Emergent behavior and neural dynamics in artificial agents tracking turbulent plumes

Satpreet H. Singh,*, Floris van Breugel, Rajesh P. N. Rao, Bingni W. Brunton

*Department of Electrical and Computer Engineering, University of Washington, Seattle, USA.
†Department of Mechanical Engineering, University of Nevada, Reno, USA.
‡Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, USA.
§Center for Neurotechnology, University of Washington, Seattle, USA.
¶Department of Biology, University of Washington, Seattle, USA.
∗eScience Institute, University of Washington, Seattle, USA.

Abstract

Tracking a turbulent plume to locate its source is a complex control problem because it requires multi-sensory integration and must be robust to intermittent odors, changing wind direction, and variable plume statistics. This task is routinely performed by flying insects, often over long distances, in pursuit of food or mates. Several aspects of this remarkable behavior have been studied in detail in many experimental studies. Here, we take a complementary in silico approach, using artificial agents trained with reinforcement learning to develop an integrated understanding of the behaviors and neural computations that support plume tracking. Specifically, we use deep reinforcement learning (DRL) to train recurrent neural network (RNN) agents to locate the source of simulated turbulent plumes. Interestingly, the agents’ emergent behaviors resemble those of flying insects, and the RNNs learn to represent task-relevant variables, such as head direction and time since last odor encounter. Our analyses suggest an intriguing experimentally testable hypothesis for tracking plumes in changing wind direction—that agents follow local plume shape rather than the current wind direction. While reflexive short-memory behaviors are sufficient for tracking plumes in constant wind, longer timescales of memory are essential for tracking plumes that switch direction. At the level of neural dynamics, the RNNs’ population activity is low-dimensional and organized into distinct dynamical structures, with some correspondence to behavioral modules. Our in silico approach provides key intuitions for turbulent plume tracking strategies and motivates future targeted experimental and theoretical developments.

Keywords: deep reinforcement learning, olfactory search, recurrent neural networks, computational neuroscience, control theory

1 Introduction

Locating the source of an odor in a windy environment is a challenging control problem, where an agent must act to correct course in the face of intermittent odor signals, changing wind directions, and the variability in odor plume shape [Celani et al., 2014]. Moreover, an agent tracking an intermittent plume needs memory, where current and past egocentric odor, visual, and wind sensory signals must be integrated to determine the next action. For flying insects, localizing the source of odor plumes emanating from potential food sources or mates is critical for survival and reproduction. Therefore, many aspects of their plume tracking abilities have been experimentally studied in great detail [Baker et al., 2018; Cardé and Willis, 2008; Currier and Nagel, 2020; Park et al., 2016]. However, most such studies are limited to one or two levels of analysis such as behavior [van Breugel et al., 2008], computation [Lochmatter and Martinoli, 2009; Pang et al., 2018] or neural implementation [Sun et al., 2018].

Despite the wide adoption of wind tunnel experiments to study odor plume tracking [van Breugel and Dickinson, 2014], generating controlled dynamic turbulent plumes and recording flight trajectories at high resolution is expensive and laborious. Exciting alternative approaches have been developed using virtual reality [Kaushik et al., 2020] and kilometer-scale outdoor dispersal experiments [Leitch et al., 2020]. While behavioral experiments are now tractable, collecting significant neural data during free flight in small insects remains technologically infeasible, and larger insects require larger wind tunnels. Here we are motivated to take a complementary in silico approach using artificial recurrent neural network (RNN) agents trained to track simulated turbulent plumes, with the goal of developing an integrated understanding of the behavioral strategies and the associated neural computations that support plume tracking.

*Author for correspondence

Email address: satsingh@uw.edu (Satpreet H. Singh)
In recent years, artificial neural networks (ANNs) have gained increasing popularity for modeling and understanding aspects of neural function and animal behavior [Cichy and Kaiser, 2019; Kietzmann et al., 2019], including vision [Kriegeskorte, 2015], movement [Sussillo et al., 2015], navigation [Cueva et al., 2019; Cueva and Wei, 2018; Haeseley et al., 2019; Kanitscheider and Pietz, 2017], and collective behaviors [Verma et al., 2018]. Whereas many ANNs have been trained using supervised approaches that rely on labeled training data, an alternative emerging algorithmic toolkit known as deep reinforcement learning (DRL) has made it computationally tractable to train ANN agents (Figure 1d). In particular, an ANN agent receives sensory observations and task-aligned rewards based on its actions at each step and tries to learn a strategy for its next actions to maximize total expected reward [Arulkumaran et al., 2017; Sutton and Barto, 2018]. Such learning and optimization based models are normative in the sense that they can prescribe how a neural system *should* behave, rather than describing how it has been observed to behave. As neuroscience moves towards studying increasingly naturalistic behaviors [Gomez-Marin and Ghazanfar, 2019; Huk et al., 2018; Nastase et al., 2020; Sonkusare et al., 2019], such normative approaches are gaining traction as tools to gain insight, rapidly explore hypotheses, and generate ideas for theoretical development [Ahrens, 2019; Banino et al., 2018; Colabrese et al., 2017; Le Moël and Wystrach, 2020; Merel et al., 2019; Richards et al., 2019; Verma et al., 2018].

Flying insects search for sources of odor using several strategies, depending on the spatial scale being considered and odor source visibility [Baker et al., 2018] (Figure 1a). Close to the odor source, insects can fly to the source guided by vision. At larger ranges (from several meters up to about 100 meters [Wall and Perry, 1987]) or when the odor source is not yet visible, their search must be guided by olfaction to detect odors and mechanosensation to estimate wind velocity. At this larger scale, there are a few stereotyped behavioral sequences that are known to be important for plume tracking: *upwind surges* when the insect can sense the odor, and *crosswind casts* and *U-turns* to locate the plume body when the insect loses the odor scent [Cardé and Willis, 2008]. Here we focus on this larger-scale odor and wind guided regime, where agents have access to only mechanosensory and olfactory cues.

In this paper, we describe behaviors that emerge in RNN agents trained to track odors in a flexible plume simulation and analyze the neural dynamics that underlie these behaviors. At a behavioral level, we find that the agents’ actions can be summarized by several modules that closely resemble those observed in flying insects (Section 4.1). While odor plumes that do not change in direction can be tracked using a few steps of history, longer timescales of memory are essential for plumes that are non-stationary and change directions unpredictably. Interestingly, the learned tracking behavior of RNN agents in non-stationary plumes suggests a testable experimental hypothesis: that tracking is accomplished through local plume shape rather than wind direction (Section 4.2). The RNNs learn to represent variables known to be important to flying insect navigation, such as head direction and time between odor encounters (Section 4.4). Further, the low-dimensional neural activity associated with the emergent behavior modules is structured into two distinct regimes (Section 4.5), and transitions between these regimes are asymmetric in duration (Section 4.6).

2 Related work

An emerging body of work has used DRL to train ANNs that solve tasks closely inspired by tasks from neuroscience. For instance, agents have been trained to study learning and dynamics in the motor-cortex [Song et al., 2020; Weinstein and Botvinick, 2017], time encoding in the hippocampus [Lin and Richards, 2021], reward-based learning and meta-learning in the pre-frontal cortex [Botvinick et al., 2019; Song et al., 2017; Wang et al., 2018], and task-associated representations across multiple brain areas [Cross et al., 2021]. There have been several recent perspectives articulating the relevance of this emerging algorithmic toolkit to neuroscience [Botvinick et al., 2020; Gershman and Ölveczky, 2020] and ethology [Crosby, 2020].

