PacMouse: Mice solve hidden abstract graph traversal problems in freely-moving virtual reality

Matthew Rosenberg
Princeton Neuroscience Institute
`mhr3@princeton.edu`

Zac Levy
Princeton Electrical and Computer Engineering
`zac@princeton.edu`

David Tank
Princeton Neuroscience Institute
`dwtank@princeton.edu`

Abstract

Biological agents such as mice solve navigation problems in uncertain and partially observable environments. We introduce *PacMouse*, a rodent behavioral paradigm that elicits exploration of latent graphs while minimizing confounding internal and external cues that could trivialize state estimation. The task is implemented through a freely-moving virtual reality mouse experiment paradigm, in which all decisions occur at a single physical junction with motorized gates controlling access to four radial arms. Proximity sensors in the radial arms trigger virtual transitions between adjacent nodes in a hidden latent graph. This design precludes reliance on path integration or landmark signals, enabling direct comparison between animal behavior and simulated agents. We show that mice frequently outperform a range of baseline policies. The framework provides open-source tools for linking mouse behavioral data to standard RL implementations, establishing a common task in which to benchmark head-to-head biological and artificial decision making.

1 Introduction

Despite recent advances in machine learning, thinking biological organisms nevertheless consume far less energy than artificial systems, while demonstrating the capacity to solve complex cognitive problems like navigation in uncertain environments [1–4]. Estimates for the energy consumed by large language models and even consumer GPUs are orders of magnitude higher than the total energy required to sustain a mouse [5–8]. Small animals are frequently forced to navigate in complex environments that lack global beacon signals or explicit landmarks, like tunnels or paths through underbrush. Navigation in these settings requires either an accurate path integration system ([9])—often termed simultaneous localization and mapping (SLAM) in machine-learning contexts [10–13]—or an ability to adaptively combine current sensory observations with memories of prior observations and actions. While engineering improvements to SLAM and neuroscience investigations into brain mechanisms underlying path integration remain areas of active research, permitting agents to simply assign actions to provided xy positions narrows the scope of applicable cognitive problems these systems can address.

While theoretical and practical advances have been made in reinforcement learning (RL) using *fully observed* Markov Decision Processes (MDPs), where the agent's policy is informed by a given state, comparatively less work has been devoted to the *partially observed* (POMDP) scenario in which an

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Data on the Brain & Mind.

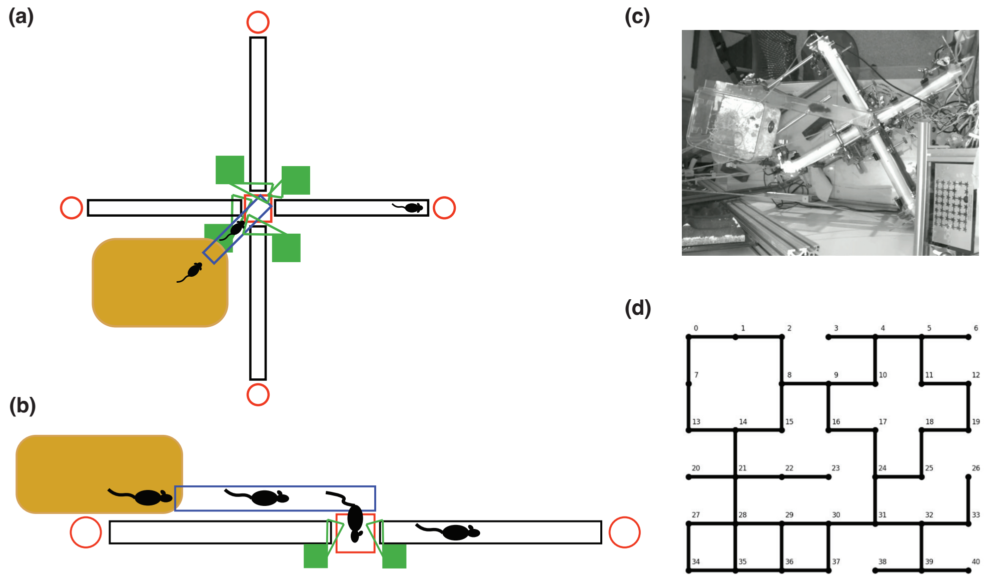

Figure 1: PacMouse latent maze experiment paradigm. (a) Schematic top view with the home cage (brown) connected to the central junction (red square). (b) Schematic side view showing the clear tubing (blue) and gates (green) controlling access to the maze arms (black). (c) Video frame of the real apparatus showing a mouse entering the experiment from the home cage. (d) Example latent graph representation of the maze, with nodes and edges corresponding to discrete states and possible transitions.

agent's true state in the environment is not directly accessible to the agent. Importantly, observations may be aliased in that multiple states in the environment may emit the same observation to the agent. Partially-observed or state-aliased environments supply to the agent merely an observation rather than a true state, presenting a challenging but much more general framework to consider cognition more broadly.

