FORAGERSIM: EVALUATING EMBODIED MAP-BASED REASONING IN LARGE-SCALE DYNAMIC WORLDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Animals build cognitive maps of vast environments, use them to perform map-based navigation, and might even be reusing the same circuitry used in spatial reasoning for creating and using maps of arbitrary task-spaces. In contrast, current embodied AI systems either rely on hand-engineered mapping pipelines, which break in the presence of unmodeled dynamics and are restricted to metric space, or on learned policies that rarely form map-like representations at all. Developing architectures that could learn to build and use map-like representations as flexibly as animals is, however, difficult because few embodied AI benchmarks actually require map-based reasoning. To address this, we present *ForagerSim*— an inter-disciplinary research platform and simulation environment focused on embodied tasks that require map-based reasoning, with scenarios ranging from mobile-robot search and rescue to animal foraging in both complex mazes and vast, potentially infinite open areas. *ForagerSim* is highly modular and allows modifying indepen-dent axes of difficulty related to perception (abstract 2D sensing to full RGB or LiDAR), scale (room-scale to multi-km scale), topology (open-fields to complex mazes), dynamics (smoke, debris, sensorimotor failures and others) and tasks (from simple 2D coverage to foraging-inspired well-alternation tasks). We compare reinforcement learning baselines with a simple engineered robotics stack and a human player on 8 example worlds which show how different difficulty axes affect existing methods. Our results show that, when sensing is simplified to 2D, learned policies can perform multi-object search and well-alternation tasks surprisingly well in simple topologies, but they fall short of human performance in larger or more complex-topology environments where map-based baselines perform well, suggesting that map-based reasoning is the bottleneck in these scenarios. Code and data are shared in supplementary materials and will be made open-source upon acceptance.

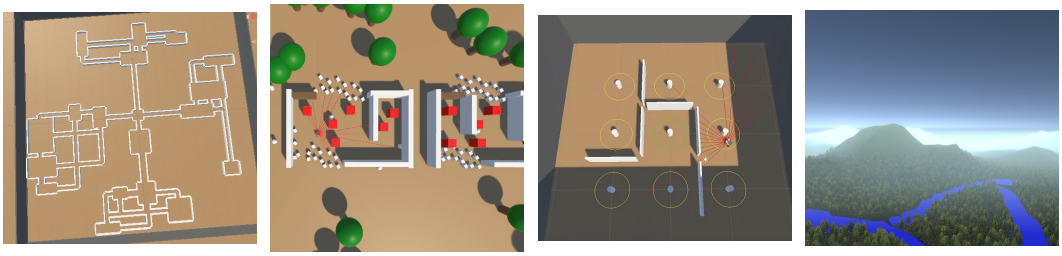


Figure 1: Example scenarios in *ForagerSim*- from left to right: The `4arm-submazes` maze; multi-object search in a dynamic-environment consisting of houses with movable rubble and chaotically walking entities; `ratmaze-foraging` experiment with rewards alternating between random wells and a home well; massive-scale realistic bat-level foraging scenario

1 INTRODUCTION

Searching for sparse objects (e.g. food) in vast, dynamic environments (deformable and changing), and then returning to known locations (e.g. a nest, a storage, or favorite feeding locations) are

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

critical capabilities for the survival of nearly all animal species. When performing this task in large environments, animals and humans do not behave purely reactively, but they build highly scalable spatial representations (cognitive maps) (Tolman, 1948; Epstein et al., 2017; Geva-Sagiv et al., 2015) which enable efficient planning, navigation, and reacting to environment changes. Bats and other animals are capable of navigating at scales as high as tens of kilometers (Goldshtein et al., 2024; Gurarie et al., 2022). Dogs trained for search and rescue missions learn to navigate rubble, enter damaged structures, and search systematically over large areas through a curriculum of reinforcement, without being told how to map or plan (Gerritsen & Haak, 2014), and without human-level language understanding. Some research even suggests that the cognitive map construction could play a significant role in general task learning in the form of learning reusable *schemata* that agents can then use for faster learning, efficient decision making, or imagining future trajectories in both physical and abstract spaces (Aronov et al., 2017; Baraduc et al., 2019; Epstein et al., 2017). Importantly, replicating the learnable map-based reasoning observed in animals and humans could enable agents to learn to perform tasks that are currently unsolvable by neither hand-crafted map-based reasoning systems nor by existing end-to-end learning approaches by combining the strengths of both. The main motivating example, which we also use as inspiration in designing some of the dynamics and topologies in *ForagerSim*, are search-and-rescue missions in environments that are both large, and highly dynamic (e.g. autonomously finding survivors in a suburb during a wildfire, flood or earthquake), which have not yet been reliably solved by any existing embodied AI approach.

Engineered mobile robotics systems, often called autonomy stacks (Tranzatto et al., 2022; Hudson et al., 2022; Petrlík et al., 2025), serve as examples showing that performing localization, mapping and planning at multiple levels of abstraction (e.g. using a global+local planner, and combining both metric and topological maps) can allow robots to search and navigate unknown environments at scales of even up to several kilometers. However, they are also built on assumptions that break in highly dynamic environments, since every new type of dynamics has to be engineered into the pipeline by hand (e.g. how the mapping and plannning has to be changed, compared to a static environment, in the case of smoke, humans or e.g. actuator or odometry failures). Even though deep learning is sometimes used in parts of these systems (e.g. for perception or low-level control), the mapping and planning modules, their interactions and their hierarchy levels are hand-designed and not learnable end-to-end.

While deep learning approaches can learn to perform arbitrary tasks in worlds with arbitrary dynamics, for which handcrafting autonomy stacks would be completely infeasible, map-like representations and online planning capabilities are severely limited in existing learning architectures. Some recent works have investigated the map-building capabilities of commonly used learned memory architectures, and found that map-like representations can emerge, but only in highly specific conditions, (Wijmans et al., 2023; Banino et al., 2018) or when a structural prior on spatial transformations is embedded into the architecture (Yang et al., 2026). While recent research has shown that the popularly used large language models can sometimes provide reasonable estimates on "where to go to search for a given object", studies have shown that they are very brittle when tested on long-horizon embodied tasks that require spatial reasoning (Paglieri et al., 2025). Some approaches have combined classical autonomy stacks with taking LLM outputs as high-level guidance, which has had success in static environments (Kim et al., 2026), but this approach still suffers from the core problems of engineered autonomy stacks (inability to learn and brittleness to noise and hard-to-model situations).

Finding learning architectures that could learn to build and utilize maps at the scales and flexibility seen in animals could benefit the robotics community (by removing the need to re-engineer mapping/planning pipelines and heuristics for different morphologies and environments), the reinforcement learning community (by helping with long-term memory and behavior), and the cognitive science community (by providing testable models of how map-like representations may arise in animals). However, each community evaluates these capabilities on its own tasks, under its own assumptions: robotics on realistic sensing in largely static worlds, reinforcement learning on memory tasks in small abstract environments, and cognitive science on animal experiments that are rarely reproduced in simulation and often only in small box-arenas. To provide a platform for these communities to collaboratively investigate the problem of learnable map-based reasoning, our paper brings the following contributions:

- **Environment** We introduce ForagerSim, an open-source environment for embodied search and navigation in which perception, scale, topology and dynamics can be varied indepen-

dently, allowing the construction of tasks that truly require map-like memory and online planning. To keep it accessible, we design the worlds such that the same scenarios can be addressed in both a fast, low-dimensionality abstract 2D depth+semantics observation mode and a more detailed, 3D mode with RGB/LiDAR sensing. We replicate scenarios from mobile robot navigation, memory-learning benchmarks and animal experiments, design new environmental topologies and dynamics which challenge existing methods and present an example curriculum learning experiment which smoothly increases topological complexity.

- **Baseline RL Scores** We benchmark a model-free and a model-based agent on 8 world configurations along 3 axes that isolate topology, dynamics, and task complexity, and find that while RL baselines show some learning on simple worlds, they are easily broken by more complex world topologies and by the more map-focused well-alternation task.
- **Comparison with a Classic Mobile-Robotics Exploration Stack** For the multi-object search settings, we compare learned approaches with a simple engineered autonomy stack implemented in ROS2 and note its high performance on complex topologies and failure cases in worlds with dynamics or non-uniform distributions of reward objects.
- **Human Reference** We record the performance of a human player and provide a robotics-inspired UI that makes the comparison fair when simulating in the fast, 2D raycasting mode. This offers an estimate of the gap between current methods and human performance.

2 Related Work

In this section, we summarize relevant existing benchmarks and simulation environments and discuss their limitations which have motivated the development of *ForagerSim*. Table 1 summarizes how *ForagerSim* differs along the axes most relevant to evaluating map-based reasoning.

Robotics Simulation Environments The most popular embodied-AI simulation environments, such as Habitat (Savva et al., 2019; Puig et al., 2024) and ProcTHOR (Deitke et al., 2022), allow simulating embodied agents in photorealistic scanned or generated household interiors, but their worlds are limited to the scale of a single building and mostly static. ExploreBench (Xu et al., 2022) is focused specifically on benchmarking exploration approaches, but its worlds are static and also limited to indoor layouts. Outdoor and large-scale simulators, such as AirSim (Shah et al., 2018), UnrealZoo (Zhong et al., 2025) or MetaUrban (Wu et al., 2025), offer simulation of various robot morphologies in large, photorealistic worlds, including urban scenes with pedestrians. However, their high rendering cost makes RL training expensive (e.g. UnrealZoo reaches 83 FPS on an RTX 4090).

Long-Horizon Behavior RL Benchmarks MemoryMaze Pašukonis et al. (2023) simulates ordered multi-object search tasks in mazes and focuses on RL memory evaluation. We replicate the same environment topology and scale in `memorymaze15x15` and show that for the task of unordered object-search, even the largest maze can easily be solved with memoryless policies. MemoryGym Pleines et al. (2025) is a newer benchmark focusing on memory on multiple videogame-like scenarios, which can be endless, but are constrained to abstract grid-world scenarios. POPGym (Morad et al., 2023) contains some grid-world maze-navigation scenarios, but the authors state that wall-following memoryless policies were able to solve these, suggesting that navigation on its own might not be a strong memory benchmark. Videogame RL wrappers like Crafter (Hafner, 2022), Craftax (Matthews et al., 2024), NetHack (Küttler et al., 2020), and MiniHack (Samvelyan et al., 2021) offer rich open-ended videogame worlds that test long-horizon behavior, but are constrained to highly abstract grid-world scenarios. Minecraft-based suites like MineRL and MineDojo (Guss et al., 2019; Fan et al., 2022) do provide infinite 3D worlds, but they inherit the slowness of Minecraft and the observations and actions are still highly abstract compared to mobile robot embodiments.