Our work is most directly related to three recent research efforts. Merel et al. [2019] developed a virtual-reality model of a rodent embodied in a skeleton body and endowed with a deep ANN ‘brain.’ They trained this model using DRL to solve four tasks and then analyzed the virtual rodent’s emergent behavior and neural activity, finding similarities at an abstract level between their agent and observations from rodent studies. Reddy et al. [2021] studied the trail tracking strategies of terrestrial animals with one (e.g. one antenna) or two (e.g. two nostrils) odor sensors. They found that RL agents trained on simulated trails recapitulate the stereotypical zig-zagging tracking behavior seen in such animals. Using a static trail model and an explicit (not neural) probabilistic model for sensory integration, they studied the effect of varying agent and task parameters on the emergent stereotypical zig-zagging behavior. Rapp and Nawrot [2020] used a biologically detailed spiking neural circuit model of a fly mushroom body to study sensory processing, learning, and motor control in flying insects when foraging within turbulent odor plumes.

We build on the approach of these recent papers that study artificial agents solving neural inspired tasks, and our work is also distinct in several key ways. First, we simulate a more computationally challenging task than those tackled in Reddy et al. [2021] and Rapp and Nawrot [2020], because our odor environment is configurable, dynamic, and stochastic. Second, we have made several simplifications and abstractions that make analysis more tractable, so that we may focus on the general principles behind plume tracking. Specifically, we omit biomechanical details, impose no biologically inspired connectivity
Figure 1: Training artificial agents to track turbulent plumes with deep reinforcement learning. (a) A schematic of a flying insect performing a plume tracking task, showing upwind surges, crosswind casts, and U-turns behaviors (inspired by a figure in Baker et al. [2018]). In this work, we model the spatial scale (dashed rectangle) where the insect can use only olfactory and mechanosensory (to sense wind velocity) cues for plume tracking. (b) The plume simulator models stochastic emission of odor packets from a source carried by wind. Odor packets are subject to advection by wind, random cross-wind perturbation, and radial diffusion. (c) An example of a plume simulation where the wind direction changed several times. (d) A schematic showing how the artificial agent interacts with the environment at each time step. The environment model determines the sensory observations available to the agent \( x \) (egocentric wind direction vector and local odor concentration) and the rewards used in training. The agent navigates within the environment with actions \( a \) (turn direction and magnitude of movement). (e) Agents are modeled as neural networks and trained by deep reinforcement learning (DRL). A recurrent neural network (RNN) generates an internal state representation from sensory observations, followed by parallel Actor and Critic heads that implement the agent's control policy and predict the state values, respectively. The Actor and Critic heads are 2-layer, feedforward multi-layer perceptron (MLP) networks. (f) A schematic showing a flying agent’s head-direction, course-direction, and the wind direction, all measured with respect to the ground and counter-clockwise from the x-axis. Course direction is the direction that the agent actually moves in, accounting for the effect of the wind on the agent’s intended direction of movement (head-direction). Egocentric wind direction is the direction of the wind as sensed by the agent and is measured counter-clockwise with respect to the agent’s current head-direction.

Constraints, and do not use spiking neurons. Instead, our networks are ‘Vanilla’ RNNs (rather than the gated RNNs used in Merel et al. [2019] or the spiking neurons in Rapp and Nawrot [2020]), which facilitates analyses from the dynamical systems perspective [Maheswaranathan et al., 2019a,b; Rajan and Abbott, 2006; Sussillo and Barak, 2013; Vyas et al., 2020]. We analyze emergent behaviors and neural dynamics at the network level, which provides us with an abstract understanding of task-relevant neural computations that is robust to small changes in network architecture and training hyperparameters [Maheswaranathan et al., 2019b; Sussillo and Barak, 2013; Vyas et al., 2020]. Lastly but importantly, since we do not model vision or joint-level motor control as in Merel et al. [2019], our neural networks are simpler and can be trained on a computational budget accessible to an academic lab.

3 Training artificial agents to track turbulent plumes

Here we describe how we use DRL to train RNN agents that can track simulated turbulent odor plumes. Training episodes situate agents at random initial locations within plumes that switch directions multiple times during the
course of the episode. Agents are actor-critic neural networks that receive sensory observations as inputs—namely, egocentric instantaneous wind velocity and local odor concentration. Importantly, since the plume simulator models diffusing and advecting odor packets, the agent’s encounters with odor packets are intermittent and stochastic. To train our agents to solve this task where both the observation space and action space are continuous valued, we use the Proximal Policy Gradient (PPO) algorithm. For evaluation, we assess trained agents on additional simulations where the odors are relatively sparse and the wind switches direction at different frequencies. More details of our implementation are available in Appendix A and Appendix B.

3.1 Plume simulation

We implement a particle-based two-dimensional plume simulation model (Figure 1f) that replicates the statistics of real-world turbulent plumes [Farrell et al., 2002]. This type of simulation has been used in a wide range of domains including olfactory navigation [Cardé and Willis, 2008], robotics [Kowadlo and Russell, 2008] and sensor networks [Michaelides and Panayiotou, 2005]. The simulator (Figure 1b) comprises a spatially homogeneous wind vector-field (0.5 m/s with configurable direction) and an odor source located at the origin that emits odor puffs as a Poisson process. Puffs are initialized with a fixed initial radius and undergo radial diffusion. In addition, each emitted puff is advected downwind at the wind velocity and perturbed randomly by crosswind translation. In other words, each puff effectively performs a biased random walk downwind over time, while diffusing in concentration spatially. Our simulated plumes and agents are constrained to two dimensions for simplicity of analysis. The dimensions of the simulated arena are $[-2m, +10m]$ and $[-5m, +5m]$ in the x and y coordinates respectively, totaling a $120m^2$ arena. Plumes are simulated at 100 iterations/second. The plume’s centerline is obtained by simulating puffs that have no random crosswind translation at each iteration (Figure 1f).

We simulate the following four wind configurations. First, the wind direction is held constant (0°) throughout the simulation (‘constant’). Second, the wind direction makes one 45° counter-clockwise switch during a tracking episode (‘switch-once’). Third, the wind direction switches at multiple random times during a tracking episode (‘switch-many’). Each wind direction turn is a random draw from a Gaussian distribution with mean 0 and 45° s.d., truncated at $\pm 60°$, and occurs approximately every 3 seconds. Fourth, the wind direction is held constant, but the puff birth-rate is reduced (0.4x) compared to the ‘constant’ configuration (‘sparse’). See Appendix A for further details on the plume simulation.
Emergent plume tracking behavior can be decomposed into behavior modules. (a & b) Trajectories for a successful (a) and an unsuccessful (b) plume tracking episodes showing three distinct behavior modules: tracking (green), lost (red) and recovering (purple-blue) (c) Kernel density estimates show data aggregated from an equal number of successful and unsuccessful constant wind direction plume tracking episodes (N timesteps, E episodes). Plots reveal differences between the three behavior modules across key behavioral measures: Head-direction: Head-direction densities are concentrated around ±180°, a signature of zig-zagging but mostly upwind movement; the concentration of density around the upwind direction reduces from tracking to recover to lost, accounting for the more complex trajectories encountered in the latter two behavior modules. (angle measured counterclockwise with 0° indicating directly downwind). ∆x and ∆y: Density estimates for drift in the x-direction (∆x) and y-direction (∆y) per timestep show how tracking is characterized by primarily upwind (negative x-direction) movement in both tracking and recover modules, but lesser so in the lost module. Y-direction movements are significant in the tracking and recovering modules, corresponding to more complex turning behaviors, but minimal in the lost module. Turn action: Left/right turning movements are balanced in the tracking module as the agent closely tracks the edge of the plume, but is biased towards clockwise movements in the other two modules, especially the lost module. Move action: The agent significantly modulates its forward movement speed in the lost module only. Stray distance: The agent strays from the plume minimally in the tracking module, but significantly otherwise. See Appendix C for equivalent plots for other agents.

