and image. This ensured that human evaluators had a comprehensive understanding of the environment and the target’s position. During the evaluation, the player was randomly initialized in the environment, and human evaluators used the keyboard to control the agent’s movements. Each human evaluator repeated the experiment five times, providing multiple data points to ensure reliability and reduce variability in performance measurements. The termination conditions for the evaluation were identical to those applied to the RL-based agent, ensuring consistency in the comparison.

mance. Specifically, **five male and five female** evaluators participated in the assessment, performing the same navigation tasks under comparable conditions.

Before each human evaluator began their assessment, we provided a free-roaming perspective to familiarize them with the map structure and clearly conveyed the target’s location

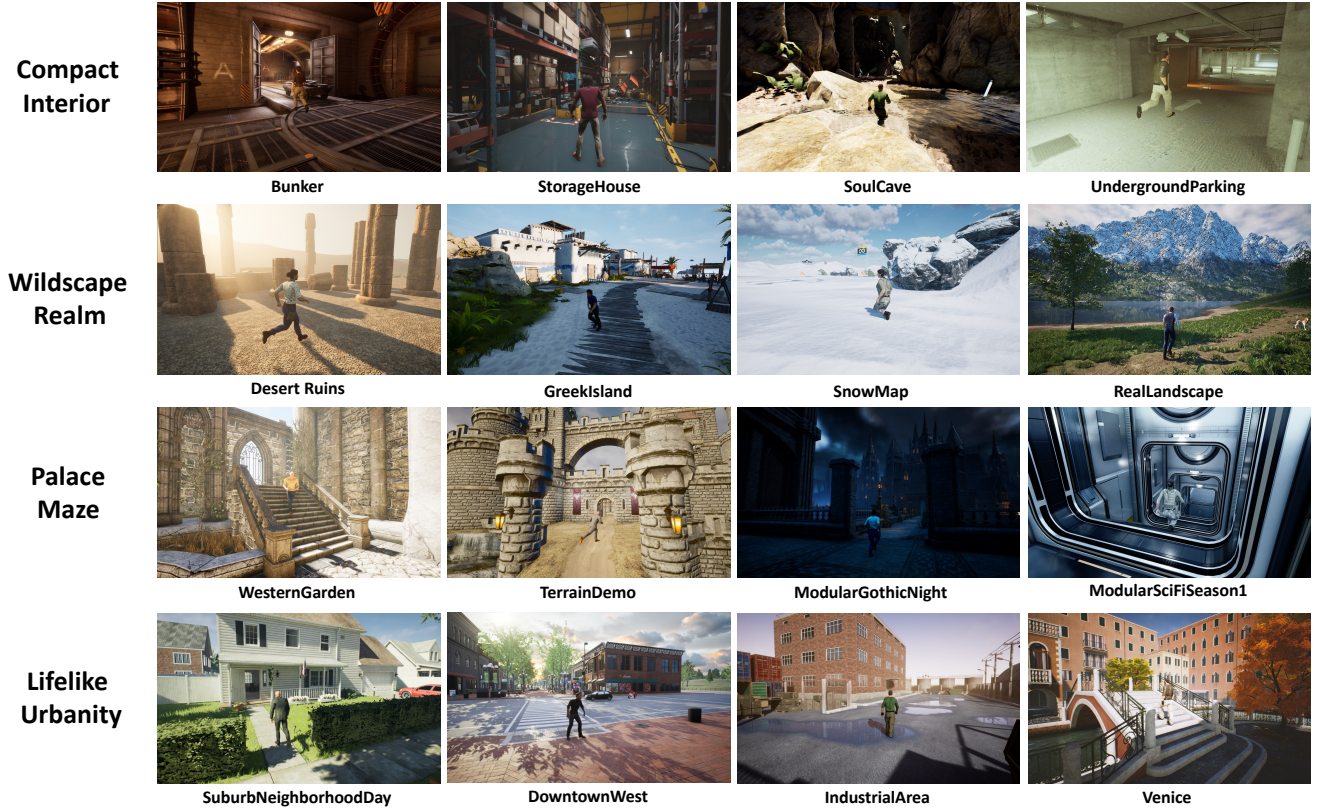


Figure 3. The snapshots of 16 environments used for testing active visual tracking agents. The text on the left indicates the category corresponding to that line of environment.

Table 5. The experiment setting and hyper-parameters used for training the RL-based navigation agent.