Animal-Inspired Environments Some simulation environments also take inspiration from specific animal capabilities and pose those as learning challenges. Avalon (Albrecht et al., 2022), focused on RL generalization, provides procedural 3D videogame-style island worlds with 20 embodied tasks sharing dynamics and action space. Its scenes range up to 600m in scale, but they contain simple topologies where the reward area can often be seen from the start, and the observation and action dimensionality are relatively high, due to being designed as VR-compatible. Animal-AI (Voudouris

Simulator	Worlds	Dynamic Environments	Infinite Scaling	Modular WorldGen	Interface
Explore-Bench	2D+3D cont.	✗	✗	∼	ROS1
Habitat / EAI suites	3D cont.	∼	✗	✓	Gym+ROS1
NetHack / MiniHack	2D grid	✓	✓	✓	Gym
Craftax / Crafter	2D grid	✓	✗	∼	Gym
MineRL / MineDojo	3D cont.	✓	✓	∼	Gym
Memory Maze	3D cont.	✗	✗	✗	Gym
Animal-AI	3D cont.	✓	✗	✓	Gym
Avalon	3D cont.	✓	✗	✓	Gym
AirSim	3D cont.	∼	✗	✗	Gym+ROS1
UnrealZoo	3D cont.	✓	✗	✗	Gym
MetaUrban	3D cont.	✓	✗	✓	Gym+ROS2
ForagerSim (ours)	2D+3D cont.	✓	✓	✓	Gym+ROS2

Table 1: Comparison of related robotics simulators, RL environments and benchmarks involving long-horizon embodied tasks. ✓ = implemented, ∼ = partial or limited, ✗ = absent. **Worlds**: dimensionality of the simulated worlds and whether it is a discrete gridworld (grid) or continuous (cont.); *ForagerSim* flattens its 3D worlds into a fast 2D mode with abstracted sensing. **Infinite Scaling**: the simulation is designed so that step cost does not grow with world size, e.g. by generating and simulating only the region around the agents.

et al., 2025) recreates 900 tasks from cognitive science, such as object permanence, radial-arm-maze tasks, but all within a 40x40 units wide arena, which is not suitable for large-scale navigation.

Compared to the existing environments, *ForagerSim* is the only one that combines massive scalability (through Minecraft-inspired world generation and streaming), modularity (specifically in being able to independently change axis of difficulty of topology, scale, observations, dynamics and switching between 2D and 3D variants of the same scenario), and cross-disciplinary focus (through offering both a ROS2 and Gym interface and by presenting example environments replicated from across mobile robotics, memory learning benchmarks and animal studies) which are necessary for investigating the problem of learning how to map, plan and navigate at the level of freely foraging animals.

3 THE *ForagerSim* ENVIRONMENT

ForagerSim is a domain of randomized 2D and 3D environments of various topologies, scales and dynamics designed for evaluating map-based reasoning in scenarios inspired by mobile robotics applications and animal studies. The core simulation and procedural generation logic of *ForagerSim* is implemented from scratch inside the Unity engine, which communicates with external agents via either an AI Gym (Towers et al., 2025) interface or a ROS 2 interface, with messages routed through a lightweight usage-agnostic Python wrapper of the simulator shared by both interfaces. In this section, we present some details about *ForagerSim* and the design decisions that led to them.

3.1 MODULAR SCENARIO GENERATION AND LARGE-WORLD SIMULATION

In *ForagerSim*, any given simulation scenario is fully defined by three JSON configuration files specifying the *world* (procedural generation rules, presence of dynamics, object placements etc.), the *agent(s)* (number of agents, sensors, actuators, and their noise models), and the *task* (rewards and termination conditions). Both the ROS2 and Gym interfaces use the same config files, and the simulation wrapper contains an independent tracker evaluating rewards and termination, which allows evaluating a Gym-trained RL policy and a ROS 2 autonomy stack under identical environments and the same scoring. These config files allow for easily creating new variations of existing scenarios, including the ability to define an adaptive difficulty curriculum that smoothly interpolates between values of any world-parameter (e.g. the arena size, number of rooms etc.), as visualized in Figure 7. With only a few lines of config-file changes, a user can easily create new scenarios and independently vary the following axes of difficulty:

- **Environment topology and scale**: can vary from open-spaces and suburb-like configurations to mazes (including the MemoryMaze topology from Pašukonis et al. (2023) and the rat-maze from Widloski & Foster (2022)), with potentially infinite scales due to procedural world streaming only close to the agents' sensing radius.
- **Environment dynamics**: inspired by search-and-rescue missions, we implement movable rubble, openable or lockable doors, chaotically moving walkers and moving smoke which degrades depth perception.
- **Task complexity**: we allow independently defining distributions of static rewards (e.g. in specific structures or uniformly distributed), and also switching the complexity of reward obtaining from simple static pickupable objects to 'wells' that look identically from afar and the agent must get close to to receive rewards, as is done in some cognitive science expeirments Widloski & Foster (2022)),
- **Agents' sensing and actuation configurations**: While the worlds are fundamentally 3D, the environment is designed so that sensing can run in both a fast abstracted 2D depth and semantics sensing, which can run at more than 3000 steps per second on a standard GPU-less laptop, or a 3D mode with a 3D LiDAR or RGB/RGB-D cameras. Noisy IMU or perfect pose estimation can also be used and the simulation supports multi-agent definitions.

Inspired by Minecraft's ability to generate and simulate infinite worlds, *ForagerSim* only streams the world in an area near the agent, which enables simulating missions that span tens of kilometers in scale. All procedural generation is deterministic given a seed, with each 'axis' (topology, dynamics, task) deriving their own seeds, ensuring reproducibility and modularity. Implementation details of the world generation can be found in Appendix E.

3.2 Reward Signal and Tasks

To enable evaluating multi-task learning in future work, we use a unified reward function in all our presented experiment and only change the nature of the task in each scenario through the *world* config. In any world, the agent gets a +50 reward for touching a reward object (which then disappears), a small negative reward -0.1 for colliding with other objects, and also a small reward of 0.01 for every square meter that it saw with its sensors, which helps in bootstraping learning, especially in cases where the reward objects or wells are extremely sparse. *ForagerSim* implements this spatial coverage reward by tracking the explored area covered by an agent as a 2D occupancy grid spanning the world, updated using the ground-truth pose and raycast data (even if the agent's own observations are noisy) as in traditional occupancy mapping approaches Yamauchi (1997)). We evaluate two principial task variations:

Multi-Object Search. In these world configurations, the reward objects (red cubes, with different semantic classes than obstacles) are spawned in the world and only need to be found and touched to be obtained. The spawn locations differ between worlds, but we keep a fixed maximum number of spawned objects per world config. Since our results show that this task can be sometimes be solved with reactive approaches in simpler topologies, we also designed the following task:

Foraging (well alternation). These world configrations have no freely placed reward objects, but rather they contain a variable number of *wells*, which are static objects with different semantic classes than obstacles. To obtain rewards from a well, the well must first be in a primed state (invisible to the agent from distance) and the agent must stand for a specified number of timesteps near the well, after which a number of reward objects is spawned next to the well and the agent can pick them up, just like in multi-object search. In our experiments, we recreate the well-alternation pattern from Widloski & Foster (2022), where the agent starts next to a "home" well, and the well priming alternates between one randomly selected well, followed by the home well, ad infinitum. The reasoning behind this pattern is that it simulates the scenario of an animal repeatedly going away from its home to search for food, and then returning to one known location (e.g. nest), which should benefit from building and keeping a map of the environment and is harder to solve with random-walk approaches than the basic multi-object search. As in Widloski & Foster (2022), we also provide a directional signal for guiding the agent to the random wells, but the signal is turned off for the home well, forcing the agent to utilize map-based reasoning to return home efficiently.

3.3 Observations, Actions, and Embodiment

Observation space. *ForagerSim* supports a range of sensors common to mobile robotics: RGB and RGBD cameras, 2D lidar, 3D lidar, GPS, compass, and general odometry. To isolate long-horizon embodied behavior from perception while allowing fast simulation, we design and use an abstraction on sensing, which we dub the *2D Semantic Depth Sensor* (2DSDS), visualized in Figure 2. The observation is an $N \times (D + 1)$ tensor, where N is the number of rays cast in the horizontal plane (19 in our experiments) and each ray returns a distance normalized by the sensor's max range together with a D-dimensional one-hot class embedding. Language labels for classes are also available but not used in our experiments. This abstraction is groundable in current mobile robot systems: depth sensors are standard in modern autonomy stacks, and semantic segmentation can be assumed to work robustly at moderate distances with modern foundation models (Kirillov et al., 2023). The 2DSDS sensing abstraciton allows the simulation to run at over 3000 steps per second on a standard CPU-only laptop, enabling large-scale RL experiments in large-scale dynamics worlds with millions of environment steps at negligible computational cost. In the well-alternation task, we also give the agent cues that indicate the direction to the next random well (but never the home well). We also allow simulating sensory failures and faults which are common in field robotics but often neglected in embodied AI research, such as odometry drift, lens occlusion and noise generated by smoke particles (used e.g. in the `5houses-sar` world).

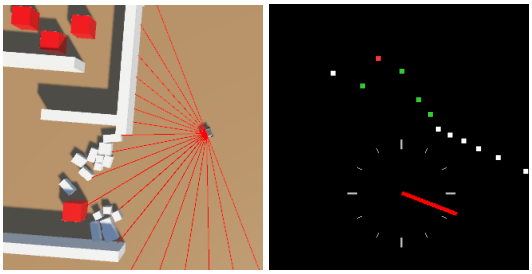


Figure 2: Illustration of the agent's observations. Left: top-down view of the environment and the raycasts. Right: human-UI visualization of the depth, semantics and compass data.

Action space and embodiment We use an idealized velocity controller discretized into 3 linear velocity and 5 angular velocity bins. We also offer acceleration-based control and simulating noise in the controls, or even complete failures that could be encountered in real-world robotic missions such as the inability to turn one way or the inability to move backwards. The agent is a spherical 1-unit wide object moving in the world, able to push dynamic objects with its movement.

4 Experiments

In this section, we present evaluations of multiple baseline approaches on the 8 world configurations described in Appendix A, and curriculum learning experiments on the hard-topology worlds in subsection 4.2. We have evaluated the following RL baselines, a handcrafted classical autonomy stack and a human player. A complete list of hyperparameters is available in Appendix F.

- **DreamerV3** As a model-based baseline with learned memory we use DreamerV3 Hafner et al. (2025) (official JAX implementation, 12M-parameter preset), whose recurrent latent state is the policy's only memory. Since the 2DSDS observation is a vector rather than an image, the encoder and decoder reduce to their MLP branches. We also test the Memory-Maze Pašukonis et al. (2023) memory variants (truncated BPTT with carried state, longer sequences) in subsection 4.2.
- **PPO** As a model-free, memoryless baseline we use PPO Schulman et al. (2017) from Stable-Baselines3 Raffin et al. (2021) with the default two-layer MLP actor and critic on the flattened observation, without frame stacking or recurrence.
- **Frontier Explorer** For the multi-object search worlds, we implemented a standard frontier-based exploration Yamauchi (1997) stack in ROS2. It builds a 2D occupancy grid using the pose and raycast data, greedily selects the nearest frontier cell on that grid, plans a path to it, follows the path, and selects the next frontier. The stack also switches to navigating towards a reward object whenever it sees one, then switches back to exploration. More implementation details are in Appendix D.