3.2 Agent architecture

Our agents are actor-critic networks (Figure 1c), where a recurrent neural network (RNN) receives sensory observations and passes a transformed representation of them onto parallel Actor and Critic heads that are both two-layer multi-layer perceptrons (MLPs) [Konda and Tsitsiklis, 2000]. The Actor head implements a control policy to map the RNN’s learned state representation to actions, while the Critic head implements a value function that maps the state representation to an estimate of the state’s value based on rewards. This value function is used only during agent training and not thereafter. In the DRL literature, two-layer deep heads are typically sufficiently expressive for such control problems [Hill et al., 2018]. At each time step, an agent receives a 3-dimensional real-valued input vector comprising egocentric wind velocities (x, y) and odor concentration at its current location. In response, the agent produces continuous valued turn (maximum ±6.25π radians/s) and forward-movement (maximum 2.5 m/s) actions; these velocities are matched to the capabilities of flying fruit flies [van Breugel and Dickinson, 2014; van Breugel et al., 2008]. In contrast to the orthogonal initialization typically employed in the mainstream machine learning literature [Henaff et al., 2016], we initialize our RNNs with normally distributed weights to facilitate comparisons with the computational neuroscience literature [Sussillo, 2014; Vogels et al., 2005; Yang et al., 2019].

Additionally, to understand the role of memory on tracking performance (Section 4.3), we compare the RNN-based agents with an alternative feedforward-only network (multi-layer perceptron, MLP) architecture with fixed-length memory, simulated by appending historical sensory observations onto instantaneous network inputs [Mnih et al., 2013]. Although such MLPs are far from being biologically plausible architectures, they serve as useful tools for abstract comparison since their memory capacities can be controlled precisely. Both RNN and MLP layers across
4.3 Agent training and evaluation

We train our agents using the Proximal Policy Gradient (PPO) algorithm [Schulman et al., 2017], which is known to robustly solve continuous observation-space continuous action-space control problems without needing significant hyperparameter tuning. To guide agent training, we developed a curriculum and a simple reward function that greatly rewards homing in on odor source, mildly rewards actions that reduce the radial distance between agent and odor source, and penalizes longer duration trajectories and straying too far from the plume. We train 14 independently randomly initialized networks for each architecture type, i.e. RNNs and MLPs with 2, 4, 6, 8, 10 & 12 timesteps of observation history.

Next, we evaluate each trained agent’s performance with a behavioral assay. Each trained agent is evaluated with 240 episodes at different initializations (15 initial locations, 2 initial simulation timestamps, and 8 initial head-directions), and at each of the ‘constant’, ‘switch-once’ and ‘switch-many’ plume configurations. For each architecture type, we proceed to analyze only the top 5 seeds with the best performance, as measured by total number of successful episodes across the four plume configurations. Agent training/evaluation episodes are run at 25 frames per second on a sub-sampled plume and limited to 300 frames/timesteps (12 seconds of flight) per episode to accelerate DRL training. See Appendix A for additional details on agent training and evaluation, and Appendix B for a full list of associated hyperparameters.

4 Behavior and neural dynamics of trained agents

Our trained agents learn strategies to successfully localize plume sources in turbulent, non-stationary environments. In this section, we characterize their performance, then highlight their emergent behavioral and neural features. In addition to comparing artificial agents to biology, we discover behavioral strategies that motivate future experiments and gain intuition about the neural computations underlying these emergent behaviors.

Unless otherwise specified, this section describes results from one agent randomly chosen from the trained RNNs as evaluated on randomly selected episodes in the evaluation subset. This evaluation subset is chosen to include trajectories from test behavioral assays, balanced across successful and unsuccessful tracking episodes, in each of the ‘constant,’ ‘switch-once,’ and ‘switch-many’ configurations (see Appendix A.3 for evaluation details).

4.1 Agents track plumes with varying wind conditions using distinct behavioral modules

Our trained RNN agents are able to complete the plume tracking task with changing wind direction and varying plume sparsity (Figure 2 shows example trajectories). The observed trajectories can be summarized as one of three behavior modules, determined approximately by the time elapsed since the agent last sensed odor (Figure 3). We refer to these three modules as tracking, lost, and recovering. In the tracking module, the agent rapidly moves closer to the plume source, using either straight-line trajectories when it is well within the plume, or a quasi-periodic ‘plume skimming’ behavior where it stays close to the edge of the plume while moving in and out of it. The interval between the agent’s encounters with odor packets in this module is
Figure 5: Memory capacity improves plume tracking, especially in non-stationary wind direction plumes. Number of successful homing episodes for different agent architectures, across different plume configurations for the same set of 240 initial conditions across varying agent starting location and head direction, and plume simulator state (Section 3.3). ‘MLP_X’ refers to feedforward networks with X timesteps of sensory history (Section 3.2). RNNs generally outperform feedforward networks, with more pronounced gains for more complex, switching wind direction (‘switch-once’, ‘switch-many’) plume tasks. In feedforward networks, performance on plumes with switching wind direction can improve significantly with increasing memory. Regression lines (solid black) are fit on only MLP data, but are extended slightly (dotted line) for comparison with RNNs (p-values are for a two-sided Wald Test with null hypothesis that the slope is zero).

under 0.5 seconds. Recovering corresponds to an irregular behavior where the agent makes large, usually cross-wind, movements after having lost track of the plume for a relatively short period of time (about 0.5 second). Lost corresponds to a periodic behavior that appears variable across trained agents as either a spiraling or slithering/oscillating motion, often with an additional slow drift in an arbitrary direction. This behavior is seen when the agent has not encountered the plume for a relatively long time, typically over 1 second. See Appendix C for the exact thresholds used to segment each agent’s trajectories into behavior modules.

Agents that are successful in tracking plumes in constant wind direction primarily use the tracking and recovering modules. Successful trajectories on the ‘switch-once’ and ‘switch-many’ plumes reveal that RNN agents use more complex strategies in the face of changing wind directions. If an agent is in the tracking module and well within the plume at the time of wind direction change, it continues along its path until it reaches the edge of the plume without changing its actions. If it is skimming the edge of the plume when the wind direction switch happens, it tries to compensate for the added movement of the plume by making more pronounced oscillations in and out of the plume. The shape of these oscillations appears to depend more on the local shape of the plume than on the current direction of the wind (explored further in Section 4.2). Finally, if the agent cannot keep up with the movement of the plume, it typically orchestrates a sequence of large oscillations and spiral-like movements, corresponding to the recovering and lost modules, to try to find the plume boundary. On returning to the plume, it resumes the tracking module behaviors once again.