Name	Value	Name	Value
Learning Rate	1e-4	LSTM update step	20
workers (Roof)	6	LSTM Input Dimension	256
workers (Factory)	4	LSTM Output Dimension	128
Position Input Dimension	2	LSTM Hidden Layer size	1

4. Additional Results

4.1. Learning Curve

We provide the CQL loss curve under the *1 Env.*, *4 Envs.* and *8 Envs.* training setup. As shown in Figure 9, the offline model approaches convergence after two hours and we continued training for another three hours after nearing convergence, observing no significant further decrease in the loss. Note that the offline training was conducted on a Nvidia RTX 4090 GPU.

4.2. Testing in 16 Unseen Environments

We provide the detailed quantitative evaluation results (episodic returns, episode length, success rate) of the RL-based embodied tracking agents across 16 environments, listed in Table 8. In each environment, we report the av-

erage results over 50 episodes. The results show that in the *Palace Maze*, which contains abundant structural obstacles, the agent’s tracking performance was generally weaker compared to the other three categories. In contrast, the agent performed generally better in *Lifelike Urbanity*, characterized by its relatively regular and flat terrain. Additionally, we observed that as the diversity of the training environments increased, the agent’s tracking performance improved across all four environment categories. This highlights the positive impact of diverse training data on enhancing the agent’s overall tracking effectiveness. We also provide vivid demo videos in <https://unrealzoo.notion.site/task-evt>.

Table 6. Network structure used in the offline RL method [6], where 8×8 -16S4 means 16 filters of size 8×8 and stride 4, FC256 indicates a fully connected layer with dimension 256, and LSTM64 indicates that all sizes in the LSTM unit are 64.

Module	Mask Encoder			Temporal Encoder	Actor	Critic
Layer#	CNN	CNN	FC	LSTM	FC	FC
Parameters	8×8 -16S4	4×4 -32S2	256	64	2	2



Figure 4. An example of the NavMesh with semantic segmentation. The human character will prioritize using the pink area for pedestrian navigation tasks, while the vehicles will use the blue area.

Table 7. The hyper-parameters used for offline training and the policy network.

Parameter	Value	Parameter	Value
Learning Rate	3e-5	LSTM steps	20
Discount	0.99	LSTM In Dim	256
Batch Size	32	LSTM Out Dim	64
LSTM Hidden Layers	1		

4.3. Testing in Social Tracking Scenarios

We select 4 environments from different categories as the testing environments, including StorageHouse, DesertRuins, TerrainDemo, and SurburNeighborhoodDay. We test the distraction robustness of the social tracking agents by adding different numbers of distractors (4, 8, 10) in the environment. The distractors randomly walk around the environment, which may produce various unexpected perturbations to the tracker, such as visual distractions, occlusion, or blocking the tracker’s path. As shown in Table 9, the tracking performance of the three agents steadily decays with the increasing number of distractors.

4.4. Cross-platform Evaluation

To evaluate the generalization ability of policies trained in UnrealZoo, we conducted cross-platform experiments in both simulation and real-world settings. During our investigation, we found that many existing simulation environments [4, 5, 7] lack flexibility in task customization, making it difficult to add or modify task definitions. Among the available

options, we selected ThreeDWorld [1] for its minimal migration overhead.

For evaluation, we used the publicly available "Suburb Scene 2023" map and designated the robot Magnebot as the tracking target, as shown in Figure 11. We tested two policies trained under different environment settings: a single environment (1-Env) and eight diverse environments (8-Envs). Due to limitations in ThreeDWorld, specifically, the inability to retrieve the absolute position of Magnebot, we could not compute the Average Reward (AR) metric. Instead, we report the episode length (EL) and success rate (SR) as alternative indicators of tracking performance.

Beyond simulation, we further validated the trained policies in the real world, illustrated in Figure 11. In this scenario, the target person walks along an S-shaped path in an open area with various distractors. We conducted five trials for each policy. Since all trials were successful, the success rate alone could not reflect performance differences. Additionally, absolute position data was unavailable in the real-world setting, so we introduced two alternative evaluation metrics:

- **Average Deviation (AD):** the average pixel offset between the target’s bounding box center and the image center.
- **IoU:** the Intersection over Union between the current target bounding box and the initial bounding box.

These metrics assess the agent’s ability to maintain consistent and centered tracking under dynamic motion and environmental interference. The final results are listed in Table 10. Overall, the experimental findings highlight that training with diverse environments in UnrealZoo significantly improves the policy’s generalization ability and robustness.