- **Human Player** - The data was collected from one person and is meant to serve as an estimated upper bound for evaluating performance of the other approaches. The player first played a few episodes of each scenario to get an understanding of each of them, and then was evaluated on before unseen seeds of the scenarios.

4.1 Testing Effects of Topology, Dynamics and Task Complexity

We evaluate the learned, engineered, and human baselines on 8 representative world configurations generated by *ForagerSim*, with example trajectories from eval runs shown in Figure 5 and in Appendix C. We separate the scenarios into simple multi-object search in static environments, multi-object search in dynamic environments, and the well-alternation task. The frontier exploration baseline is not evaluated on the alternation task as the behavior module would require significant reengineering for this specific task (both in the behavior and in having to do SLAM since we do not provide the agent's position as input in those experiments). We originally ran the well-alternation task with the position-relative-to-spawn observation given to the agent, which allowed Dreamer to discover a trivial solution in the sense of "if well-direction-signal is not available -> navigate to position (0,0) and dodge obstacles reactively". For this reason, the well-alternation runs shown in Table 2 do not use position observations. The human does not receive position information in any of the experiments, but still can still outperform baselines in nearly all runs.

We trained the RL baselines for a different number of training steps per scenario since the time to convergence varied drastically across scenarios (e.g. 1.5M steps in `memorymaze15x15` vs 60M steps in `forest-foraging`), with 3 seeds per method. Episode lengths varied from 1000 to 5000 across scenarios. We trained using 4 parallel envs per training run and A100 GPUs for Dreamer (PPO on CPU). Full training curves are shown in Figure 8 and detailed world descriptions in Appendix A. To obtain the resulting data in Table 2, we evaluated all training seeds from each fully trained baseline on a set of 5 separate randomly generated worlds from each scenario. Results of the eval runs are shown in Figure 3 and Table 2, with object and coverage metrics in Table 3.

Our main general observation is that in essentially all the settings we tested, in both the coverage task and the well-alternation task, PPO and Dreamer converged to random-walk-like behaviors which often inefficiently revisited areas multiple times. We found that these learned strategies can reach relatively high performance in surprisingly large scales and topologies , especially if the environment is bounded and highly interconnected, but they fail on more complex topologies in which, in general, a wrong turn can lead to very disadvantageous state (e.g. getting lost in one of the sub-mazes of the `4arm-submazes` topology).

One example of an easy scenario in the multi-object search task is the largest variant of the Memory-Maze topology which we reproduced from Pašukonis et al. (2023), in which all our tested baselines reach a near perfect score, suggesting that coverage-oriented tasks in indoor-scale environments do not require any mapping or planning at all (which might not be that surprising given the popularity of even the cheaper, non-mapping robot vacuums). We then tested whether a larger environment with more 'bottlenecks' could be a better test of map-based reasoning and found that indeed, on the `10houses-fenced` scenario, the random walk-like policy learned by the RL baselines struggled on account of often revisiting the fenced plots. This scenario also serves as a failure case of the frontier explorer, since uniform coverage is not feasible in the alloted episode length, whereas a human player is able to understand that the rewards are always within the houses and can exploit this to find all the reward objects in time. In the `4arm-submazes` topology, both PPO and Dreamer struggled significantly and did not find a policy that would search the whole environment, while it was fairly easy for both a human and the frontier explorer to cover the maze fully.

To test the effect of environment dynamics, we simulated a setup of 5 randomly scattered houses in a bounded environment (which was a trivial topology for all baselines to fully cover when no dynamics were present) and tested first a setup where the agent has to push through rubble to get inside houses in the `5houses-dynamic` world. In `5houses-sar`, we kept the same topology but made the dynamics harder by making just one of the rubble-walls movable (one was always too heavy to push through although they looked the same) and by adding sensor-blocking smoke surrounding the houses and chaotically walking objects. As expected, both scenarios broke the frontier exploration approach, since it assumes a static environment. Additionally, PPO struggled to learn in the harder-dynamics scenario, but Dreamer managed to reach near-human performance on both.

To test the effect of task complexity, we evaluated multiple worlds with the well-alternation task. We observed that in the `ratmaze-foraging` scenario (a near 1:1 recreation of the experiment done in Widloski & Foster (2022)), Dreamer achieved approx. 75% of human performance, which can be attributed to the fact that the environment is small, bounded and well-connected and thus Dreamer was able to reach this score even with a relatively simple policy that revisited wells often. For this reason, we then ran experiments in an open-ended environment in `forest-foraging` (where trees continue to spawn infinitely even if the agent leaves the 'content area' with the landmark objects and wells). Surprisingly, we found that Dreamer was able to learn a policy that mostly stays in the 'content area', but its overall performance was lower than human and had a very high per-world variance. We also tested the same topology as in `4arm-submazes`, but with the well-alternation task, and found that here the RL baselines struggle extremely compared to human performance, indicating that it is the topology itself that is difficult to handle without map-based reasoning.

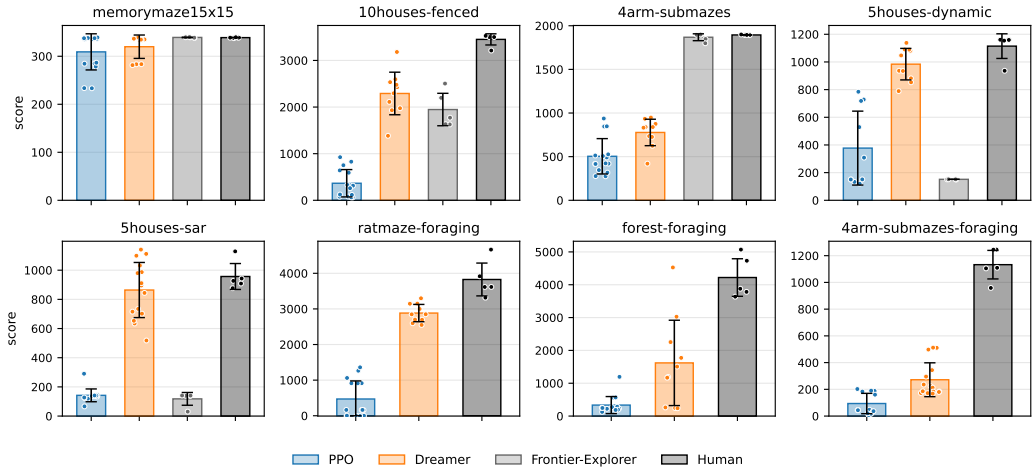


Figure 3: Bar charts of all the eval runs for each of the 8 scenarios described in subsection 4.1

World	PPO	Dreamer	Frontier-Explorer	Human
memorymaze15x15	$309_{\pm 39}$	$320_{\pm 26}$	$\mathbf{339}_{\pm 1}$	$\mathbf{339}_{\pm 1}$
10houses-fenced	$366_{\pm 303}$	$2292_{\pm 481}$	$1947_{\pm 389}$	$\mathbf{3452}_{\pm 134}$
4arm-submazes	$504_{\pm 210}$	$777_{\pm 160}$	$1868_{\pm 44}$	$\mathbf{1895}_{\pm 3}$
5houses-dynamic	$377_{\pm 282}$	$984_{\pm 120}$	$152_{\pm 0}$	$\mathbf{1114}_{\pm 100}$
5houses-sar	$142_{\pm 45}$	$864_{\pm 196}$	$118_{\pm 49}$	$\mathbf{957}_{\pm 99}$
ratmaze-foraging	$470_{\pm 525}$	$2883_{\pm 256}$	–	$\mathbf{3827}_{\pm 515}$
forest-foraging	$335_{\pm 269}$	$1619_{\pm 1370}$	–	$\mathbf{4220}_{\pm 639}$
4arm-submazes-foraging	$94_{\pm 79}$	$272_{\pm 131}$	–	$\mathbf{1133}_{\pm 119}$

Table 2: Benchmark results on the 8 world configurations. All numbers show mean per-episode return and its standard deviation, computed from 5 eval seeds per scenario, averaged across training seeds

4.2 Difficulty Curriculum Experiment

Based on the low performance of learned baselines in `4arm-submazes-foraging` (not only the foraging task but also the multi-object search), we investigated whether Dreamer could learn better if the scale and complexity of the environment is smoothly increased as the agent's performance rises. Similar to Portelas et al. (2020), we use an adaptive difficulty curriculum where we define a single scalar "difficulty" (0-1) which controls the scale of the environment, mainly the length of the maze center-arms and the number of rooms in each submaze, visualized in Figure 7. The curriculum defines a 'success' as finding 8 objects or more during one episode,

which corresponds to navigating to a random well, back to the home well and to one more random well. On a successful episode, the difficulty rises by 0.025, otherwise it decreases by 0.025. Since we run multiple environments in parallel (4 in our experiments), we track one difficulty value for all parallel envs, and we apply the step up or down whenever any one env finishes an episode. We additionally try to aid Dreamer's learning by increasing memory-related parameters. The bl256 variant lengthens the backpropagation window to 256 steps, while consec4 keeps the 48-step gradient window but chains 4 consecutive windows with the recurrent state carried across them. We also implement the trunacted backpropagation through time (TBTT) used in Pašukonis et al. (2023) where the authors note that it increased performance in a memory task in mazes. Figure 4 shows how the adaptive difficulty changes during training for each variant. Each variant converged betweeen 0.4 and 0.6 difficulty (tbtt and consec4 slower than bl256 and baseline), and neither was able to reach the full difficulty of `4arm-submazes-foraging`.

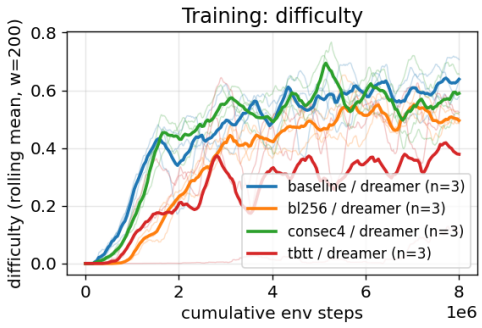


Figure 4: Adaptive difficulty curves for the curriculum experiment described in subsection 4.2.