4.2 Agents track plume centerline, not current wind direction

Successful trajectories in plumes that switch direction suggest that agents take the local shape of the plume into account, rather than just the current wind conditions (Figure 2c–d and supplementary animations). To quantify this, we look at the empirical distributions of the agent’s course direction computed with respect to the current wind direction, and with respect to the centerline of the nearby plume (Figure 1f). The agent’s course direction (Figure 1c) is defined as the direction of its instantaneous movement with respect to the ground. Subtracting the current wind direction angle from the course direction provides the course direction with respect to the wind. To find the course direction with respect to the centerline, we first find the median local centerline angle using centerline puffs (Section 3.1) within a ±2 c.m. band of the x-coordinate of the agent’s location, then subtract this from the course-direction with respect to the ground. Subtracting the current wind direction angle from the course direction provides the course direction with respect to the wind. To find the course direction with respect to the centerline, we first find the median local centerline angle using centerline puffs (Section 3.1) within a ±2 c.m. band of the x-coordinate of the agent’s location, then subtract this from the course-direction with respect to the ground. The empirical distributions include aggregate data from when agents are in the tracking module from up to 60 random successful trajectories from the ‘constant’, ‘switch-once’ and ‘switch-many’ plume configurations. Additionally, for the ‘switch-once’ configuration, we trim trajectories to consider only the timesteps after the wind direction switch has occurred.

Figure 4 shows the empirical course direction distributions are much better aligned with the plume centerline than to the wind for one example agent. For ‘switch-once’ plumes, the peak of the course direction distribution is

---

1 Animations: https://github.com/BruntonUWBio/plumetracknets/blob/main/constant.md
2 https://github.com/BruntonUWBio/plumetracknets/blob/main/switch-once.md
3 https://github.com/BruntonUWBio/plumetracknets/blob/main/switch-many.md
Figure 6: Neural activity is low-dimensional and represents biologically relevant variables. (a–d) Neural activity trajectories plotted over a diversity of plume conditions and tracking outcomes, (a) colored by agent head-direction $\Theta_{\text{HEAD}}$, (b) steps since last odor encounter $T_{\text{last}}$, (c) exponentially-weighted moving-average of odor concentration ($\text{odor}_{\text{EWMA}}$; window-size = 8 steps), and (d) exponentially-weighted moving-average of recent odor encounters ($\text{odor}_{\text{ENC}}$; window-size = 46 steps). Figure 6 (e) Quality of fit ($R^2$) of a linear model regressing neural activity onto $\text{odor}_{\text{EWMA}}$ and $\text{odor}_{\text{ENC}}$ for sliding-windows of varying lengths. The sliding-window size for subfigures (c) and (d) are determined by identifying the peaks of these curves. (f) Permutation importance scores of features of a classifier trained to predict agent actions using the aforementioned plotted features ($T_{\text{last}}$, $\Theta_{\text{HEAD}}$, $\text{odor}_{\text{EWMA}}$, and $\text{odor}_{\text{ENC}}$), and instantaneous sensory observations (wind $w_X$, $w_Y$ and odor). (g) Plot of cumulative variance explained by top principal components of neural activity aggregated across multiple plume configurations (‘constant’, ‘switch-once’ & ‘switch-many’) suggests a low-dimensional structure. 90% of the variance of the 64-dimensional neural activity can be explained by the first-5 principal components. See Appendix C for equivalent plots for all 5 RNN agents.

4.3 Recurrence and memory enable plume tracking

To understand the role of memory capacity in plume tracking, we compare the performance of our trained RNNs to trained feedforward networks (multi-layer perceptrons, MLPs) that receive varying timescales of sensory history (see Section 3.2 for more information on the MLP architecture). As seen in Figure 5, RNNs outperform MLPs for every plume tracking task, with the performance gains being largest in the most challenging tasks. For MLPs, longer duration sensory memories support much better performance on tougher tracking tasks, where the plumes switch more often or odor packets are sparser.

4.4 Neural activity is low-dimensional and represents task-relevant variables

We now turn our attention to the neural dynamics of the RNNs as agents perform plume tracking. Rather than characterizing the activity of individual units, we consider the population activity of the network [Ebitz and Hayden, 2021; Saxena and Cunningham, 2019].

First, we reduce and visualize the population activity of our RNN across the ‘constant’, ‘switch-once’ & ‘switch-many’ plume configurations and find that the neural ac-
Neural dynamics appear to organize themselves into overlapping yet distinct regimes. (a) Plume tracking episode that ends in successful homing-in on the odor source, and (b) Unsuccessful episode that strays from the plume and ends up exceeding the simulator’s bounds. (c & d) Neural activity plots corresponding to each row’s trajectory projected on a 2D subspace (state-space) generated from the first 2 principal components of that episode’s neural activity. Quiver arrows correspond to direction of neural activity gradient, and are colored by the agent’s current behavior module. (c) A ‘funnel’ like structure (in green) emerges in the state-space corresponding to the tracking behavior module. (d) The agent’s periodic lost behavior shows up as a limit-cycle in the state-space (red). (e) Neural activity plotted over multiple trajectories comprising a diversity of plume conditions and tracking outcomes, projected onto the first 3 principal components of the aggregated neural activity and colored by behavior module (Section 4.1). Examples from RNN agent 3. See Appendix F for equivalent plots for all 5 RNN agents.

To gain insights into the computations supporting the plume tracking behavior, we look for variables represented in this low-dimensional population activity that are relevant for solving the task. We find that the RNNs have learned to represent task-relevant quantities beyond the instantaneous sensory observations that are provided to it by the simulator (Figure 6a–d).

Interestingly, these quantities reflect information necessary for solving these challenging plume tracking tasks and require memories of past sensory cues encountered by the agent. First, the agent’s head-direction, or its the orientation with respect to the ground, is evident in Figure 6a. The time since plume was last encountered is encoded as in Figure 6b and may be involved in determining transitions between behavior modules. Whereas the agent only receives local odor concentrations as a sensory input, we find that an exponentially-weighted moving-average of sensed odor concentrations is present in Figure 6c. We conjecture this quantity may be useful as a memory in the face of an intermittent odor signal arising from a turbulent plume. Similarly, an exponentially-weighted moving-average of a discretized odor encounters signal is evident in Figure 6d.

We determine the window-sizes [Pandas, 2021] for odor concentrations and encounters by linearly regressing neural activity onto them for sliding-windows of varying lengths, and we chose the window-size that produces the best fit as measured by the coefficient of determination $R^2$ (Figure 6e). The best moving-average window length for time-averaged odor concentrations (7 timesteps or 0.3s on average across all 5 agents) is significantly shorter than that for time-averaged odor encounters (47 timesteps or 1.9s on average across all 5 agents). Time-averaged odor concentrations are also better encoded ($R^2=0.91$ on average across 5 agents) than time-averaged odor encounters ($R^2=0.59$ on average across 5 agents). See Appendix Table E.6 for data on each individual agent.