Simulation work in reinforcement learning and robotics communities frequently use maze environments with similarities to rodent neuroscience experiments. However, careful comparison between mice and simulated agents is complicated by several factors. Modeling work often assumes that the simulated agents represent the environment in ways that are difficult to verify in animal experiments or minimally supported by data. For instance, modern reinforcement learning algorithms are often trained on video frames [14, 15] and yet we know that mice have markedly different visual systems than humans[16], tending to forage most at night when photons are least available[17]. Even when mice act under complete visual darkness and care is taken to minimize local maze cues, prior work[18] and unpublished preliminary data from real (non-virtual) physical maze navigation experiments suggest that mice are seldom confused about their xy position in space, likely due to path integration mechanisms or uncontrolled environment cues like airflow or ventilation sounds.

Here, we introduce PacMouse, a novel rodent behavioral task, that elicits non-trivial exploration behaviors in abstract latent graphs while removing troublesome internal (e.g. path integration) and external sensory cues (e.g. landmarks) that might otherwise de-alias environment states, trivializing observations into veridical states. This is accomplished by allowing mice to engage with a novel form of freely-moving virtual reality in which the animal traverses an abstract hidden latent graph, composed of nodes and edges embedded in a square lattice. To eliminate the confounds and modeling complications associated with path integration, the task is designed such that all decisions are made at a single physical junction with motorized gates controlling access to four attached arms. Left, right, forward, and backward virtual actions in the latent graph are preserved relative to the mouse's direction of travel upon entering the physical junction choice point. We find that mouse performance often exceeds that of a wide range of policies, including those that approximate the task via random actions or via actions biases matching that of observed mouse data. This work establishes a complete end-to-end biologically plausible benchmarking paradigm, connecting mouse behavioral data and tightly aligned simulations, via open-source behavioral data analysis and the potential for integration with commonly used deep reinforcement learning implementations.

2 Methods

2.1 Overview

To completely eliminate the confounding variables of global room cues and internal path integration signals, we developed a virtual reality latent maze task in which the direction the mouse selects at the central junction in the physical 4-arm radial maze determines which edge is traversed in the latent virtual graph. A freely moving mouse's egocentric actions (forward, back, left, and right) are recorded via the sequence of nose insertions into the ports at the end of the arms. Whether the entrance to a given arm is blocked by a gate or not signals the presence of an edge between one of the four possible adjacent neighbor nodes in the latent virtual graph maze. The mouse's action consists of selecting one of the open arms or reversing and traveling back along the same arm to press the prior port again. Upon pressing the port, the animal's current position in the latent maze updates to the selected node and the physical gates change configuration to reflect the available edges at the new latent node now occupied by the animal in the virtual maze.

Path integration signals are unavailable to the mouse because it is forced to return to the same central junction to make each new decision, whereas in typical rodent navigation each subsequent decision is made at a distinct xy coordinate in real physical space. Thus, the mouse may traverse a hidden virtual graph maze of arbitrary xy dimensions via real actions confined to the 4-arms and central junction. Furthermore, the task is partially-observed in that the virtual environment provides observations that reflect the occupied state rather than the state itself.

2.2 Mouse behavioral data collection

The latent maze paradigm is fundamentally a virtual reality system that converts electrical signals from mouse proximity sensors in four nose-poke ports into edge traversals between states (equivalently, vertices or "nodes") in an abstract virtual graph governed by a python script running on a connected microcontroller (Raspberry Pi 4). Four gates controlled by stepper motors determine whether entry into specific radial arms is possible from a central choice junction. Which gates are open at a given time is governed by the current state occupied in the latent graph, the animal's direction of travel in the virtual graph (allocentric heading), and the most recently poked sensor in the physical 4-arm apparatus (real heading). The mouse receives a small fluid reward upon poking into a port if that action brings it to a novel state in the latent maze. Similar to the classic arcade game PacMan, revisiting nodes in the virtual graph does not yield subsequent further reward until enough of the nodes have been visited (0-2 remaining unvisited). Upon reaching a sufficient proportion of the states, the environment replenishes rewards at all states.

Proximity sensors at the ends of four tubes, joined at a central junction, control virtual transitions along edges between vertices ("nodes" or "states", equivalently) of graphs embedded in a square lattice (fig 1a). Crucially, the graph determining the adjacency of vertices is never directly observed by the mouse. The animal is only permitted to observe the adjacent edges that connect it to proximal vertices via the status of motorized doors that swing to open or close access to a given arm from the central junction. The presence of an edge leading from the animal's current state in the latent graph to adjacent states is signaled only by whether the arms connected in the forward, left, or right egocentric directions are open from the central junction. The sensor last poked by the mouse defines the direction of travel in the latent maze. Mice are always permitted to move in the reverse direction by re-poking the sensor that was last poked, after a 45–90 second timeout that prevents rapid jitter between adjacent states due to rapid repeated pokes incurred as the animal attempts to extract as much water as possible from a given port.

Importantly, egocentric (F, B, L, R) actions are equivalent between the physical 4-arm apparatus and the hidden latent maze graph. For instance, running along a straight corridor in the virtual maze is achieved by alternating pokes at sensors at opposite sides of the central junction, such that forward actions are performed in both real apparatus and the latent maze. Likewise, clockwise circling among 4 adjacent nodes in the virtual maze results from the animal visiting the physical ports in a counterclockwise fashion; a series of right turns are made in both the physical apparatus and the virtual graph. Refer to Supplementary Method Section A.1 for further details.