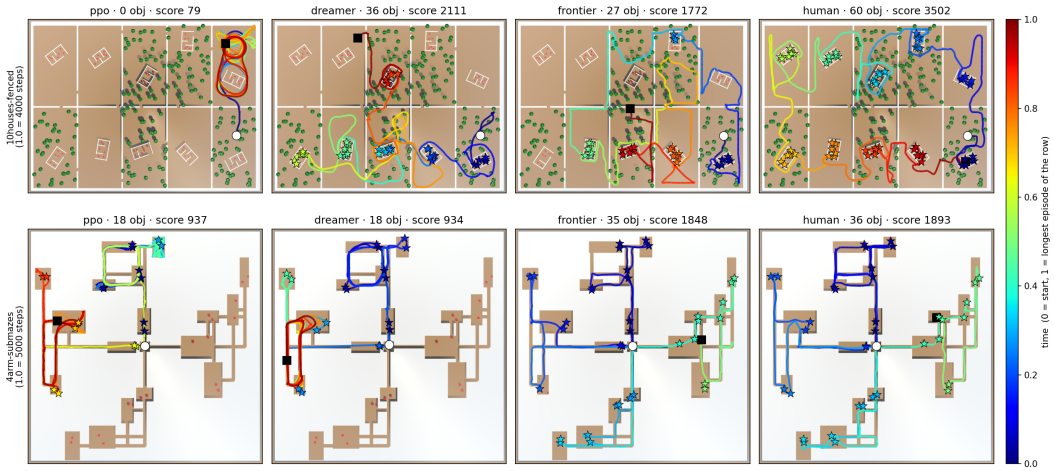


Figure 5: Visualization of eval trajectories on the `10houses-fenced` and `4arm-submazes` topologies. Color of trajectory indicates step index in episode, white circle = start, white square = end, stars = reward object pickups.

5 Conclusion

We have presented *ForagerSim*, which we designed as a cross-disciplinary platform for evaluating both learned and engineered approaches on tasks that require map-based reasoning. Through quantitative comparison of RL baselines, human performance, and the performance of a classical map-based autonomy stack on 8 representative scenarios, we show that environment topology is the main factor exposing the limitations of RL baselines in both the coverage and well-alternation tasks, and our results suggest that the presented well-alternation task is a stronger test of map-based reasoning than pure search or coverage since it requires the agent both to explore and to return to known locations. Furthermore, we performed an adaptive-curriculum expeirment in the complex maze scenario, smoothly scaling the maze size and complexity and found that neither this curriculum nor memory-related parameter tuning helped Dreamer to solve the task at full difficulty. The comparatively high performance of the simple autonomy stack and the human suggests that map-based reasoning is a key limiting factor of RL baselines in these scenarios, making *ForagerSim* well-suited for future development of end-to-end learnable map-based reasoning architectures.

AI USE STATEMENT

In this work, we used generative AI agents during the implementation of the majority of the *ForagerSim* simulator (although the core of the architecture was written by hand), the training scheduler and evaluation tooling, the analysis and plotting scripts, the TBTT training variant, the frontier-explorer method, and code documentation. We have not used generative AI tools for generating synthetic datasets, developing theoretical models or conceptual frameworks, proposing or refining hypotheses, designing or providing feedback on research methodology, assisting with translation, cleaning or reformatting of datasets, supporting qualitative or thematic data analysis or interpreting results. The categories of 'formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs,' are not applicable to this work as we have no mathematical claims or proofs. Additionally, we used generative AI tools for double-checking whether the parameters written in the manuscript reflect those in the codebase, searching for potential missed related works to cite, and for improving the readability of the paper by fixing typos, grammar and by chatting about the hard-to-read sections of the draft. We have reviewed all AI-assisted work. The majority of the code for which we used AI assistants was easy to check due to the nature of the work, for example the world generation code was checked by generating multiple worlds and seeing if the code did what the authors intended. The ROS2-based frontier-exploration node's behavior was checked by visualizations both in the simulator and through standard RVIZ visualizations of the occupancy map, planned path etc. The evaluations were verified by running the evaluations inside the Unity editor (i.e. not in headless mode) and observing whether the analyzed data (the trajectory, score etc.) matched the behavior of the agent in the simulation. The training-related code written with the help of AI agents is mostly wrappers, environment adapters and orchestration of runs and logging, with the core training logic unchanged from the Stable-Baselines3 and the Dreamer's original JAX implementation. We verified the TBTT implementation by logging whether the recurrent state is carried correctly. The training data was visualized through WeightsAndBiases. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work presents a simulation environment and benchmark. It involves no personally identifiable data and no experiments on animals. The only human data are the gameplay episodes of a single volunteer participant collected for the human reference baseline, with informed consent and without any personal information being recorded. Some environments ForagerSim are inspired by search-and-rescue scenarios and simulate only abstract agents, obstacles and reward objects, it does not model or simulate any harm to people. We release it to support research in robotics, reinforcement learning and cognitive science and see no immediate risks beyond those common to the field and the related simulation environments. We reuse the experimental protocol of Widloski & Foster (2022) in simulation only, no animal data are used.

REPRODUCIBILITY STATEMENT

All experiments in this paper use the ForagerSim simulator, which we share as a Linux build together with the Unity source code, Python wrapper, the Gym and ROS 2 interfaces, the eight world, agent and task configuration files used in Section 4, and the training and evaluation scripts, in the supplementary material. They contain a README file with instructions for running the simulation both in a human-control mode and for running the training, evaluation and analysis as done in this paper. World generation is deterministic given the seeds listed in the configuration files (saved in a JSON alongside the eval data). Hyperparameters for all RL baselines, including every deviation from library defaults, are listed in Appendix F. The frontier-explorer implementation is described in Appendix D. The evaluation protocol (number of training seeds, evaluation worlds and episode lengths) is stated in subsection 4.1 and Appendix A. The recorded trajectories from all baselines including the human reference are shared in the supplementary materials.

REFERENCES

Joshua Albrecht, Abraham J. Fetterman, Bryden Fogelman, Ellie Kitanidis, Bartosz Wróblewski, Nicole Seo, Michael Rosenthal, Maksis Knutins, Zachary Polizzi, James B. Simon, and Kanjun Qiu. Avalon: A Benchmark for RL Generalization Using Procedurally Generated Worlds. In

Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track, 2022.

Dmitriy Aronov, Rhino Nevers, and David W. Tank. Mapping of a non-spatial dimension by the hippocampal–entorhinal circuit. *Nature*, 543(7647):719–722, March 2017.

Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J. Chadwick, Thomas Degris, Joseph Modayil, Greg Wayne, Hubert Soyer, Fabio Viola, Brian Zhang, Ross Goroshin, Neil Rabinowitz, Razvan Pascanu, Charlie Beattie, Stig Petersen, Amir Sadik, Stephen Gaffney, Helen King, Koray Kavukcuoglu, Demis Hassabis, Raia Hadsell, and Dharshan Kumaran. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433, 2018. doi: 10.1038/s41586-018-0102-6.

Pierre Baraduc, Jean-René Duhamel, and Sylvia Wirth. Schema cells in the macaque hippocampus. *Science*, 363(6427):635–639, February 2019.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Russell A. Epstein, Eva Zita Patai, Joshua B. Julian, and Hugo J. Spiers. The cognitive map in humans: Spatial navigation and beyond. *Nature Neuroscience*, 20(11):1504–1513, 2017. doi: 10.1038/nn.4656.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2022.

Resi Gerritsen and Ruud Haak. *K9 Search and Rescue: A Manual for Training the Natural Way*. K9 Professional Training Series. Dog Training Press (Brush Education), Edmonton, 2 edition, 2014. ISBN 9781550594478.

Maya Geva-Sagiv, Liora Las, Yossi Yovel, and Nachum Ulanovsky. Spatial cognition in bats and rats: From sensory acquisition to multiscale maps and navigation. *Nature Reviews Neuroscience*, 16(2):94–108, February 2015.

Aya Goldshtein, Xing Chen, Eran Amichai, Arjan Boonman, Lee Harten, Omer Yinon, Yotam Orchan, Ran Nathan, Sivan Toledo, Iain D. Couzin, and Yossi Yovel. Acoustic cognitive map–based navigation in echolocating bats. *Science*, 386(6721):561–567, 2024. doi: 10.1126/science.adn6269.

Eliezer Gurarie, Chloe Bracis, Angelina Brilliantova, Ilpo Kojola, Johanna Suutarinen, Otso Ovaskainen, Sriya Potluri, and William F. Fagan. Spatial memory drives foraging strategies of wolves, but in highly individual ways. *Frontiers in Ecology and Evolution*, 10:768478, 2022. doi: 10.3389/fevo.2022.768478.

William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden R. Codel, Manuela M. Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. In *International Joint Conference on Artificial Intelligence*, pp. 2442–2448, 2019. doi: 10.24963/ijcai.2019/339.

Danijar Hafner. Benchmarking the Spectrum of Agent Capabilities. In *International Conference on Learning Representations (ICLR)*, 2022.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *Nature*, 640(8059):647–653, 2025. doi: 10.1038/s41586-025-08744-2.

Nicolas Hudson, Fletcher Talbot, Mark Cox, Jason Williams, Thomas Hines, Alex Pitt, Brett Wood, Dennis Frousheger, Katrina Lo Surdo, Thomas Molnar, Ryan Steindl, Matt Wildie, Inkyu Sa, Navinda Kottege, Kazys Stepanas, Emili Hernández, Gavin Catt, William Docherty, Brendan Tidd, Benjamin Tam, Simon Murrell, Mitchell Bessell, Lauren Hanson, Lachlan Tychsen-Smith, Hajime Suzuki, Leslie Overs, Farid Kendoul, Glenn Wagner, Duncan Palmer, Peter Milani, Matthew

O'Brien, Shu Jiang, Shengkang Chen, and Ronald C. Arkin. Heterogeneous Ground and Air Platforms, Homogeneous Sensing: Team CSIRO Data61's Approach to the DARPA Subterranean Challenge. *Field Robotics*, 2:595–636, 2022. doi: 10.55417/fr.2022021.

Seungchan Kim, Omar Alama, Dmytro Kurdydyk, John Keller, Nikhil Keetha, Wenshan Wang, Yonatan Bisk, and Sebastian Scherer. RAVEN: Resilient aerial navigation via open-set semantic memory and behavior adaptation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026. arXiv:2509.23563.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3992–4003, 2023. doi: 10.1109/ICCV51070.2023.00371.

Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The NetHack Learning Environment. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Thomas Jackson, Samuel Coward, and Jakob Nicolaus Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35104–35137. PMLR, 2024.

Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. POP-Gym: Benchmarking partially observable reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2023.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wołczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob N. Foerster, Jack Parker-Holder, and Tim Rocktäschel. BALROG: Benchmarking agentic LLM and VLM reasoning on games. In *International Conference on Learning Representations (ICLR)*, 2025.

Jurgis Pašukonis, Timothy P. Lillicrap, and Danijar Hafner. Evaluating long-term memory in 3d mazes. In *International Conference on Learning Representations (ICLR)*, 2023.

Matěj Petrlík, Pavel Petráček, Vít Krátký, Tomáš Musil, Yurii Stasinchuk, Matouš Vrba, Tomáš Báča, Daniel Heřt, Martin Pecka, Tomáš Svoboda, and Martin Saska. UAVs Beneath the Surface: Cooperative Autonomy for Subterranean Search and Rescue in DARPA SubT. *IEEE Transactions on Field Robotics*, 2:643–689, 2025. doi: 10.1109/TFR.2024.3492160.