To quantify how important these represented variables are to actual task performance, we train a Random Forest (RF) [Breiman, 2001] classifier to predict the (discretized) actions taken by the agent over successful trajectories (see Appendix E for details). We also estimate the relative importance of each input feature by calculating its permutation importance score [Breiman, 2001; Strobl et al., 2008], which is an estimate of the reduction in the classifier’s accuracy across several (N=30) randomized permutations of that feature. Classifier accuracies using all aforementioned represented features (Figure 6) along with instantaneous sensory features is 10–18% higher across all agents than that using classifiers receiving just instantaneous sensory observations, and 26–51% higher across all agents than that produced by a majority-class classifier (See E.7 for each agent’s classifier accuracies). Represented variables have permutation importance scores within the range covered by the importance scores of the instantaneous sen-
Figure 8: Transitions between neural activity regimes are asymmetric in duration. (a) Neural activity from multiple tracking episodes plotted on same top-3 principal component subspace as in Figure 7. Points are colored by the neural regime centroid they are closest to (see Appendix G for centroid definitions; orange color for points nearer to Out-Of-Bounds/OOB centroid and blue for those closer to HOME centroid; black circles denote centroid locations). (b–d) Example trajectories where the agent (b) enters the tracking behavior module after entering the plume from the recovering behavior module, (c) enters the tracking behavior module after entering the plume from the lost behavior module, and (d) enters the lost behavior module after a brief encounter with the plume. (e–g) Time courses of neural activity distances to the HOME and OOB centroid (in blue and orange respectively), associated with the respective trajectories in the center column. Dotted vertical lines show time of entering or leaving plume, while dashed vertical lines show time when the agent has entered the target neural activity regime, i.e. neural activity is less than D/2 units away from target centroid, where D is the distance between centroids. (h) Box plots compare transition times to target regimes over a large set of trajectories across varying plume conditions (‘constant’, ‘switch-once’, ‘switch-many’). Transitions into the lost neural activity regime (TTL) tend to take longer than transitions into the tracking neural activity regime (TTT-NL or TTT-L) (two-sided Mann-Whitney-Wilcoxon test with Bonferroni correction, ****: p ≤ 1.00e-04, ns: not significant). See Appendix G for plots of all 5 RNNs.

4.5 Neural dynamics organized into two structured regimes and a transition region

We now examine the dynamics of the RNN’s hidden state $h$ and how it evolves over the course of tracking episodes. This analysis is inspired by previous work characterizing the nonlinear dynamics of RNN agents by their fixed-points and transitions among them [Maheswaranathan et al., 2019b; Sussillo and Barak, 2013; Vyas et al., 2020]. However, in a noteworthy deviation from these structures, we did not find any fixed points in our RNNs. Instead, our RNNs adopt neural dynamics that are better described by dynamical regimes. Specifically, the dynamics appear to organize themselves into overlapping but distinctly structures associated with the tracking and lost behavioral modules (Figure 7). Interestingly, the periodic spiral or oscillatory movements seen in the lost behavioral module appear to also have a quasi-periodic limit-cycle structure in the neural state space (Figure 7d), while the neural dynamics associated with the tracking behavior are represented as quasi-periodic ‘funnel’ like structures (Figure 7c). We also see an amorphous transition region associated with the recovering behavioral module. We see the same approximate structures (limit-cycles and funnel) emerge in the neural dynamics for 4 of the 5 RNN agents. See Appendix F and associated animations for data on all 5 agents.

4.6 Macroscopic transitions between neural activity regimes are asymmetric in duration

After having found distinct neural activity regimes for the tracking and lost behaviors in the previous section, we now explore transitions between these two regimes. Specifically, we look at differences in the duration between (1) when an agent enters the plume and when it ‘enters’ the tracking neural activity regime, and (2) when an agent
leaves the plume and when it ‘enters’ the lost neural activity regime. Entry into a neural activity regime is determined by when the neural activity is within a pre-specified distance from a ‘centroid’ corresponding to that regime (see details in Appendix G). As shown in Figure 8, the time taken to enter the lost neural activity regime after the agent leaves the plume is significantly longer than the time taken to enter the tracking neural activity regime after the agent enters the plume. In Appendix G, we see that this trend holds across 4 out of 5 agents.

4.7 Connectivity of trained RNNs reveal signatures of instability and memory

The weight matrices and recurrence Jacobians of our RNNs after training offer some theoretical insights into how the neural dynamics of the artificial agents are shaped to track turbulent plumes. The update rule for a Vanilla RNN with hidden state vector \( \mathbf{h}_t \) is given by

\[
\mathbf{h}_t = F(\mathbf{h}_{t-1}, \mathbf{x}_t) = \tanh (\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t + b),
\]

where \( \mathbf{W}_h \) is recurrence (connectivity) matrix of the hidden layer, \( \mathbf{x}_t \) are the network’s inputs, \( \mathbf{W}_x \) is the input-to-hidden layer matrix, and \( b \) is a bias term [Sussillo and Barak, 2013].

We find that the training process reorganizes the eigenvalue spectrum of the RNN recurrence matrix \( \mathbf{W}_h \) (Figure 9a). Before training, weights are initialized as normally distributed random variables. After training, there are multiple eigenvalues outside the unit circle in the complex plane. Interestingly, for all 5 agents, there is at least one strictly real-valued eigenvalue larger than unity. Along with external stimuli, these unstable eigenvalues drive the network’s hidden dynamics.\(^5\)

Next, we consider a linearization of this nonlinear system around arbitrary expansion points. The RNN update equation can be linearized around an arbitrary expansion point \((\mathbf{h}^e, \mathbf{x}^e)\) to get a linear dynamical system approximated by:

\[
\mathbf{h}_t \approx F(\mathbf{h}^e, \mathbf{x}^e) + \mathbf{J}^{\text{rec}} \Delta \mathbf{h}_{t-1} + \mathbf{J}^{\text{imp}} \Delta \mathbf{x}_t,
\]

where \( \Delta \mathbf{h}_{t-1} = \mathbf{h}_{t-1} - \mathbf{h}^e \) is the state of the linearized system, \( \Delta \mathbf{x}_t = \mathbf{x}_t - \mathbf{x}^e \) is the linearized system’s input, \( \mathbf{J}^{\text{rec}} \) is the recurrence Jacobian, and \( \mathbf{J}^{\text{imp}} \) is the input Jacobian [Maheswaranathan et al., 2019a]. To be explicit,

\[
\mathbf{J}^{\text{rec}} = \frac{\partial F(\mathbf{h}, \mathbf{x})}{\partial \mathbf{h}}, \quad \mathbf{J}^{\text{imp}} = \frac{\partial F(\mathbf{h}, \mathbf{x})}{\partial \mathbf{x}}.
\]

Note that \( \mathbf{J}^{\text{rec}}|_{(0,0)} = \mathbf{W}_h \) and \( \mathbf{J}^{\text{imp}}|_{(0,0)} = \mathbf{W}_x \).