2.3 Latent maze task formalism

We cast the task as a POMDP $(S, A, \mathcal{T}, R, \Omega, O)$. States are nodes on a subgraph $G = (V, E)$, actions A are unit steps, and transitions are deterministic if the edge exists:

$$\mathcal{T}(s' \mid s, a) = \begin{cases} 1, & s' = s + a,\ \{s, s'\} \in E, \\ 0, & \text{otherwise.} \end{cases}$$

Rewards, governed by the PacMouse reward function $R(s)$, are non-zero only upon the first entry to a node, until reset when sufficiently few remain unvisited. See Supplementary Method Section A.3 for a more complete description.

2.3.1 Action representations.

The underlying action set A can be expressed in three different coordinate frames.

Allocentric-latent: Define

$$A_{\text{allo}} = \{\, \text{east} : (1, 0),\ \text{west} : (-1, 0),\ \text{north} : (0, 1),\ \text{south} : (0, -1) \,\}.$$

This frame is simply a relabeling of $\mathcal{D}$ into compass directions relative to the latent lattice.

Allocentric-real: This frame defines the actual (or would-be, for simulations) contingency between mouse nose-pokes into the physical experiment sensors and the virtual edge traversals in the latent graph E. Let n_t denote the number of states encountered by the agent up to time t. The physical action set A_{real} is then defined in terms of the allocentric-latent set A_{allo} by

$$A_{\text{real}} = \begin{cases} A_{\text{allo}}, & \text{if } n_t \bmod 2 = 0, \\ \text{Rot}_\pi(A_{\text{allo}}) = \left\{ \begin{array}{ll} \text{east:} & (-1, 0) \\ \text{west:} & (1, 0) \\ \text{north:} & (0, -1) \\ \text{south:} & (0, 1) \end{array} \right\}, & \text{else.} \end{cases}$$

where Rot_π denotes a rotation by $180°$ (i.e. π radians). Thus A_{real} coincides with A_{allo} on even-indexed states and is flipped by $180°$ on odd-indexed states. This rotation of the sensor-to-edge mapping, A_{real}, relative to the latent avatar's heading in the graph, A_{allo}, enables traversal of an arbitrary lattice graph while confining the instantiation of the physical mouse experiment footprint to a single central junction (with 1-4 accessible arms depending on the current location in E).

Egocentric: Define

$$A_{\text{ego}} = \{\text{forward}, \text{backward}, \text{left}, \text{right}\}.$$

For a transition $s = (x, y) \to s' = (x', y')$, let the allocentric displacement angle be

$$\theta_{\text{allo}} = \arctan 2(y' - y,\ x' - x).$$

Given the agent's current heading θ_t (initialized to north if no action has been taken), the relative angle is

$$\theta_{\text{ego}} = \big(\theta_{\text{allo}} - \theta_t\big) \bmod 2\pi.$$

Discretize θ_{ego} via

$$\{0 \mapsto \text{forward},\ \pi \mapsto \text{backward},\ \tfrac{\pi}{2} \mapsto \text{left},\ \tfrac{3\pi}{2} \mapsto \text{right}\},$$

yielding the corresponding element of A_{ego}.

Remark. The construction ensures A_{ego} is consistent between the latent avatar's first-person perspective and the choices presented to the mouse (or simulated agent) upon returning to the central junction after poking a distal arm sensor.

2.3.2 Simulated agent observation spaces

Mice must interact with the experiment via A_{real}, but how they represent the task internally is not known a priori. To remain agnostic about how the task might be represented, we explore simulated agents that differ in representation (as well as in the policies described below).

Simulated agent observations mirror the action coordinate frames defined above. At each step, the agent receives $o_t \in \Omega$, which may be allocentric-real, allocentric-latent, or egocentric, depending on the model setup. The observation model is deterministic but representation-dependent:

$$\forall \{s_t, s_t + a_i\} \in E : \quad o_t = O(o_t \mid s_t, a_t) = f_{\text{rep}}(a_i, s_t).$$

where f_{rep} is the relabeling function that maps the latent direction set $\mathcal{D}$ into the chosen coordinate frame (allocentric-latent, allocentric-real, or egocentric), following the approach described above for actions. Thus, O encodes the same underlying graph adjacency information, but expressed in the desired representation.

2.4 Simulations

2.4.1 Performance assessment and benchmarking

Since revisiting states does not typically yield further reward, the number of rewards obtained in a given session provides a sensible normative metric by which to assess mouse and simulation policy/representation quality. Because mouse experiments yielded variable numbers of actions per session, we normalize the reward count by the number of actions taken to yield the average reward per action,

$$\bar{r} = \frac{1}{T} \sum_{t=1}^{T} r_t,$$

where T is the total number of actions in the session and r_t the reward obtained at time t. By construction, $\bar{r} \in [0, 1]$. Under our PacMouse reward function R, the upper bound $\bar{r} = 1$ is attainable given an optimal policy and the latent graph G admits a Hamiltonian path, i.e. a traversal that visits every state exactly once. At the other extreme, the worst-case policy takes an initial action followed by an indefinite sequence of immediate reversals ($a_{2:T} = \text{reverse}$), yielding $\bar{r} \to 0$ as $T \to \infty$.