Marco Pleines, Matthias Pallasch, Frank Zimmer, and Mike Preuss. Memory gym: Towards endless tasks to benchmark memory capabilities of agents. *Journal of Machine Learning Research*, 26(6): 1–40, 2025.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep RL: A short survey. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4819–4825, 2020. doi: 10.24963/ijcai.2020/671.

Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William 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 *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.13724.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Küttler, Edward Grefenstette, and Tim Rocktäschel. MiniHack the Planet: A Sandbox for Open-Ended Reinforcement Learning Research. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9338–9346, 2019. doi: 10.1109/ICCV.2019.00943.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics (FSR)*, volume 5 of *Springer Proceedings in Advanced Robotics*, pp. 621–635. Springer, 2018. doi: 10.1007/978-3-319-67361-5_40. arXiv:1705.05065.

Edward C. Tolman. Cognitive maps in rats and men. *Psychological Review*, 55(4):189–208, 1948.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John Balis, Gianluca De Cola, Tristan Deleu, et al. Gymnasium: A Standard Interface for Reinforcement Learning Environments. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025.

Marco Tranzatto, Takahiro Miki, Mihir Dharmadhikari, Lukas Bernreiter, Mihir Kulkarni, Frank Mascarich, Olov Andersson, Shehryar Khattak, Marco Hutter, Roland Y. Siegwart, and Kostas Alexis. CERBERUS in the DARPA Subterranean Challenge. *Science Robotics*, 7(66):eabp9742, 2022. doi: 10.1126/scirobotics.abp9742.

Konstantinos Voudouris, Ibrahim Alhas, Wout Schellaert, Matteo G. Mecattaf, Ben Slater, Matthew Crosby, Joel Holmes, John Burden, Niharika Chaubey, Niall Donnelly, Matishalin Patel, Marta Halina, José Hernández-Orallo, and Lucy G. Cheke. The animal-ai environment: A virtual laboratory for comparative cognition and artificial intelligence research. *Behavior Research Methods*, 57(4):107, 2025. doi: 10.3758/s13428-025-02616-3.

John Widloski and David J. Foster. Flexible rerouting of hippocampal replay sequences around changing barriers in the absence of global place field remapping. *Neuron*, 110(9):1547–1558.e8, May 2022.

Erik Wijmans, Manolis Savva, Irfan Essa, Stefan Lee, Ari S. Morcos, and Dhruv Batra. Emergence of maps in the memories of blind navigation agents. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*, 2023.

Wayne Wu, Honglin He, Jack He, Yiran Wang, Chenda Duan, Zhizheng Liu, Quanyi Li, and Bolei Zhou. MetaUrban: An embodied AI simulation platform for urban micromobility. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2407.08725.

Yuanfan Xu, Jincheng Yu, Jiahao Tang, Jiantao Qiu, Jian Wang, Yuan Shen, Yu Wang, and Huazhong Yang. Explore-Bench: Data sets, metrics and evaluations for frontier-based and deep-reinforcement-learning-based autonomous exploration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6225–6231, 2022. doi: 10.1109/ICRA46639.2022.9812344.

B. Yamauchi. A Frontier-Based Approach for Autonomous Exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97*, pp. 146–151, 1997. doi: 10.1109/CIRA.1997.613851.

Fan Yang, Per Frivik, David Hoeller, Chen Wang, Cesar Cadena, and Marco Hutter. Spatially-enhanced recurrent memory for long-range mapless navigation via end-to-end reinforcement learning. *The International Journal of Robotics Research*, 45(11):1682–1706, 2026. doi: 10.1177/02783649251401926.

Fangwei Zhong, Kui Wu, Churan Wang, Hao Chen, Hai Ci, Zhoujun Li, and Yizhou Wang. UnrealZoo: Enriching photo-realistic virtual worlds for embodied AI. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. arXiv:2412.20977.

A Detailed Scenario Descriptions

Here we describe the 8 presented scenarios in more detail and note how many steps are in each episode in any given scenario. The episode lengths for each scenario were designed to be solvable by a human player. Visualizations of 2 random seeds for each scenario are shown in in Figure 6, and visualizations of the effect of the difficulty in the curriculum experiment on the environmental scale and topology are visualized in Figure 7.

A.1 Static-Environment Multi-Object Search

`memorymaze15x15` (1000 steps, 75×75 m, 6 reward objects) is a recreation of the largest MemoryMaze environment from Pašukonis et al. (2023), implemented by rewriting the generation code used by the authors in DMLab into C# inside the Unity project of *ForagerSim*. The environment is generated by placing a set of 6 rooms which contain 1 reward object each and 3 rooms that contain no reward (mimicking the placement of reward objects from the original, although in our task the agent can pick up the objects in any order), and the rooms are connected by corridors to be fully connected. The topologies remain the same as in the original, the only change is that we do not color the walls as in MemoryMaze.

`10houses-fenced` (4000 steps, 298×220 m, 60 reward objects) is a scenario designed to emulate a section of a suburb in an outdoor search-and-rescue scenario, with 10 plots of land arranged in two rows, each with a single house and in 50% cases with dense vegetation as well, where the plots are separated by a smooth wall which has 1 small hole at a random position between any 2 adjacent plots. Each house contains 6 randomly placed reward objects, no objects can be found outside the houses. Each plot is 56x104 meters, which, compared to the sensor range of 30 meters forces the agent to *search* for both the house, the entrance to the house and for the hole between adjacent cells, which is even harder in the vegetation-dense cells since they sometimes block these objects.

`4arm-submazes` (5000 steps, 200×200 m, 36 reward objects) demonstrates the multi-object search task in a maze topology which we designed specifically to break any random-walk strategies, which the RL baselines often converged to. The maze topology consists of 1 central chamber, with 4 arms leading in orthogonal directions from it, and a submaze at the end of each arm. Each submaze is generated by randomly placing randomly shaped rectangular rooms and connecting them so that they are fully connected within the submaze, but only remain connected to the other submazes through the central chamber. This way, policies that do not explicitly stop themselves from revisiting areas often spend a large portion of the episode wandering inside the submazes.

A.2 Dynamic-Environment Multi-Object Search

`5houses-dynamic` (2000 steps, 130×130 m, 20 reward objects) is a multi-object search scenario designed to show the effects of simple world dynamics in a simple world topology where 5 house objects are randomly placed and rotated within a box arena. Each house contains 4 randomly placed reward objects. This is a topology with no bottlenecks and without the dynamics, both RL policies and the frontier baseline are able to search it fully. The dynamics are simple in the sense that each house is fully enclosed with the exception of 2 walls which are converted into 'rubble walls'. However, even with these simple dynamics (motivated by a real real-world problem of a robot having to push a few objects out of its way to get somewhere) force the frontier exploration approach to never reach inside the houses and would require some scenario-specific engineering to force the agent to push through the boxes and to clear its map of the dynamic objects. These rubble walls consist of approx. 15 movable boxes of variable sizes which block the view into the house and which have a different semantic class than obstacles and reward objects. The agent can reach inside the house simply by pushing through the rubble, however it then has to maneuver around the rubble inside the house.

`5houses-sar` (2000 steps, 130×130 m, 20 reward objects) is a harder variant of `5houses-dynamic`, where on each house, one wall has the mass of its rubble objects set extremely high so that they are impossible to move by the agent but it looks the same as the movable rubble wall. In addition, smoke is present around each house and it degrades the sensing by simulating random lidar hits from smoke particles. The smoke has the same semantic class as obstacles. The

environment also contains chaotically moving objects (implemented as simple capsules) out of which 50% will always move out of the way of the agent and 50% will not care about the agent, emulating people with different behaviors moving in the area (but fundamentally this aims to simulate any chaotically moving objects in the scenario). The chaotic walkers have their own semantic class.

A.3 Foraging (Well Alternation)

All three foraging worlds use the same well scheduler and exactly one well is armed at any time. A well dispenses when the agent stays within 3 m of it for 10 consecutive steps. It then spawns 3 reward objects 1–1.7 m from the well, and the next well is armed after all of them are collected. Well priming then alternate between random and home. During random priming, a non-home well is drawn uniformly, excluding the previous random well, and it is cued. The cue is an 8-sector egocentric signal, which for each angular bin (45 degrees) has a value ranging from 0-1 which decreases with the angle between the bin normal and the direction to the cue, and with distance to the cue. Its range covers the whole arena and it provides direction and distance information for the armed random well, but provides no information about home wells (all values are 0 during return to home). Every well looks the same (however it has a different semantic class than other objects) regardless of whether it is a home or random well, and regardless of whether it is primed or not. After a random well is activated and the reward is collected, the home well is armed, and if the agent reaches it and collects the reward, a new random well is armed and so forth, with no maximum number of cycles. Episodes start with the agent placed next to the home well and the priming of a random well.

`ratmaze-foraging` (2000 steps, 42×42 m, ∞ reward objects) is a foraging (well-alternation) scenario where we replicate simplified experimental conditions seen in the rat study described in Widloski & Foster (2022). In this scenario, the agent is placed within a square box with 9 well objects and randomly placed walls creating various topologies. The generation logic makes sure that in each scenario, any 2 wells are co-reachable. To make the scenario compatible with our 2DSDS sensing mode and reasonable RL episode lengths, we make a few changes from the original animal experiment. Specifically, the inner walls in our scenario are not transparent, the outer walls are not textured (which the authors mainly use to give the rats a sense of direction, which we give to the agent directly through the compass observations), and the wells are not inside the ground but are standing objects, with a different semantic class than obstacles and reward objects. Additionally, we remove the delay between well activations, and the agent can immediately move to the next well after collecting the reward from a previous one.

`forest-foraging` (4000 steps, 130×130 m (containing wells and landmarks, otherwise infinite), ∞ reward objects) is a foraging scenario unique in that it does not contain a wall on its edges and is thus unbounded. The environment contains uniformly scattered tree objects, and if the agent moves outside of the area with the wells, it will continue to see trees spawned around it even if it were to move infinitely far from start (and, as shown in Figure 9, PPO often does leave the main area). Inside the main area, we randomly place 8 wells, and also 3 landmark objects for easier navigation, which have the same class as any other obstacle, but have unique shapes (a big cylinder, a rock formation composed form multiple boxes, and a spiky object simulating a bush or a jagged rock).

`4arm-submazes-foraging` (2000 steps, 200×200 m, ∞ reward objects) uses the same topology and scale as `4arm-submazes`, but changes the task. Instead of spawning 2 reward objects per room, this scenario spawns one well per every room (aside for the central one), for a total of 18 wells.

B Additional Metrics and RL Training Curves

Training curves for PPO and Dreamer in the 8 presented scenarios are shown in Figure 8.

C Extra Trajectory Visualizations

Visualizations of the eval trajectories from the most informative runs are shown in Figure 11, Figure 9 and Figure 10. These show some of the runs recorded as the eval runs in the experimental section.