Following the approach of [Maheswaranathan et al., 2019a], we consider the eigenvalues of the recurrence Jacobian and associated stimulus integration timescales along the trajectories of several episodes. This timescale governs the integration of stimuli in the direction of the corresponding eigenvectors. We chose at random one successful and one unsuccessful episode from each of three plume configurations (‘constant,’ ‘switch-once,’ and ‘switch-many’). At each time step of the trajectory, we computed the recurrence Jacobian assuming zero input \( \mathbf{J}^{\text{rec}}|_{(h,0)} \). The stimulus integration timescale \( \tau_i \) associated with a stable eigenvalue \( \lambda_i \) (i.e., \( |\lambda_i| \leq 1 \)) can be interpreted as a timescale with the conversion \( \tau_i = (1/|\ln |\lambda_i||) \).

\(^5\)Animations showing how recurrence Jacobian eigenspectra change over the course of training episodes: https://github.com/BruntonUWBio/plumetracknets/blob/main/VRNN3-eigen.md
Comparing the time-averaged stimulus integration timescales with those from the untrained RNN reveals that training adjusts these timescales to lie well within the maximum episode length of 300 timesteps (Figure 9b). Furthermore, we see that the bulk of these timescales are within about 12 timesteps ($\approx 0.5$s), suggesting that the plume tracking task predominantly needs short timescale memories. In Appendix H, we see that this trend holds across all 5 RNNs.

5 Connections to tracking turbulent plumes in biology

Our artificial RNN agents exhibit similarities to biology at the levels of behavior, computation, and neural dynamics. In this section, we draw these comparisons, discuss the significance of these connections, and suggest theoretical insights that may be relevant for researchers interested in biological plume tracking.

5.1 Behavioral features

The complex behavior exhibited by our agents can be decomposed into simpler modules, sequenced by the time elapsed since the agent last encountered the plume (Section 4.1). These modules show features similar to upwind surging, crosswind casting and U-turn behaviors previously reported in many studies on moths, fruit flies, and other flying insects [Baker et al., 2018; van Breugel and Dickinson, 2014; Budick and Dickinson, 2006; Cardé and Willis, 2008]. Furthermore, the variable sequencing behavior modules resembles the odor loss activated clock mechanism that has been previously proposed to drive changes in flight behavior in moths [Baker, 1990; Kennedy, 1983; Kennedy and Marsh, 1974].

Our observations make a behavioral hypothesis that agent track plumes with respect to the centerline rather than with respect to the current wind direction (Section 4.2). In a previous study on tracking in constant wind direction plumes, Grünbaum and Willis [2015] proposed a model where insects explicitly performed upwind surges when close to the plume centerline. However, a later study by Pang et al. [2018] failed to find support for this model. Our analysis provides intuition for the role of centerline tracking in non-stationary plumes and suggests a testable hypothesis: we predict that centerline tracking behaviors will be more apparent in flying insects when they track plumes in wind that switches direction.

5.2 Neural representations

Our RNN agents learn to represent variables that have been previously reported to be crucial to odor navigation (Section 4.4). First, agent head-direction has been found to be implemented as a ring attractor circuit in the central complex of many flying insects and is implicated in navigation [Green et al., 2017; Kim et al., 2019; Okubo et al., 2020; Pfeiffer and Homberg, 2014; Seelig and Jayaraman, 2015]. Second, time since plume was last encountered is analogous to the hypothesized internal-clock that determines behavior switching in moths [Baker, 1990; Kennedy, 1983; Kennedy and Marsh, 1974]. Additionally, Park et al. [2016] showed how this variable is encoded by the bursting olfactory receptor neurons (bORNs) in many animals, and that it contains information relevant to navigating in turbulent odors.

Third, exponentially moving-average of odor encounters was found by Demir et al. [2020] to determine the probability of turn and stop behaviors in walking flies navigating in turbulent plumes. Specifically, higher odor encounter rates were associated with more frequent saw-cadic upwind turns [Celani, 2020]. Fourth, exponentially moving-average of sensed odor concentration is motivated by theoretical work by Maheswaranathan et al. [2019a] that posits exponentially-weighted moving averages to be good canonical models for stimulus integration in RNNs. Between these two time-averaged odor variables, the best represented window length for time-averaged concentration is significantly shorter ($\approx 0.3$s) than that for time-averaged encounters ($\approx 1.9$s). Furthermore, we find that time-averaged odor concentration is relatively better represented and more important in predicting agent behavior, corroborating the intuition that turn decisions during flight would require quick decision making on sub-second timescales. We note that alternative variables beyond these four may exist that better explain agent navigation decisions.

5.3 Neural dynamics

The agents’ neural activity is low dimensional and structured, with an interesting asymmetry in macroscopic transitions between these structures. Like often seen in neurobiological recordings [Cunningham and Byron, 2014; Pang et al., 2016], the population activity of our RNNs is low-dimensional, with the top 5–8 principal components explaining an overwhelming majority of the 64-dimensional population’s total variance (Section 4.4).

The neural dynamics associated with behavior modules further exhibit interesting structure. Lost behaviors are represented as quasi-limit-cycles, while tracking behaviors show a ‘funnel’ like structure (Section 4.5). Similar 1-D circular manifolds and 2-D funnels [Kriegeskorte and Wei, 2021; Vyas et al., 2020] have been previously reported on the representational geometry of sensory populations, but not, to the best of our knowledge, in the closed-loop agent setting.

Finally, we find that the interval between entering the neural activity cluster associated with the lost behavior and leaving the plume, is significantly longer than the interval between entering neural activity cluster associated with the tracking behavior and entering the plume. This asymmetry in macro-scale transitions in the neural state resembles an asymmetry in behavior transitions reported in van Breugel and Dickinson [2014], where the authors experimentally observe that flies take about twice as long
to cast crosswind after plume loss, than to surge upwind on encountering attractive odors.

5.4 The role of memory

Two independent analyses give us insight into the memory requirements of the plume tracking task. First, our comparison of RNN agents to MLPs with fixed amounts of sensory history input (Section 4.4.3) suggests that longer sensory histories and recurrence lead to better performance on tougher tracking tasks, such as those with plumes that switch one or more times. Second, analyzing stable eigenmodes of the RNN recurrence Jacobians suggests that only a couple of long stimulus integration timescales are involved in the neural computation. The bulk of stimulus integration timescales are within ~12 steps or 0.5s (Section 4.7). Together, we believe that memory is crucial for tracking non-stationary wind direction plumes, but short timescale (under ~0.5s) and reflexive mechanisms may be sufficient for tracking constant wind direction plumes. This corroborates results by Pang et al. [2018] and Grunebaum and Willis [2015] and extends them by highlighting the importance of longer term memory in cases where wind changes direction.

6 Limitations and future work

Our results motivate further development in using DRL to model and understand complex behaviors in several ways. First, here we used vanilla recurrent units with no biomechanical body model, and models that incorporate known complexity from biology as constraints may give rise to further insights. For instance, DRL agents may be trained using spiking neural networks [Jia et al., 2021; Recanatesi et al., 2019; Yuan et al., 2019]. Further, the wealth of architectural insights emerging from the fly connectome may be used to constrain wiring motifs in artificial networks [Scheffer et al., 2020]. A neuromechanical body model would enrich the interactions between the agent and the simulation environment [Merel et al., 2019; Plum and Labonte, 2021; Ríos et al., 2021].