2.4.2 Yoking simulations to mouse data

Mouse experiment data were highly heterogeneous due to individual differences across mice, within-mouse changes in behavior over the course of training (i.e., learning), and exploratory experimenter-induced manipulations of the latent graph G. To enable direct comparison, all simulation variants were run in distinct episodes aligned ("yoked") to the key features from observed mouse data: the latent graph G a given mouse explored that session; the mouse's initial state s_0 (typically s_T from the preceding mouse session); the PacMouse reward function R initialized as S; the latent avatar's initial heading $\theta_{t=0}$; and—most critically—the observed number of mouse actions prior to experiment reset, which defined each agent's action budget T for that episode.

Thus, each mouse experiment session was directly compared to every simulation policy and representation explored. To reduce variance in simulation performance estimates, each combination of agent variant and yoked-simulation were repeated 200 times, and the results were averaged to obtain one simulated reward rate for each combination.

2.4.3 Simulation policies

An agent's policy specifies how the agent selects actions on the basis of its observations. Formally, a (possibly stochastic) policy is a conditional distribution

$$\pi(a \mid o_t) = \Pr(a_t = a \mid o_t),$$

where $o_t \in \Omega$ is the observation at time t and $a \in A$ an available action. To avoid making unwarranted assumptions about the representation mice *actually use* or simulations *might best* exploit, we run separate simulations for each policy type under allocentric-latent, allocentric-real, and egocentric A

and Ω reference frames. To partially disentangle the separate influences of the latent graph G and the reward function R, we also define an additional representation in which the agent is provided direct access to its current latent state s_t.

Performance benchmarking results are provided for the following families of policies.

Static policies explored

Random. Here the policy is defined as

$$\pi(a_t \mid o_t) = \pi(a_t) = \frac{1}{|S_{\text{adj}}|},$$

where $|S_{\text{adj}}|$ indicates the number of adjacent states available from the current node. To reflect the natural bias of mice to avoid reversals, as well as the sensor timeout period that further discouraged backward actions, we define a variant of this random policy in which reversals are only permitted when no other actions are available, i.e. when the agent has reached a dead end.

Action biased. These policies convert the observed tallies of mouse actions in a given experiment into probabilities of selecting candidate actions. When a given action is unavailable, that action is removed from the set of possible actions and the probabilities of the remaining actions are renormalized to sum to 1.

Let $c(a)$ denote the observed tally of action $a \in A_{\text{avail}}(s_t) \subseteq A$ in a given mouse experiment. The action-biased policy is then

$$\pi(a \mid o_t) = \frac{c(a)}{\sum_{a' \in A_{\text{avail}}(s_t)} c(a')}.$$

To allow for the possibility that mice might develop decision biases that group together sequences of actions, we analogously define variants that operate over observed sequences of mouse data, i.e.

$$\pi(a_t, a_{t+1}, \ldots, a_L),$$

where L is the stipulated sequence length considered. Candidate sequences are truncated when none of the observed mouse action subsequences are supported by G or when the action budget T is reached.

Observation–action biased. This class uses observed mouse choice data to define conditional action probabilities. Let $c(a, o)$ denote the tally of times action $a \in A_{\text{avail}}(s_t) \subseteq A$ was taken following observation $o \in \Omega$ in a given mouse experiment.

The observation–action biased policy is then

$$\pi(a \mid o_t) = \frac{c(a \mid o_t)}{\sum_{a' \in A_{\text{avail}}(s_t)} c(a' \mid o_t)}.$$

Latent-state biased. This class defines transition probabilities based on observed state-occupancy transitions. Let $c(s, s')$ denote the tally of transitions from latent state s to s' observed in mouse experiment sessions. Define

$$\pi(s_{t+1} \mid s_t) = \frac{c(s_t, s_{t+1})}{\sum_{s' \in S_{\text{adj}}} c(s_t, s')}.$$

where S_{adj} denotes all the available latent states that the agent can reach from state s_t. Thus the empirical transition tallies are normalized to yield a valid distribution over next states.

3 Results

3.1 Mice obtained more rewards than baseline simulations overall

The PacMouse latent maze paradigm is motivated by the desire to directly compare mouse task performance to that of various simulations, varying both in how observations provided by the