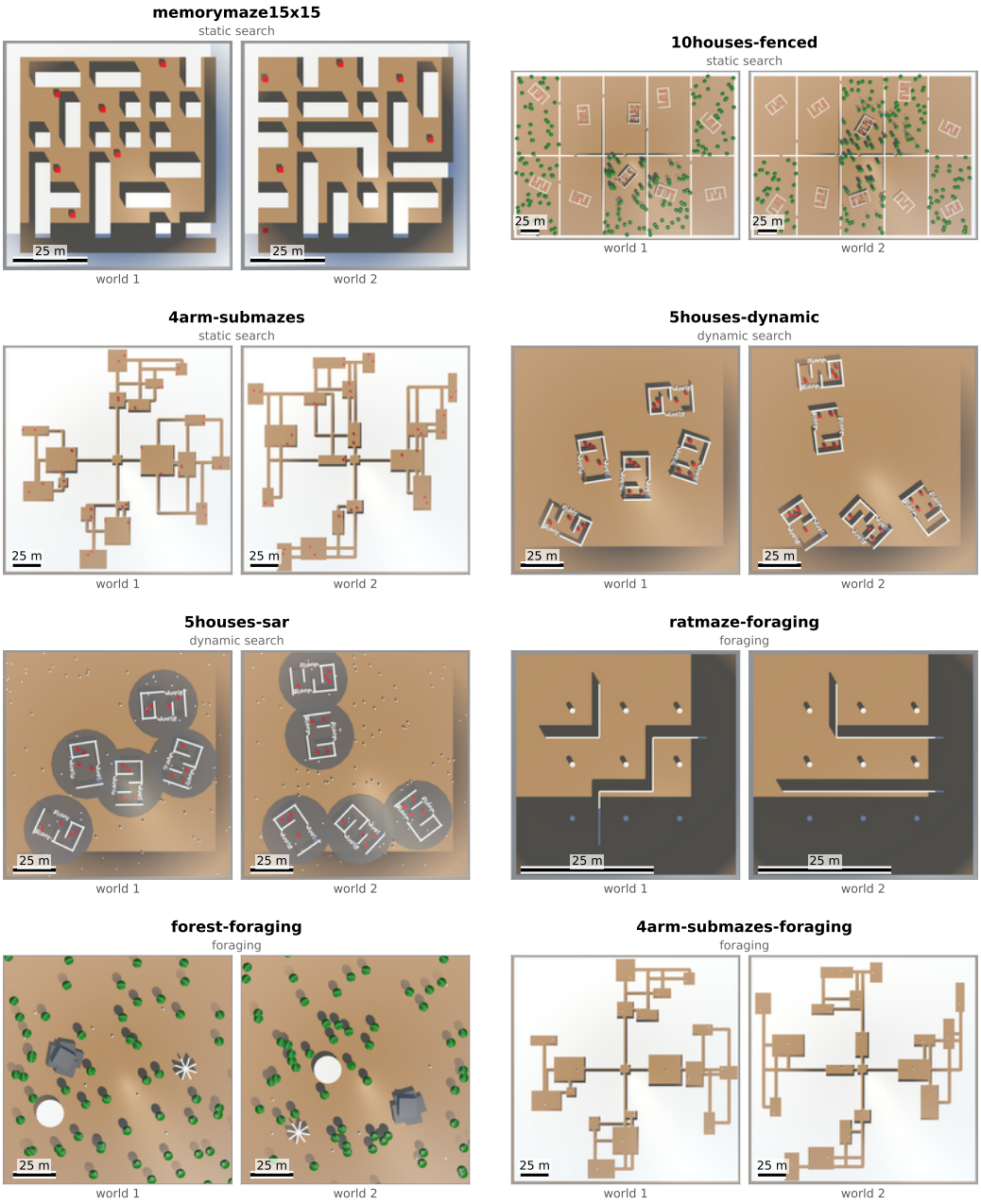


Figure 6: A visualization of 2 random seed for each of the 8 worlds used in the comparisons in subsection 4.1

D FRONTIER-EXPLORER IMPLEMENTATION DETAILS

The frontier explorer is implemented as a ROS2 node which runs in lockstep with the simulator (meaning that the simulator waits for the node's output target velocity, to be fair since RL and the Human also have theoretically unbounded time to compute the next step). The node builds a classic occupancy grid using the same 2DSDS sensing data as if it was a 2D LiDAR, however without marking hits of reward objects as obstacles in the map (the semantic class label for them is supplied to the node), and using the pose observations that RL policies get as well in the multi-object search scenarios. The occupancy grid has a resolution of 0.5 m per cell. Frontiers (Yamauchi, 1997) are

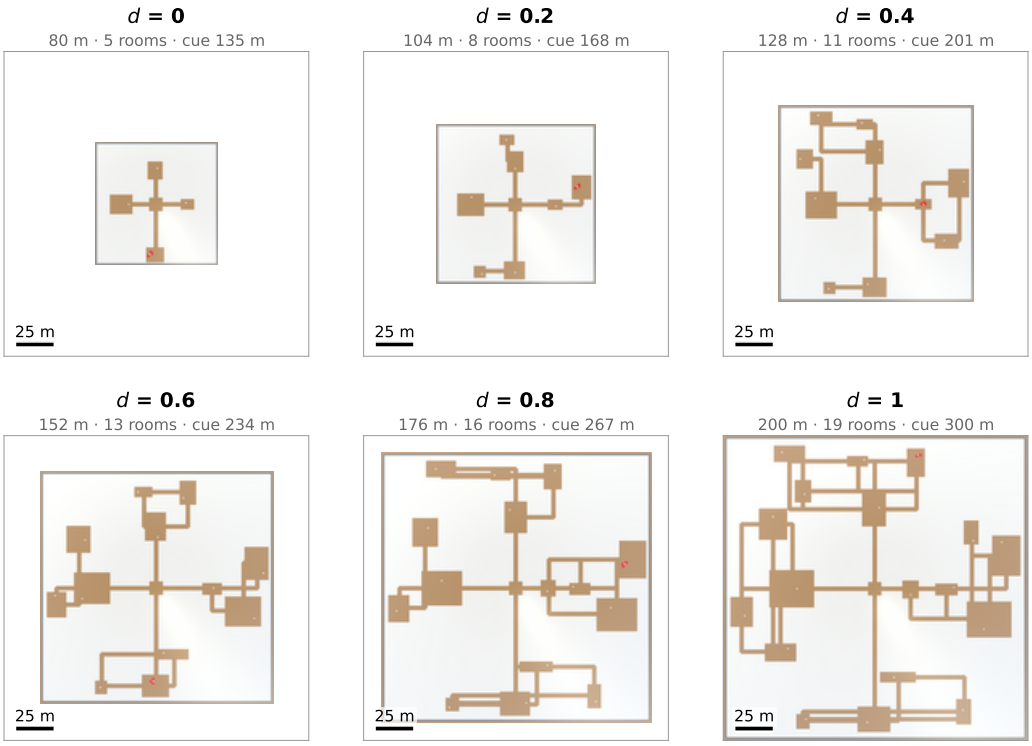


Figure 7: A visualization of adaptive difficulty scaling of world complexity on the example of the `4arm-submazes` world in the experiments described in subsection 4.2

computed as free cells adjacent to unknown cells, grouped into clusters and clusters under 5 cells are discarded. When the node selects a goal (either a frontier or a reward object), it plans paths using A*, with an inflation radius of 0.4 m from obstacles and plans only through free space. These paths are then followed using a standard carrot-on-a-stick path-following approach which calculates the output target velocities. The node alternates between 2 modes — frontier exploration and reward-object collection. Frontier-exploration is the default mode and it repeatedly detects frontier clusters, selects the shortest one (based on a Dijstra flood-fill from the agent's position), selects that as the next navigation goal and repeats this process. If a reward object is detected, the agent switches to the reward-collection mode. In this mode, the agent simply tries to navigate to the object's detected position and it returns to frontier exploration upon reaching the reward object position or after 10 steps of not seeing the object. The node also finishes the episode sooner if it detects no new reachable frontiers (e.g. if it clears the whole environment).

E Simulation Environment Details

Here, we describe the simulator implementation in more detail. An overview diagram of the simulator is shown in Figure 12. We also show two of the more realistic 3D worlds which are available in *ForagerSim*, but were not used in the experiments since their scale is above what current RL baselines are able to cover, and since we wanted to focus on the problems of topology, dynamics and scale while isolating away perception complexity of high-dimensional RGB observations. These worlds are shown in Figure 13 and show a recreation of the Hula valley, which is used in many publications documenting the scalability of the navigation capabilities of bats (e.g. Goldshtein et al. (2024)), and an example of a suburb environment with more realistic textures and models than the simple ones we use in the main experiments. We designed these as a next step to help the development of future methods, once they are able to perform the search and well-alternation tasks in large and dynamic 2D

World	Method	Score	OBJ	Coverage [10^2 m^2]
memorymaze15x15	PPO	$309_{\pm 39}$	$5.5_{\pm 0.7}$	$36.4_{\pm 2.7}$
	Dreamer	$320_{\pm 26}$	$5.7_{\pm 0.5}$	$35.9_{\pm 2.2}$
	Frontier-Explorer	$\mathbf{339}_{\pm 1}$	$6.0_{\pm 0.0}$	$39.2_{\pm 0.7}$
	Human	$\mathbf{339}_{\pm 1}$	$6.0_{\pm 0.0}$	$38.9_{\pm 0.7}$
10houses-fenced	PPO	$366_{\pm 303}$	$3.7_{\pm 3.6}$	$190.0_{\pm 159.8}$
	Dreamer	$2292_{\pm 481}$	$37.9_{\pm 8.4}$	$401.0_{\pm 92.6}$
	Frontier-Explorer	$1947_{\pm 389}$	$30.2_{\pm 8.5}$	$436.9_{\pm 40.2}$
	Human	$\mathbf{3452}_{\pm 134}$	$58.8_{\pm 2.7}$	$512.6_{\pm 10.7}$
4arm-submazes	PPO	$504_{\pm 210}$	$9.3_{\pm 4.3}$	$47.8_{\pm 36.4}$
	Dreamer	$777_{\pm 160}$	$14.9_{\pm 3.1}$	$43.3_{\pm 8.9}$
	Frontier-Explorer	$1868_{\pm 44}$	$35.4_{\pm 0.9}$	$98.5_{\pm 1.5}$
	Human	$\mathbf{1895}_{\pm 3}$	$36.0_{\pm 0.0}$	$99.5_{\pm 3.5}$
5houses-dynamic	PPO	$377_{\pm 282}$	$5.1_{\pm 5.8}$	$132.6_{\pm 30.9}$
	Dreamer	$984_{\pm 120}$	$16.9_{\pm 2.4}$	$152.9_{\pm 5.7}$
	Frontier-Explorer	$152_{\pm 0}$	$0.0_{\pm 0.0}$	$151.6_{\pm 0.2}$
	Human	$\mathbf{1114}_{\pm 100}$	$19.2_{\pm 1.8}$	$159.4_{\pm 10.1}$
5houses-sar	PPO	$142_{\pm 45}$	$0.2_{\pm 0.8}$	$133.2_{\pm 19.7}$
	Dreamer	$864_{\pm 196}$	$14.8_{\pm 3.7}$	$135.0_{\pm 20.9}$
	Frontier-Explorer	$118_{\pm 49}$	$0.0_{\pm 0.0}$	$118.0_{\pm 49.0}$
	Human	$\mathbf{957}_{\pm 99}$	$16.6_{\pm 1.9}$	$132.2_{\pm 11.6}$
ratmaze-foraging	PPO	$470_{\pm 525}$	$9.3_{\pm 10.4}$	$12.9_{\pm 6.2}$
	Dreamer	$2883_{\pm 256}$	$57.7_{\pm 5.1}$	$17.6_{\pm 0.0}$
	Human	$\mathbf{3827}_{\pm 515}$	$76.2_{\pm 10.3}$	$17.6_{\pm 0.0}$
forest-foraging	PPO	$335_{\pm 269}$	$4.8_{\pm 5.0}$	$99.5_{\pm 29.9}$
	Dreamer	$1619_{\pm 1370}$	$29.9_{\pm 27.2}$	$136.0_{\pm 20.6}$
	Human	$\mathbf{4220}_{\pm 639}$	$81.8_{\pm 12.9}$	$130.6_{\pm 5.9}$
4arm-submazes-foraging	PPO	$94_{\pm 79}$	$1.0_{\pm 1.5}$	$46.0_{\pm 45.6}$
	Dreamer	$272_{\pm 131}$	$4.5_{\pm 2.5}$	$49.2_{\pm 19.5}$
	Human	$\mathbf{1133}_{\pm 119}$	$21.6_{\pm 2.5}$	$55.3_{\pm 5.7}$