5. Limitations and Discussions.

While our proposed environment provides diverse and complex scenarios for visual navigation, tracking, and other embodied vision tasks, it currently has some limitations that should be overcome in future work: 1) Limited Physical Simulation Fidelity. The current version focuses on visual rendering and interaction rather than the precision of physical interactions for control. We will consider enhancing physics engine integration for more accurate and realistic physical interactions. 2) Licensing Restrictions and Shareability. Due to marketplace content licensing restrictions, we are unable to open-source the complete environment source code for more in-depth customization. To address

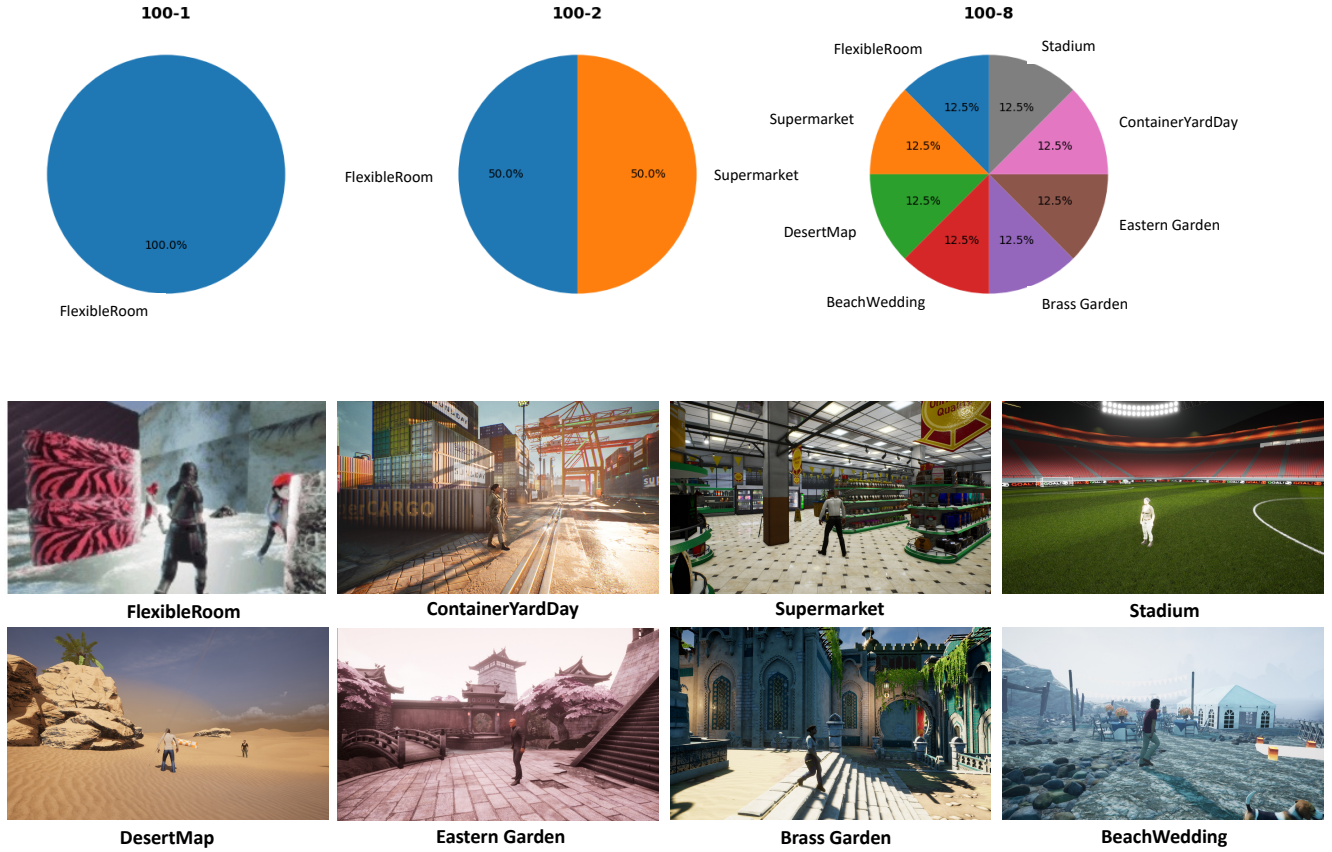


Figure 5. The 8 environments used for collecting offline dataset.

this, we will develop alternative scenes based on free assets or negotiate special licensing agreements with asset creators, and extend the UnrealCV command system to support more advanced customization in the released binary. 3) Limited Interactions. Interaction capabilities are primarily limited to pre-defined objects, and multi-agent interactions remain relatively basic, lacking complex social dynamics. We will extend to more complex interaction tasks such as object manipulation, tool use, and environmental modification, and implement more sophisticated social agent behavior models including dialogue and emotional expression.