Second, multi-task training should produce agents with richer behaviors and more complex neural activity structures with shared and task-specific adaptations [Crawshaw, 2020; Duncker et al., 2020; Mlynarski and Hermundstad, 2018; Weber and Fairhall, 2019; Yang et al., 2019]. Adding other sensory modalities like vision and training the agents in a 3D virtual reality environment could produce more realistic perceptual representations in the agent [Crosby, 2020; Crosby et al., 2019].

Finally, future work could explore learning algorithms that respect biological constraints like excitation-inhibition balance and Dale’s law [Delahunt et al., 2018; Ehrlich et al., 2021; Goulas et al., 2021]. More complex training curricula [Bengio et al., 2009] or alternative training algorithms using evolutionary techniques [Gupta et al., 2021; Stanley et al., 2019] might be able to mitigate the significant performance variability we observed in our agents (Section 4.3).

Our analyses also motivate further methodological development in theoretical tools to understand actor-critic RNNs. Currently available reverse-engineering methods that characterize RNNs using discrete dynamical features such as fixed-points [Maheswaranathan et al., 2019a,b; Sussillo and Barak, 2013] are not applicable to the continuous and amorphous dynamical structures that we encountered in our analyses (Section 4.5). New methods are also needed for comparing multiple agents at the behavioral level, specifically taking into account the compounding differences that arise from small differences in action-stimulus loops. Finally, further theoretical work is required to understand the role of training-induced unstable RNN connectivity eigenmodes, such as those observed in Section 4.7. Straightforward application of analysis techniques developed for RNNs trained using supervised-learning [Maheswaranathan et al., 2019a; Rajan and Abbott, 2006; Sussillo and Abbott, 2009] was not possible.

7 Conclusion

In this paper, we used deep reinforcement learning to train recurrent neural network agents to solve a stochastic plume tracking task. We find several behavioral and neural features that emerge in these trained agents and connect these features with how flying insects track turbulent plumes. Our findings motivate future experiments and theoretical developments, and provide a foundation for more nuanced future work. We hope our approach will contribute to the growing convergence in the understanding of artificial and biological networks [Hassabis et al., 2017; Hasson et al., 2020]. Efforts to reverse engineer such neural network agents will help accelerate the development of similar methods for biological agents [Ashwood et al., 2020; Kwon et al., 2020]. Moreover, our RNN agents may serve as generative models of complex naturalistic behaviors, which may facilitate the development of behavior analysis tools for biology [Berman et al., 2016; Nassar et al., 2018; Singh et al., 2021]. Insights from these studies may also inspire the development of robotic agents with artificial [Vouloutsi et al., 2013] or hybrid [Anderson et al., 2019] olfactory sensing.

Online supplement

Animations accompanying this manuscript can be found at: https://github.com/BruntonUWBio/plumetracknets. Code will be released at this location on manuscript publication.

Author Contributions

SHS, FvB, RPNR and BWB conceived of the study/analysis. SHS engineered the agents. SHS per-
formed the data analysis. SHS, FvB, RPNR, and BWB interpreted the results. SHS and BWB wrote the manuscript. All authors reviewed and edited the manuscript.

Acknowledgements

We thank Aravind Rajeswaran, Stefano Recanatesi, ScottSterrett, NiruMaheswaranathan, and SteveBrunton for helpful comments and discussions. The plume tracking task graphic in Figure 1 is inspired by a similar figure in Baker et al. [2018], and uses parts of an open source fruitfly graphic from scidraw.io [Costa, 2020]. We are grateful for the well-documented open source implementation of PPO by IlyaKostrikov [Kostrikov, 2021] which we heavily adapted and built upon for our work.

This work has been funded by the Air Force Research Lab award FA8651-20-1-0002 and FA9550-21-0122 (FvB); the National Institutes of Health award NIH P20GM103650 (FvB); the Defense Advanced Research Projects Agency HR001120C0021 (RPNR); the Templeton World Charity Foundation (RPNR); the National Science Foundation award ECC-1028725 (RPNR); the Air Force Office of Scientific Research awards FA9550-19-1-0386 and FA9550-18-1-0114 (BWB); and the Washington Research Foundation (BWB).

References

Appendix A Supplementary details on agent training and evaluation

Figure A.10: Snapshot of training plume: Plume (dull grey) originating at crosshairs (dotted black lines, bottom left). Current wind direction shown by arrow in dashed circle (bottom left). 50 randomly chosen initialization points (red) overlayed on plume. Agent is initialized with a uniformly randomly chosen head-direction at a random location near or on the plume. Wind direction switches by a random amount at random times as described in Section Appendix A.1

[We repeat some details from the main text for the sake of readability.]

Appendix A.1 Plume pre-computation:

Puffs are generated at the source located at (0, 0), at the rate of \( r_t \sim \text{Poisson}(R) \) puffs/step, where \( R = 1.0 \). Each puff’s location \( p_t = (x_t, y_t) \) is henceforth governed by the stochastic differential equation, \( p_t = p_{t-1} + w_t \delta + \xi \), where \( w_t \) is the wind-velocity at time \( t \), \( \xi \sim N(0, \sigma) \) is cross wind i.i.d. random Gaussian noise added per the turbulent plume model of [Farrell et al., 2002]. Each puff trajectory is integrated using a simple forward Euler integrator at 100 frames/sec. Furthermore, each puff starts with a radius \( r_0 = 0.01 \) m and undergoes a diffusion process that increases it’s radius at the rate of 0.01 m/s.

We compute 120-second long (clock time) plumes ahead of training/evaluation time for the ‘constant’, ‘switch-once’ and ‘switch-many’ configurations. A 40s window (60s - 100s) of the ‘switch-many’ and ‘constant’ plumes are used for training agents (Figure A.10). ‘Sparse’ plumes are simulated by downsampling the number of puffs simulated in a ‘constant’ plume simulation.

Wind velocity \( w_t \) is held constant at \((0.5, 0.0)\) m/s for all \( t \) for the ‘constant’ plume configuration. For the ‘switch-once’ plume configuration, wind velocity \( w_t \) starts at \((0.5, 0.0)\) m/s till \( t = 60.00 \) s, when it makes a single 45° counter-clockwise turn and stays there for the rest of the simulation. For the ‘switch-many’ plume configuration, wind velocity \( w_t \) changes once every \( 3.0 + \tau \) seconds, where \( \tau \sim \text{Uniform}(-0.3, +0.3) \) is a random shift added i.i.d. at each change. Wind direction turns are sampled i.i.d. from a \( N(0^\circ, 30^\circ) \) Gaussian distribution truncated at \( \pm 60^\circ \).

Appendix A.2 Training

Partially Observable Markov Decision Process (POMDP): To train agents using DRL, we define a Partially Observable Markov Decision Process (POMDP) [Sutton and Barto, 2018] as follows (also see Figure 1):

- **Action space:** Agents provide a two dimensional output \( a_t \) at each timestep corresponding to how much they want to turn and how much they want to move forward.

\[
a_t = [a_\theta, a_m], \quad \text{where } a_\theta \in [-\theta_{\text{max}}, +\theta_{\text{max}}], a_m \in [0, \Delta_{\text{max}}]
\]

The maximum turn capacity of an agent (\( \theta_{\text{max}} \)) is \( 6.25\pi \) radians/s (1125 °/s), and the maximum forward movement capacity (\( \Delta_{\text{max}} \)) of an agent is \( 2.5 \) m/s.