Table 3: Additional metrics for the evaluation described in subsection 4.1 and in Table 2. OBJ signifies number of collected reward objects, coverage is the area explored by the agent's sensors in the order of 10^2, since this is equal to the contribution of the coverage to the overall score (since we give a reward of 0.01 for each covered square meter).

scenarios in *ForagerSim*, to transition to fundamentally similar but 3D worlds and eventually into real-world applications.

E.1 Interface and Simulation Control

The world generation, dynamics and sensing and action simulation of *ForagerSim* are implemented in C# inside the Unity Game Engine, with a general-purpose python TCP connector interface which is then routed into both ROS2 and Gym, which map the general messages into either ROS2 messages/services or Gym observations/actions. The way of passing data between objects in Unity and the python interface is highly inspired by the ROS architecture of messages flowing in topics, with nodes able to publish or subscribe to any topic. This allows for high modularity (e.g. designing a new sensor in Unity requires only specifying a publisher object inside Unity and publishing the data on it, or designing a new actuator requires a subscriber in Unity mapping the message data into physics and writing the publishing of the action message in python on a specified topic), but we did not want to make Unity entagled with ROS2. Messages are defined in C# inside Unity and the python

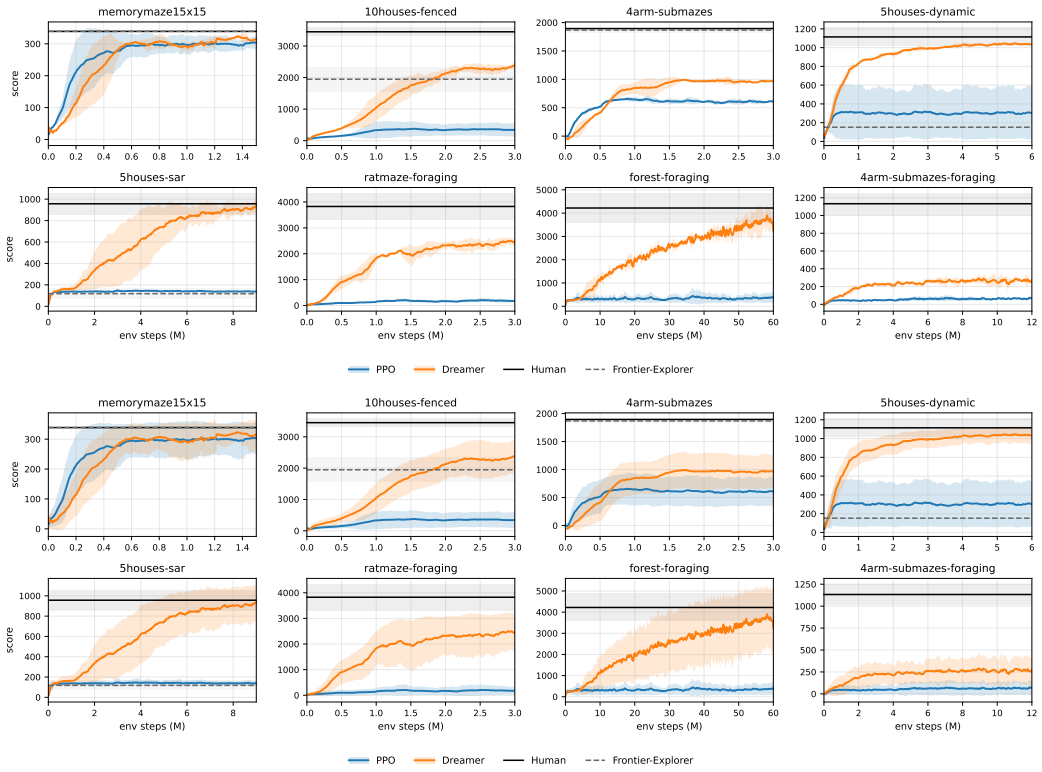


Figure 8: Full training curves of for each of the presented worlds. In all figures, the line represents mean return per episode averaged across training seeds. Top 2 rows show the standard deviation of the mean returns between individual training seeds. Bottom 2 rows show the deviation across *episodes* averaged over training seeds, which shows that the baseline performance varies drastically between individual generated worlds within any scenario, and corresponds to the high eval-world score deviation in the evaluations in subsection 4.1

package contains tooling to automatically generate their corresponding python classes (again, free of ROS2).

The simulation is fully controlled and the physics and all dynamics are stepped by the python connector, nothing is simulated unless the user (a python application using the connector) steps the simulation. On each step, the python connector sends all input messages (usually agent actions or world reset requests) into Unity through TCP, which are routed to subscribers in Unity objects. Afterwards, one step of the simulation (including physics) is performed, and then messages from all publishers within Unity are collected and sent back to python. For training, the simulation runs in headless mode with no screen output and the python interface allows running multiple Unity processes in parallel on different ports, which is how we implement multiple environments in our RL training. When evaluating human performance, we run in normal mode, with the Human UI visualized in Figure 2 dislayed on screen and the process accepting WASD/Space inputs.

E.2 Modular World Generation and World Scalability

To enable easy extension with new generation mechanisms and to reduce the headache caused by designing world generation order, *ForagerSim*'s world generation architecture is designed as a set of Generator modules, each of which can provide or require some type of data. The order of the generation determined based on these requirements upon world generation, which allows easily adding new generation mechanics (e.g. if we wanted to add generation of realistic roads, we would create a RoadGenerator module and make it require Height data and provide VegetationClearance data which is then subscribed by a VegetationGenerator).

The generation is then activated by agents — as an agent moves through the world, it *requests* world-data in chunks surrounding the agent as the agent moves through the world, and unloaded when the agent leaves the area, similar to Minecraft. This triggers individual generators by walking the generation dependency graph. However, since we want to be able to generate not just chunk-sized objects, we also use the concept of 'structures', which are oriented rectangular objects in the world, and the generation of their contents is triggered when loading of a chunk overlapping them is requested. While the simulator theoretically allows generating structures infinitely, in the presented experiments we use bounded worlds (and in the case of `forest-foraging`, we purposefully do not want the landmarks or wells to be outside of the 'content area'), and thus the high-level structures are placed once on world generation. The high-level structures (e.g. cities/mazes) can then spawn lower-level structures (houses, rooms etc.) as the agent gets near them. Grid-based data does not require any structures to exist, but can be influenced by structures, for example vegetation data can be influenced by houses. The user then can define worlds as a set of a high-level structures they want within a world, (e.g. 5 houses, 1 maze etc.) and then specify other parameters which guide the lower levels of the generation (e.g. specifying the number of rooms within a maze, types of prefabs allowed in a city, or allowing reward spawning only within rooms labeled 'reward-room' or spawning wells instead of reward objects).

F Hyperparameters for Training

Both RL baselines are trained on the same Gym environment with 4 parallel simulator instances and identical observations and action space. For parameters, PPO uses the Stable-Baselines3 v2.7.1 defaults without modification (`MultiInputPolicy`: flattened and concatenated observations, separate 2×64 tanh MLP actor and critic), trained on CPU only (16 threads per run). Table 4 lists the DreamerV3 hyperparameters that differ from the defaults of the official JAX implementation. Everything not listed is the default. Since the observations are vectors, only the MLP branches of the encoder and decoder are used.

Parameter	**Value (default)**
Model size	`size12m`: RSSM deter 2048, hidden 256, 16 classes, MLP units 256 (`size200m`)
Batch size	8 (16)
Sequence length	48 (64)
Replay context	16 (1)
Replay buffer	10^6 steps (5×10^6)
Parallel envs	4 (16)
Compute dtype	float32 (bfloat16)
Memory variants (subsection 4.2), relative to the above	
bl256	Sequence length 256
consec4	4 consecutive 48-step chunks with carried recurrent state
TBTT	Sequential replay with state carried across episodes, replay context 0, following Pašukonis et al. (2023)

Table 4: DreamerV3 hyperparameters that differ from the official implementation's defaults (defaults in parentheses). PPO uses the Stable-Baselines3 defaults.