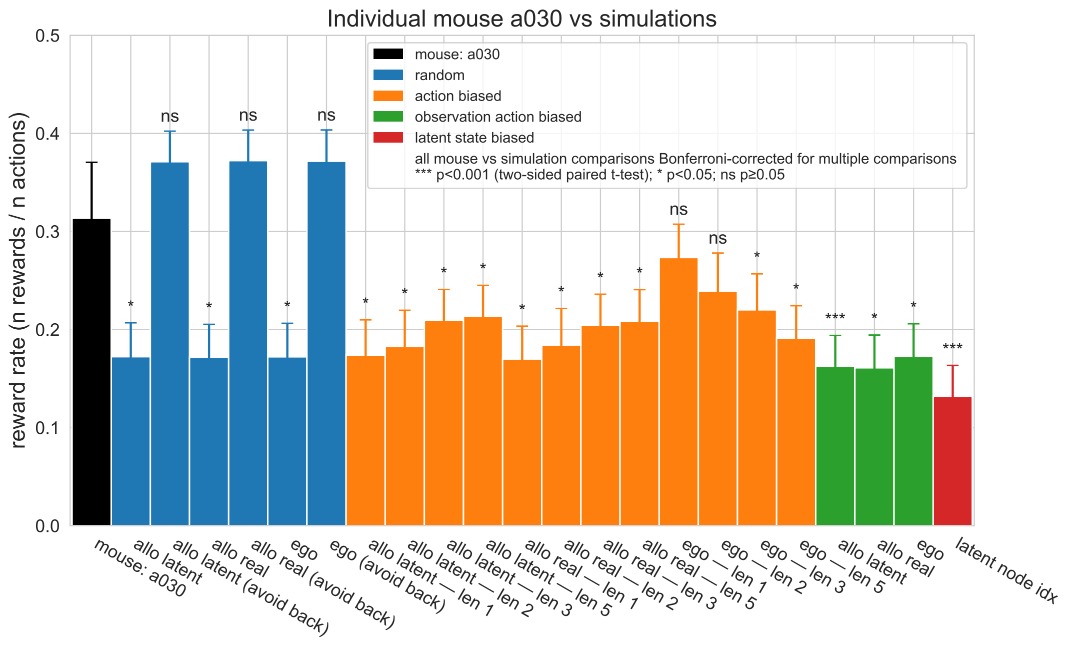

Figure 2: Performance of exemplar mouse vs yoked-baseline simulations. Asterisks indicate significance (paired two-sided t-test, Bonferroni-corrected).

environment are interpreted (representation Ω) and in how observations are mapped into actions (policy π). As an initial characterization of this dataset, the average reward $\bar{r}$ achieved by mice is compared to baseline policies of minimal algorithmic complexity.

We computed the average reward rate ($\bar{r}$) for each mouse and session. Simulation reward rates were averaged across models on the same basis, and the two sets of values were compared using paired-sample two-tailed t-tests. We find that mice obtain significantly more rewards than yoked simulations (two-sided paired t-test; $t(105) = -2.98$, $p = 3.63 \times 10^{-3}$ across 5 mice; means: 0.256 vs. 0.229; paired mean difference $= 0.0268 \pm 0.0090$ s.e.m.; Supplementary Fig. S1).

However, interpretation of statistical comparisons between mice and pooled averages of disparate simulations is complicated by the heterogeneity of average reward rates associated with different combinations of simulated agent policies and representations. Therefore, we eschewed averaging across simulation agent families in the more detailed benchmarked characterizations between mice and their yoked simulations described below.

3.2 Relative performance of mice and specific agent models

To establish an initial set of benchmarks, we compare single mice with the performance of each agent simulation type separately, averaging over all mouse experiment sessions and the experiment-yoked simulated episodes (Fig. 2 and Supplementary Fig. S2). The relative quality of the different families of simulations was largely consistent across mice. We observe that simply matching the decision biases of mice leads to low rates of reward. The poor performance of action biased and observation-action biased agents, relative to yoked mouse counterparts, suggest that the underlying mouse policy is more complicated than a decision function that simply prefers a given physical arm or action. Policies defined by multi-action sequence (length > 1) biases generally did not perform better than single action biases. Notably, agents with an egocentric representation performed slightly better than allocentric variants, potentially due to the advantage in this task associated with moving forward in the latent graph to minimize chances of returning to already visited states.

While inclusion of full-fledged POMDP agents is left to future work, we briefly consider a latent-state biased agent that is provided oracular access to its true position in the latent graph. This agent selects among candidate latent states with probabilities proportional to the yoked mouse's empirical latent state occupancy counts. We observe that this policy variant performs especially poorly, likely revealing additional challenges associated with the non-stationarity of the PacMouse reward function in this task, beyond the difficulties of state estimation alone.

Among these trivial baseline policies with fixed policies per episode, we find that forward-biased but otherwise randomly acting agents perform best, sometimes exceeding the reward rates of their mouse counterparts. The aptitude of this policy likely relates to the avoidance of wasted actions spent via returning to the immediately prior state s_{t-1} and reduced reentry into already visited areas via increased decision entropy.

The analyses presented so far collapse across the full training period, ignoring changes that unfolded as the mice became familiar with the task and as the experimenter gradually increased the difficulty of the latent graph. Despite between-session variability, mice frequently obtained rewards at a higher rate than even the best-performing forward-biased random agents (Supplementary Fig. S3)), consistent with the possibility that mice may be capable of non-trivial policies potentially reflecting knowledge of the experiment's underlying latent graph.

4 Discussion

The PacMouse latent maze paradigm leverages the natural aptitude of mice to explore complex unfamiliar confined spaces in order to expand the scope of cognitive tasks amenable to animal studies to include ambiguous environments, where state estimation is difficult. This paradigm facilitates direct comparisons between the performance of biological organisms and artificial agents by placing all task-relevant sensory features under direct experimental control. Our preliminary results demonstrate that mice achieve higher performance than simple bias-matched policies, supporting the use of this paradigm as a biologically principled framework for evaluating synthetic agent decision making relative to organic counterparts. We are currently developing more complex models to better capture mouse policies, as well as benchmarking common standardized deep reinforcement learning models implemented in stable-baselines3 [19] implementations via a custom Gym environment [20].