References

- [1] Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, Megumi Sano, Kuno Kim, Elias Wang, Michael Lingelbach, Aidan Curtis, Kevin Tyler Feiglis, Daniel Bear, Dan Gutfreund, David Daniel Cox, Antonio Torralba, James J. DiCarlo, Joshua B. Tenenbaum, Josh McDermott, and Daniel LK Yamins. ThreeDWorld: A platform for interactive multi-modal physical simulation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.
- [2] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.
- [3] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.
- [4] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.
- [5] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander Clegg, Michal Hlavac, So Yeon Min, Vladimír Vondruš, Theophile Gervet, Vincent-Pierre Berges, John M Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakrishnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and Roozbeh Mottaghi. Habitat 3.0: A co-habitat for humans, avatars, and robots. In *The Twelfth International Conference on Learning Representations*, 2024.

A Json File for Task Configuration

```
"env_name": env_name ,
"env_bin": path-to-binary ,
"env_map": map_name,
"env_bin_win": path-to-binary (for windows) ,
"third_cam": {"cam_id": 0,"pitch": -90,"yaw":0,"roll":0,"height_top_view": 1460.0,"fov": 90},
"height": 460.0,
"interval": 1000,
"agents": {
  "player": {
    "name": ["BP_Character_923"],
    "cam_id": [3],
    "class_name": ["bp_character_C"],
    "internal_nav": true ,
    "scale": [1,1,1],
    "relative_location": [20,0,0],
    "relative_rotation": [ 0,0,0],
    "head_action_continuous": {"high": [15,15,15], "low": [-15,-15,-15]},
    "head_action": [ [0,0,0],[0,30,0],[0,-30,0]],
    "animation_action": ["stand","jump","crouch"],
    "move_action": [
      [angular, velocity]
      ...
    ],
    "move_action_continuous": {"high": [30,100],"low": [-30,-100]}
  },
  "animal": {
    ...
  }
  "drone": {
    ...
  }
},
...
"safe_start": [
  [x,y,z],
  ...
],
"reset_area": [x_min,x_maxin,y_min,y_max,z_min,z_max],
"random_init": false ,
"env": {"interactive_door": []},
"obj_num": 466,
"size": 192555.0,
"area": 9900.0,
"bbox": [110.0, 90.0,19.45]
```

Figure 6. An example of the task configuration file in JSON format.

System Prompt used for active tracking

Objective :

You are an intelligent tracking agent designed to control the robot to track the person in the view. The first person in your view is your target. You need to provide concrete moving strategies to help robot tracking the target in the given environment.

Representation details :

1. Moving instructions are concrete actions that the robot can take to adjust its viewpoint and distance to the target. The moving instructions include :
 - move closer: Move the robot closer to the target. This should be chosen when the target is too far away from the robot and there is no obstacle in the way.
 - move further: Move the robot further away from the target. This should be chosen when the target is too close to the robot and only part of the target body is visible in the view.
 - keep current: Maintain the current distance and angle between the robot and the target. This is chosen when the target is fully observable in the view and there is enough space in front of both tracker and target without any potential obstacles may cause collision and occlusion.
 - turn left: Turn the robot to left direction, the target will move towards the right side in next frame.
 - turn right: Turn the robot to right direction, the target will move towards the left side in next frame.

Input Understanding :

1. ****Image:**** We provide a first-person view observation of the robot to help you understand the surrounding environment. The observation is represented as a color image from the tracker's first-person perspective.

Output Understanding :

1. ****Moving Strategy:**** A temporal reasonable move strategy to adjust the robot viewpoint and distance to achieve robots' long-term tracking task. This should be represented as a concrete moving instructions, the instructions should be chosen from "move closer", "move further", "keep current", "turn left", "turn right". Format - [Keep current].

Strategy Considerations :

1. If the person's horizontal position in the robot's field of view deviates from the center by more than 25% of the image width, we consider the target to be on one side of the image, otherwise we say the target is near the center.

Instructions :

1. Provide ONLY the decision in the [output:] strictly following the format without additional explanations or additional text.

Figure 7. System prompt used for tracking.

[6] Fangwei Zhong, Kui Wu, Hai Ci, Churan Wang, and Hao Chen. Empowering embodied visual tracking with visual foundation models and offline rl. In *European Conference on Computer Vision*, pages 139–155. Springer, 2024.