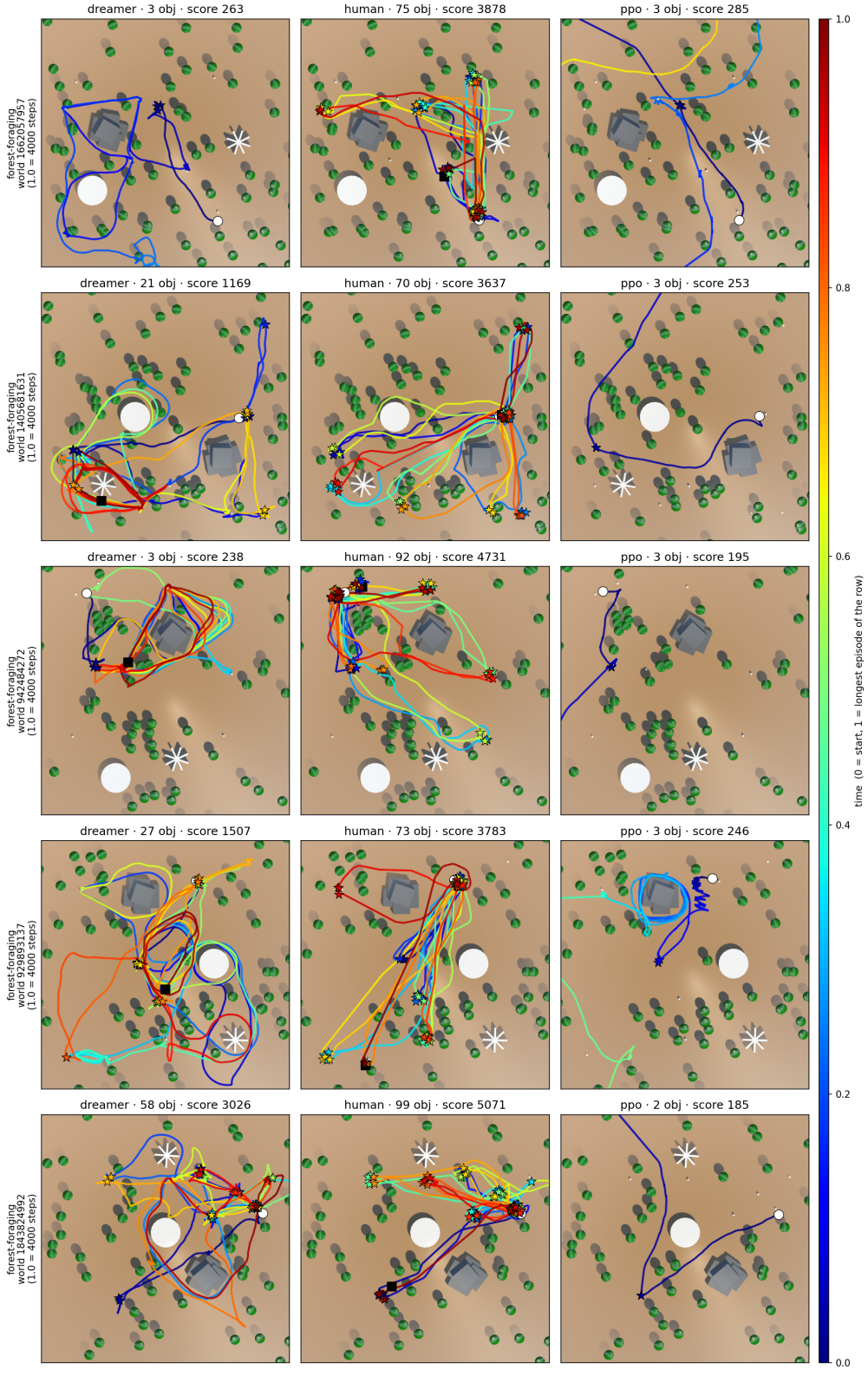


Figure 9: Visualization of eval trajectories on the `forest-foraging` unbounded well-alternation scenario. Dreamer walks outside of the main area into the infinite forest in one world, PPO in all of them.

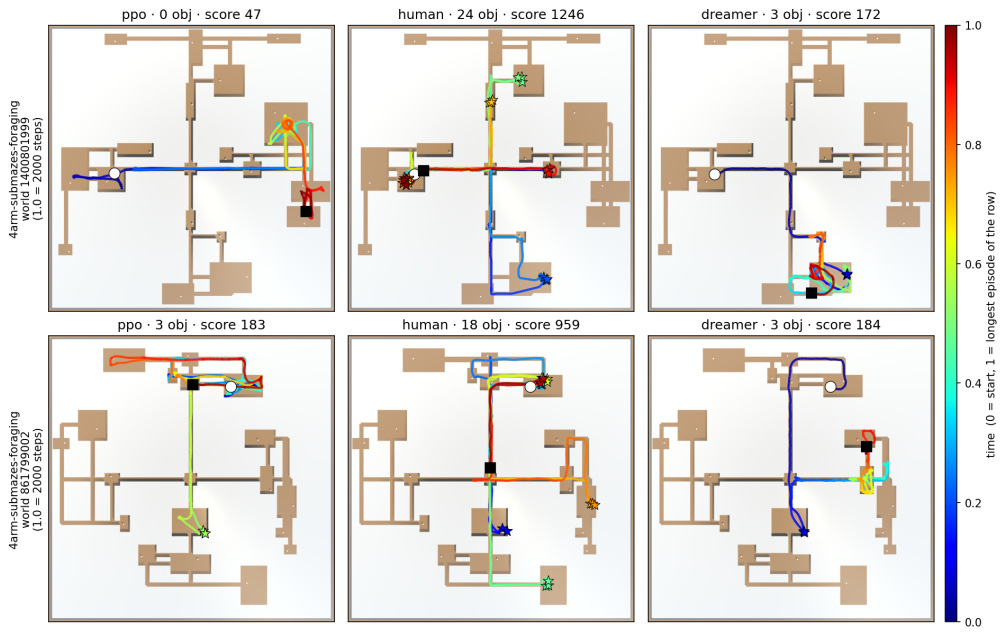


Figure 10: Visualization of eval trajectories on the `4arm-submaze-foraging` well-alternation scenario. The human player can perform 3 or more random-home alternations while both PPO and Dreamer struggle with learning to return home after finding the cued random well.

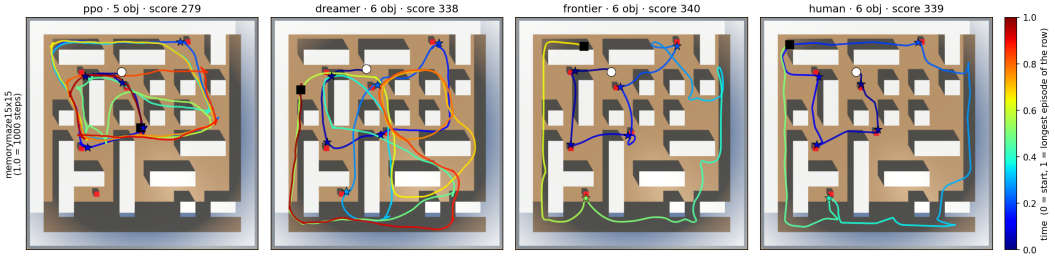


Figure 11: Trajectory visualizations for the `memorymaze15x15` world, showing that it can be easily fully covered in the multi-object search task by even a model-free, memoryless PPO agent. This suggests that with enough time (since the human and the frontier explorer were able to cover it sometimes in 50% of the episode steps), this environment is not challenging in terms of map-based reasoning or memory (although the original *ordered*-search task is, as the authors present, shown to be a good benchmark of memory).

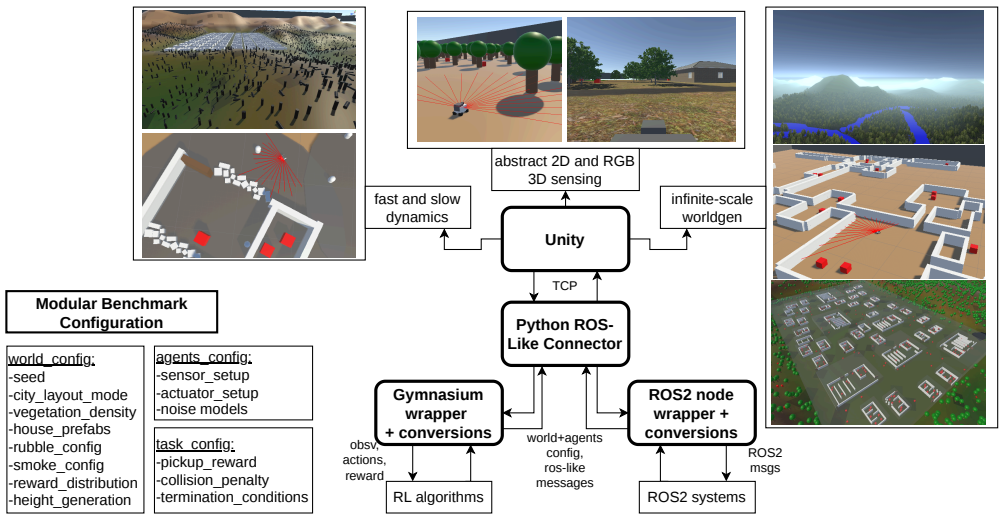


Figure 12: *ForagerSim* environment architecture overview.

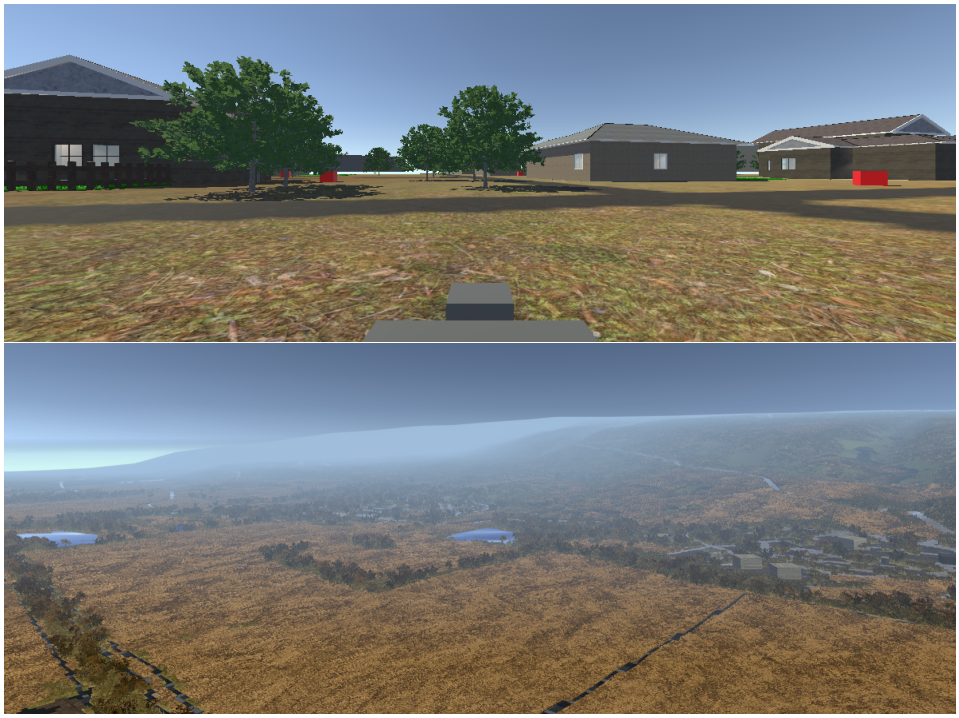


Figure 13: Examples of the 3D environments available in *ForagerSim* in the RGB/LiDAR sensing modes, Top: a 3D variant of search-and-rescue missions in suburb-scale environments with interactible, randomized houses (with ability to swap abstract red boxes as reward objects for human models), Bottom: a recreation of the Hula valley on which the animal experiments documenting the scalability of biological cognitive maps of bats have been recorded Goldshtein et al. (2024)