This initial benchmarking does not include comparison to explicit POMDP models, which is beyond the scope of the present work. To properly align the task scenario between what is experienced by mice and the simulations, the environment's transition function $\mathcal{T}$ should be learned from experience. Many POMDP agent algorithms take $\mathcal{T}$ as given rather than learned. Conducting simulations in which $\mathcal{T}$ is learned presents an exciting direction for future analysis work, with potential relevance to constraining theories of neural computation and improving the robustness and flexibility of artificial systems. Likewise, normative comparison of reward rates is only one way to assess the potential similarity of mouse and agent policies. Future work will explicitly assess the fit between observed mouse actions and those of simulated agents. Lastly, exciting avenues for future work might include comparison of differences in behavioral policy between identical graphs presented in the latent maze and a real physical maze with the same edge adjacency or combining the latent maze paradigm with simultaneous neural recordings. Open questions include whether place cell firing fields observed in physical arenas will similarly show selective firing for distinct latent states in the virtual graph.

References

[1] H. Kurniawati, D. Hsu, and W. S. Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Proceedings of RSS*, 2008.

[2] S. Swanborn and I. Malavolta. Energy efficiency in robotics software: A systematic literature review. In *ASEW*, 2020.

[3] N. N. Kyaw, A. Gamage, S. Theingi, M. H. Myint, M. M. Aye, and K. K. Soe. Energy-efficient path planning of reconfigurable robots in complex environments. *IEEE Transactions on Robotics*, 2022.

[4] Z. Liu, Y. Li, H. Wang, X. Zhao, Y. Qian, and Z. Wang. An open approach to energy-efficient autonomous mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[5] D. Patterson, J. Gonzalez, U. Hölzle, Q. Le, C. Liang, L. Miralles, M. Suleyman, and J. Dean. The carbon footprint of large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[6] Y. Wu et al. How hungry is ai? energy consumption of large language model inference. *arXiv preprint arXiv:2505.09598*, 2025.

[7] NVIDIA. Geforce rtx 4070 family graphics cards: Specifications and power. `https://www.nvidia.com/en-us/geforce/graphics-cards/40-series/rtx-4070-family/`, 2023. See *Thermal and Power Specs*; RTX 4070 Total Graphics Power listed as 200 W.

[8] NVIDIA. Nvidia system management interface (`nvidia-smi`)—user guide. `https://docs.nvidia.com/deploy/nvidia-smi/index.html`, n.d. Official documentation and usage reference.

[9] M. Muller and R. Wehner. Path integration in desert ants, Cataglyphis fortis. *Proceedings of the National Academy of Sciences*, 1988.

[10] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part i — the essential algorithms. *IEEE Robotics & Automation Magazine*, 2006.

[11] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii — state of the art. *IEEE Robotics & Automation Magazine*, 2006.

[12] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 2016.

[13] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 2015.

[14] J. Merel, D. Aldarondo, J. Marshall, Y. Tassa, G. Wayne, and R. Darshan. Deep neuroethology of a virtual rodent. *PLOS Computational Biology*, 2019.

[15] A. Banino, C. Barry, B. Uria, C. Blundell, T. Lillicrap, P. Mirowski, A. Pritzel, M. J. Chadwick, T. Degris, J. Modayil, et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 2018.

[16] F. J. Luongo, A. M. Laws, D. A. Haghverdian, Z. Miao, J. Hembrook-Short, C. Xue, et al. Mice and primates use distinct strategies for visual segmentation. *Nature Neuroscience*, 2023.

[17] Roberto Refinetti. Variability of diurnality in laboratory rodents. *Journal of Comparative Physiology A*, 192(7):701–714, 2006.

[18] M. Rosenberg, T. Zhang, P. Perona, and M. Meister. Mice in a labyrinth show rapid learning, sudden insight, and efficient exploration. *eLife*, 2021.

[19] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[20] M. Towers, J. K. Terry, J. Balis, et al. Gymnasium. Zenodo, 2023.

A Technical Appendices and Supplementary Material

A.1 Mouse experiment details

At the start of each new experiment session, at least one gate is opened permitting access to a port. Mice poke sensors to traverse the latent graph. To maximize engagement with the apparatus, the mouse is allowed to freely enter and exit the 4-arm experimental apparatus and return to their home cage whenever they wish. The latent maze is continually connected to a typical home cage containing all supplies aside from water, permitting the mouse to pass between the experiment apparatus and its home cage whenever it wishes. Entries and exits to the apparatus do not change the mouse's position in the latent graph and are not considered in any analyses below. For reference, mice are recorded via a parallel video recording.

Experiments are interrupted at least daily to weigh the animal and ensure that it receives sufficient minimum daily fluid. Thus, each experiment session typically lasts approximately 24 hours, running otherwise uninterrupted except for periods of apparatus cleaning. The volume of fluid dispensed per reward, typically around 0.02 mL per reward, is modified slightly between experiments. Early in training, we increased the reward to minimize the amount of "free" water that was necessary to provide outside of the experiment in the animal's home cage to maintain the animal's minimum weight and daily fluid requirements. As the animal's performance improved, enabling it to obtain more rewards, the volume of water per reward was decreased to make satiation less likely. All animal procedures were approved by the governing Institutional Animal Care and Use Committee.