[7] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual

navigation in indoor scenes using deep reinforcement learning. In *International Conference on Robotics and Automation (ICRA)*, 2017.

System Prompt used for navigation

Objective:

You are an intelligent navigation agent designed to control the robot to navigate to the target object location based on first-person observation and provide a relative position between the robot and the target. You need to provide an action sequence to help the robot move to the target location.

Representation details:

1. **Relative Position:** This contains three elements, in the format - [Distance, Direction, Height].
 - Distance: The relative distance between the robot and the target object.
 - Direction: The target object's relative direction to the robot, represented in degrees.
1. **Actions:** These are the movements the robot can perform to adjust its position. The available actions include:
 - Move Forward: Propel the robot forward by 100 centimeter.
 - Move Backward: Propel the robot backward by 100 centimeter.
 - Turn Left: Rotate the robot 15 degrees to the left.
 - Turn Right: Rotate the robot 15 degrees to the right.
 - Jump: Make the robot leap into the air, robot should use this action to jump over obstacles or climb over stairs.
 - Crouch: Lower the robot into a crouching position for 2 seconds, after which it will automatically stand up.
 - Keep Current: Maintain the robot's current position without any movement.

Input Understanding:

1. ****Image:**** We provide a first-person view observation of the robot to help you understand the surrounding environment. The observation is represented as a color image from the robot's first-person perspective.
2. ****Relative Position:**** This data provides the target object's relative position to the robot, including the distance, direction, and height. The distance is measured in centimeters, the direction in degrees, and the height in centimeters.

Output Understanding:

1. ****Action Sequence:**** This is a series of Three continuous actions that the robot should take to navigate toward the target object, in the format - [Action1, Action2, Action3]. Each action should be chosen from the available actions mentioned above.

Strategy Considerations:

1. **Assessing Relative Position:** Begin by evaluating the target object's relative position in terms of distance, direction, and height to inform the action sequence.
2. **Action Combination for Navigation:** Utilize the action sequence to create effective combinations, each action will last for 1 seconds.
3. **Obstacle Detection:** Leverage the first-person observation to identify obstacles. Based on their location, formulate action sequences that facilitate smooth navigation while avoiding collisions.

Instructions:

1. Provide **ONLY** the action sequence in the [output:] strictly following the format -[Action1, Action2, Action3], without additional explanations or additional text.

Figure 8. System prompt used for navigation.

Table 8. Quantitative evaluation results of the offline RL method across 16 environments. The environments are grouped into four categories: Compact Interior, Wildscape Realm, Palace Maze, and Lifelike Urbanity. The table compares the performance of agents trained on different offline dataset settings: 1 Env. (single environment), 2 Envs. (two environments), and 8 Envs. (eight environments). Each cell presents three metrics from left to right: Average Episodic Return (ER), Average Episode Length (EL), and Success Rate (SR).

Category	Environment Name	1 Env. ER/EL/SR	2 Envs. ER/EL/SR	8 Envs. ER/EL/SR
Compact Interior	Bunker	241/412/0.56	245/391/0.56	234/429/0.70
	StorageHouse	213 /424 /0.68	275/449/0.76	170/434/0.64
	SoulCave	229/402/0.60	252/422/0.56	206/405/0.58
	UndergroundParking	179/391/0.56	250/424/0.62	184/410/0.60
Wildscape Realm	Desert Ruins	209/392/0.54	293/449/0.70	277/453/0.70
	GreekIsland	245/411/0.62	264/423/0.64	257/466/0.78
	SnowMap	204/399/0.62	322/456/0.78	278/474/0.86
	RealLandscape	171 /383/0.42	225/372/0.44	223/444/0.70
Palace Maze	WesternGarden	230/403/0.54	209/408/0.54	296/472/0.82
	TerrainDemo	232/411/0.56	233/403/0.56	192/411/0.56
	ModularGothicNight	190/360/0.52	244/423/0.62	272/456/0.76
	ModularSciFiSeason1	168/365/0.42	172/354/0.42	211/393/0.48
Lifelike Urbanity	SuburbNeighborhoodDay	224/422/0.64	328/457/0.72	242/457/0.76
	DowntownWest	296/460/0.78	317/456/0.76	292/469/0.86
	Factory	278/434/0.64	291/452/0.74	249/435/0.64
	Venice	295/441/0.70	323/448/0.82	294/474/0.84

Table 9. Quantitative evaluation results of the tracking agents across 4 different category environments with **4 distractors (4D)**, **8 distractors (8D)**, and **10 distractors (10D)** respectively. The table compares the performance of agents trained on different offline dataset settings: 1 Env. (single environment), 2 Envs. (two environments), and 8 Envs. (eight environments). Each cell presents three metrics from left to right: Average Episodic Return (ER), Average Episode Length (EL), and Success Rate (SR).