A.2 Mouse learning progression

Acquisition of the task generally followed a typical progression: 1. discovery of water sources at the end of the radial arms (fig 1) , 2. repeated (generally ineffective) attempts to acquire further reward from the most recently rewarded port, 3. adoption of patterns of biased action, such as clockwise or counterclockwise circling of the physical arms, 4. decreased repetition of ineffective action sequences, combined with increased perseverance following unrewarded actions (due to gradual resource depletion via the PacMouse reward function).

A.3 Latent maze task POMDP formalism

A partially observable Markov decision process (POMDP) is a tuple $(S, A, \mathcal{T}, R, \Omega, O)$, where $\mathcal{T}(s' \mid s, a)$ are transition probabilities, $R(s, a)$ is the reward, Ω is the observation set, and $O(o \mid s', a)$ the observation model. In our setting, the state space S is a set of nodes $\{n_1, \ldots, n_N\}$ arranged on a square integer lattice such that each node has an associated (x, y) coordinate in $\mathbb{Z}^2$. A is a set of the four actions. The actions can be represented in three systems: egocentric (forward, back, left, and right), allocentric-latent (north, south, west, east relative to the positive y axis in the lattice), and allocentric-real (the actual sensors that were or *would be*, in the case of simulations, poked by the mouse). All transition probabilities in $\mathcal{T}$ are fully deterministic, but governed by a latent graph defined as follows:

$$G = (V, E), \quad V \subseteq \mathbb{Z}^2.$$

The *lattice adjacency set* is

$$E_{\text{lattice}} = \big\{\{(x, y), (x', y')\} : (x, y), (x', y') \in V,\ |x - x'| + |y - y'| = 1\big\},$$

so edges correspond to unit steps along the coordinate axes between adjacent lattice nodes. The actual set of edges in the maze is then chosen as an arbitrary subset $E \subseteq E_{\text{lattice}}$, specifying the passable connections available to the mouse. Based on prior mouse performance, the experimenter selects E at the beginning of each session, which is passed via an adjacency matrix file to the microcontroller Python script, effectively realizing a given latent maze as a subgraph of the lattice. Let $\mathcal{D} = \{(1, 0),\ (-1, 0),\ (0, 1),\ (0, -1)\}$ denote the set of unit step directions. At each time step, the agent may attempt to move from a vertex $v \in V$ to $v + d$ for any $d \in \mathcal{D}$, but the transition is valid only if $\{v, v + d\} \in E$.

Transitions are deterministic and constrained by the maze edge set E:

$$\mathcal{T}(s' \mid s, a) = \begin{cases} 1, & \text{if } s' = s + a \text{ and } \{s, s'\} \in E, \\ 0, & \text{otherwise.} \end{cases}$$

The underlying action set A is defined by the direction vectors $\mathcal{D}$, but can be expressed in three different coordinate frames as described in the main text.

A.3.1 PacMouse reward function:

Similar to the classic arcade game PacMan, rewards depend on whether a state has been previously visited. Let $r > 0$ denote the fixed reward for consuming a state the first time it is entered, and let $U_t \subseteq S$ be the set of unvisited states at time t. Then

$$R_t(s, a) = \begin{cases} r, & \text{if } s \in U_t, \\ 0, & \text{otherwise.} \end{cases}$$

In the mouse experiments, r was a fixed quantity of fluid specified by the experimenter. Each mouse experiment session or simulation episode began with $U_{t=0} = S$. After visiting s_t, the unvisited set updates as

$$U^\star_{t+1} = U_t \setminus \{s_t\}, \qquad U_{t+1} = \begin{cases} S, & \text{if } |U^\star_{t+1}| < N, \\ U^\star_{t+1}, & \text{otherwise,} \end{cases}$$

where $N \in \mathbb{N}$ is a fixed reset threshold. Thus each state yields reward exactly once per cycle, and the cycle resets when fewer than N unvisited states remain.

A.4 Data analysis and simulation infrastructure

All relevant part numbers and experiment apparatus assembly instructions, as well as all data analysis and simulation code will be published to a public GitHub repo upon acceptance.

A.5 Supplementary figures

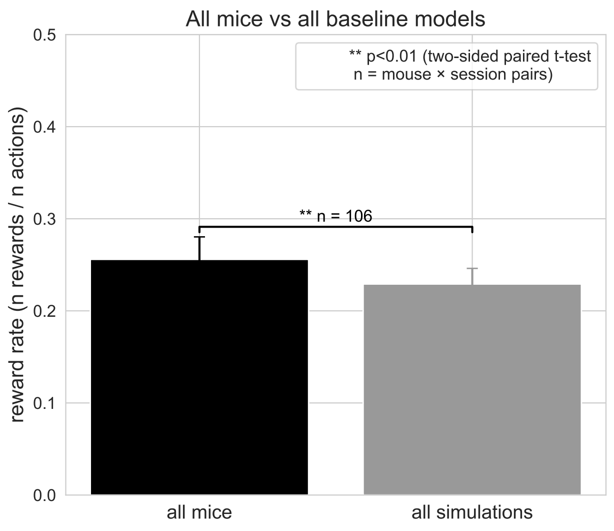

Figure S1: Performance of mice vs. yoked-baseline simulations. Reward rates were computed per mouse per session, with simulation rates averaged across all models, and compared using two-sided paired-sample t-tests.

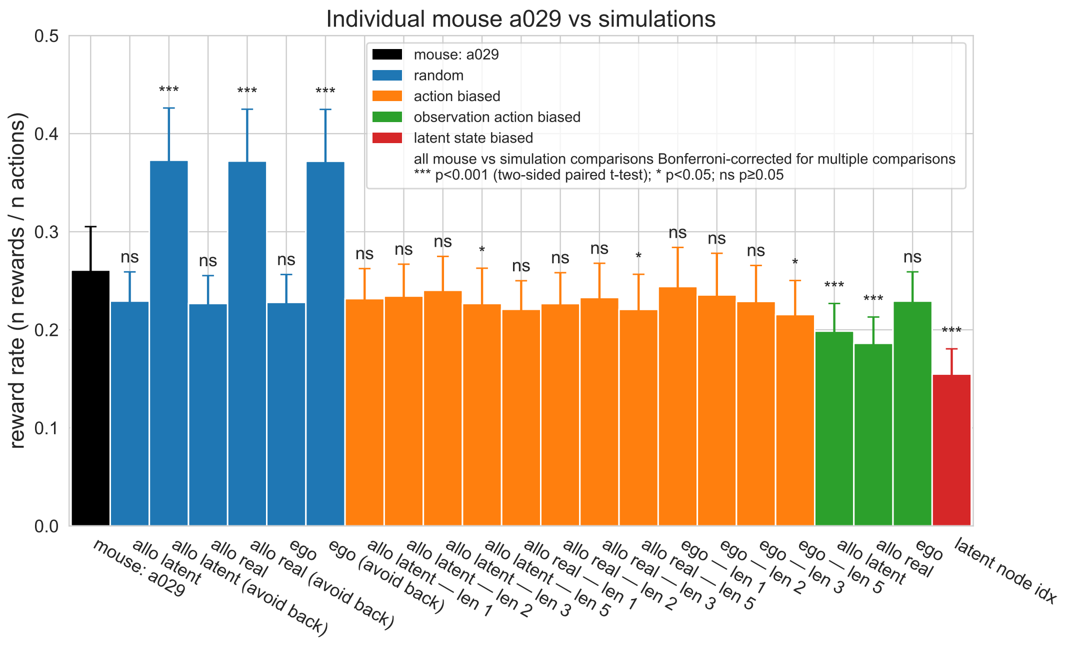

Figure S2: Performance of another exemplar mouse vs yoked-baseline simulations. Asterisks indicate significance (paired two-sided t-test, Bonferroni-corrected).

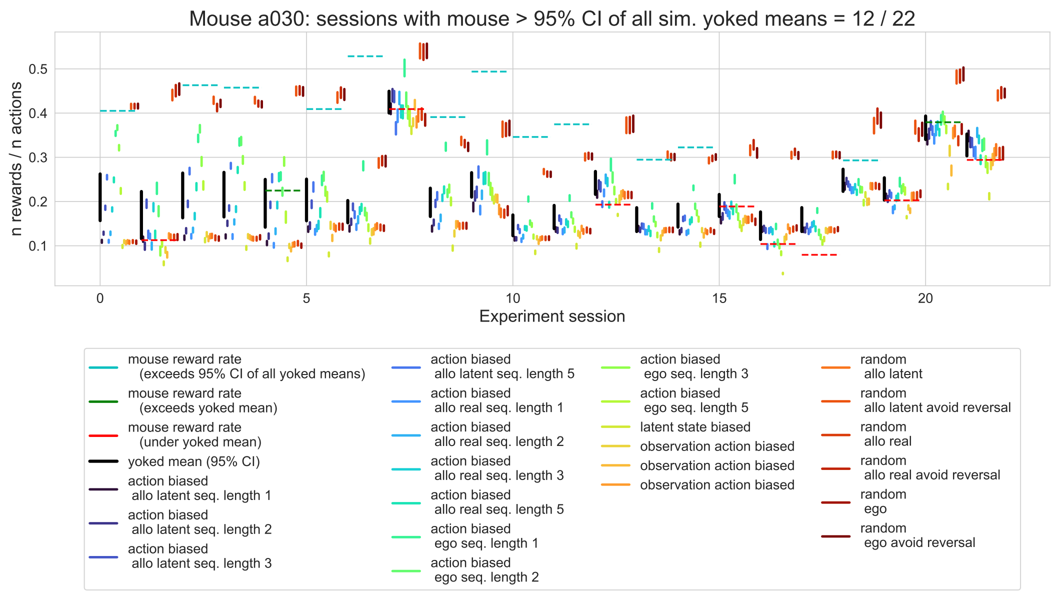

Figure S3: Performance of an exemplar mouse vs yoked-baseline simulations across each consecutive experiment session. Dashed horizontal lines indicate mouse performance while vertical lines show the 95% confidence interval of the given model (colored) or all models (black).