Category	Environment Name	1 Env. ER/EL/SR	2 Envs. ER/EL/SR	8 Envs. ER/EL/SR
Compact Interior	StorageHouse (4D)	117/343/0.40	181/375/0.52	190/428/0.62
	StorageHouse (8D)	143/341/0.34	151/338/0.44	165/366/0.49
	StorageHouse (10D)	81/324/0.36	109/331/0.42	107/357/0.50
Wildscape Realm	DesertRuins (4D)	317/469/0.72	333/456/0.70	354/466/0.74
	DesertRuins (8D)	213/406/0.50	316/445/0.58	267/444/0.68
	DesertRuins (10D)	188/390/0.44	252/382/0.50	253/447/0.64
Palace Maze	TerrainDemo (4D)	221/398/0.44	286/454/0.65	312/460/0.77
	TerrainDemo (8D)	211/384/0.39	239/412/0.49	252/420/0.52
	TerrainDemo (10D)	189/377/0.36	232/404/0.48	224/429/0.66
Lifelike Urbanity	SuburbNeighborhoodDay (4D)	192/407/0.46	256/381/0.50	265/392/0.60
	SuburbNeighborhoodDay (8D)	131/325/0.36	229/369/0.48	247/385/0.56
	SuburbNeighborhoodDay (10D)	162/355/0.44	180/340/0.40	165/376/0.44

Table 10. Cross-platform evaluation on active visual tracking task

Platform	ThreeDWorld EL/SR	Real World AD/IoU
1 Env.	377/0.65	764/328
8 Envs.	493/0.8	833/393

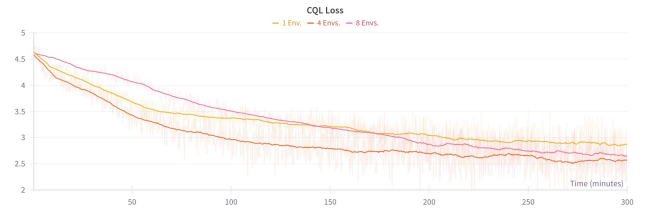


Figure 9. The CQL loss curve during offline training with different offline datasets.

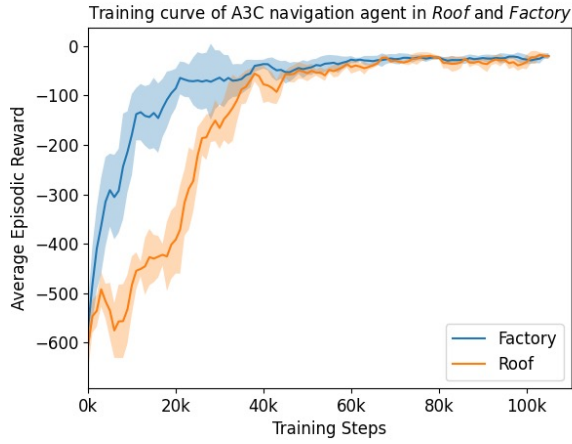


Figure 10. The learning curves for RL-based navigation agent in two environments: Roof and Factory. We use A3C [3] to learn the navigation policy via trial-and-error interactions. In the Factory (blue line plot), the number of asynchronous workers is set to 4, while in the Roof environment (orange line plot), the number of asynchronous workers is set to 6.

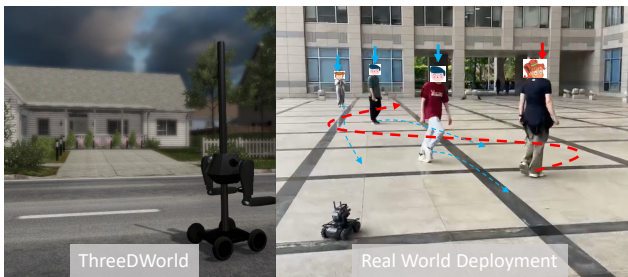


Figure 11. From left to right: 1) We utilize ThreeDWorld as the comparison simulator and use its built-in agent “Magnebot” as the target, evaluating the active tracking performance. 2) We deploy our trained policy in wheel-based robot with distractors, evaluating the active tracking performance.