- **Observation space:** Agents receive a 3-dimensional egocentric sensory observation vector \( o_t \) at timestep \( t \), comprising odor-concentration and \((x, y)\) coordinates of relative wind-velocity at the agent’s current location and orientation in the plume. Note that the agent’s current location and orientation in the plume are tracked and updated by the training environment code.

\[
o_t = [o_c, o_x, o_y], \quad \text{where } o_c \in [c_{\text{min}}, c_{\text{max}}], o_x, o_y \in [-(\Delta_{\text{max}} + |v_{\text{wind}}|), (\Delta_{\text{max}} + |v_{\text{wind}}|)]
\]

Here \( c_{\text{min}} \) and \( c_{\text{max}} \) are the minimum and maximum perceivable odor concentrations, that have been manually set to be 0.0001 and 1.0 arbitrary units respectively.
• **Reward function:** Rewards are given to encourage task completion, i.e. home in on the plume source. The agent receives: +100 when it reaches within a small fixed radius \(r_{\text{homed}} = 0.2\text{m}\) of the source, \(-\epsilon\) per timestep to simulate a ‘metabolic cost’ to flying and therefore encourage faster homing. We also provide the agent two shaping rewards, without which the training process is infeasibly slow: First, a reward proportional to the decrease in radial distance to the source per timestep \((r_{t-1} - r_t)\) as a form of shaping reward. Here, \(r_t = \sqrt{x_t^2 + y_t^2}\) is the euclidean distance of the agent to the source at timestep \(t\). Second, a fixed negative reward of \(-10\) if the agent strays more than \(r_{\text{stray}} = 2\text{m}\) away from the plume (i.e. the center of the nearest puff is greater than \(r_{\text{stray}}\)).

• **Transition function:** The agent’s location and orientation within the plume is randomly initialized at the beginning of each training episode (see Figure A.10 for example locations). The environment then deterministically updates the agent’s location and orientation at each timestep taking into account its actions and the wind velocity. Episodes end if the agent reaches within a radial distance \(r_{\text{homed}}\) of odor source, or if the agent strays more than \(r_{\text{stray}}\) from the plume, or if the episode exceeds 300 timesteps (12 seconds of clock time).

• **Augmented observation space for MLPs:** To understand the role of memory on tracking performance, in Section 4.3, we use feedforward-only networks (MLPs) with fixed-length memory. Memory is simulated by appending historical sensory observations into the MLP’s inputs (known as ‘frame stacking’ in the DRL literature [Mnih et al., 2013]). Therefore \(o_L\) for an MLP with \(L\) timesteps history is now \([o_0^{(0)}, o_0^{(0)}, o_0^{(0)}, \ldots, o_L^{(2)}, o_L^{(2)}, o_L^{(2)}]\).

We implement the POMDP environment using the OpenAI Gym [Brockman et al., 2016] and stable-baselines [Hill et al., 2018] libraries.

**Training curricula:** We adapt an open source implementation [Kostrikov, 2021] of the Proximal Policy Gradient algorithm with Generalized Advantage Estimation (PPO-GAE) [Schulman et al., 2015, 2017] to train our agents.

To train our agents to perform across dynamically varying plumes, we randomize the agent’s location, agent’s orientation, plume state and plume sparsity at the start of each training episode. Agents are initialized at random starting locations \((x, y)\), where \(x\) is chosen uniformly randomly in the range \([30, 80]\) percentile of puff locations; \(y\) is chosen by sampling from a normal distribution with mean given by the median \(y\)-coordinate of odor puffs in the range \([x - 1, x + 1]\), and variance given by the \(5^\text{th} - 50^\text{th}\) percentile \(y\)-coordinate difference of the aforementioned odor puffs. Initial agent orientation is selected at random from \([-\pi, \pi]\) radians. The ‘switch-many’ plume, which changes direction every \(\approx 3\) seconds, is used for training. Initial plume state is randomized by choosing a random time between 60s - 90s, at which to initialize the precomputed plume. The simulation is sparsified by downsampling the number of puffs to a fraction randomly uniformly chosen in the range \([0.3, 1.0]\). The plume is randomly flipped about the \(x\)-axis to mitigate any \(y\)-directional biases that might have crept into the finite plume simulation.

Curriculum based training methods are known to improve training performance by gradually increasing the difficulty of the training task over the course of the training process [Bengio et al., 2009]. We train our RNNs using a two stage curriculum, where we first train the RNN for 1 million timesteps on the constant wind direction plume, and then train it for another 4 million timesteps on the ‘switch-many’ plume. This two stage process improved the stability and performance of the training process for RNNs, but not for MLPs. MLPs are directly trained for 2 million timesteps on the ‘switch-many’ plume. Training durations have been chosen such that training updates reliably converge within these times.

**Hyperparameter selection:** Our training process has hyperparameters relating to (1) training algorithm hyperparameters, (2) training plume parameters, and (3) neural network architecture. (See Appendix B for a list of all [hyper]parameters). While PPO is not a very sample efficient algorithm, it is known to work robustly across a wide range of continuous control (continuous observation and action space) problems without needing extensive hyperparameter tuning [Schulman et al., 2017]. Furthermore, exhaustive hyperparameter tuning is computationally unfeasible on our budget. However, we do try to tweak hyperparameters one-by-one starting off from the parameters suggested in the PPO manuscript for continuous control problems. We also trained Gated Recurrent Units (GRUs) in the same manner as we did our Vanilla RNNs (RNNs), and found that the performance of the GRUs did not significantly exceed that of the RNNs (see Figure A.11).

**Other shaping rewards explored:** Flying insects are known to exhibit a significant range of speeds [van Breugel and Dickinson, 2014]. However, our trained agents mostly fly at either their maximum speed or very slowly (see Figure 3). As additional reward shaping, we did try to add movement-related penalties to the reward function to induce some speed modulation, however, did not use these agents because of drastically worse performance compared to unpenalized agents. Future work could explore ways of skewing DRL reward functions towards such auxiliary goals that are not aligned with the primary plume tracking task.

**Computational resources:** All models are trained and evaluated on an Ubuntu Linux v20.04 workstation with Intel Core i9-9940X CPU and a TITAN RTX GPU. Each seed takes \(\approx 16\) hours to train and evaluate, with MLP and RNN models using 1 and 4 cores in parallel respectively.
The same set of 240 initial conditions for each episode are used to initialize the agent and simulator, for each agent and dataset evaluated:

- Initial agent head angle (with respect to ground): 0, $\frac{1}{4}\pi$, $\frac{1}{2}\pi$, $\frac{3}{4}\pi$, $\pi$, $\frac{5}{4}\pi$, $\frac{3}{2}\pi$, $\frac{7}{4}\pi$ radians
- Initial x-coordinate: 4, 6, and 8 meters
- Initial y-coordinate: $0^{th}$, $25^{th}$, $50^{th}$, $75^{th}$, and $100^{th}$ percentile of the minimum and maximum y-coordinate of the puffs located in a 1-meter band around the initial x-coordinate. For ‘constant’ wind direction plumes (including sparse plumes), the task is made harder by selecting only $0^{th}$, $50^{th}$, and $100^{th}$ percentiles as described before (i.e. $y_{\text{min}}$, $y_{\text{median}}$, $y_{\text{max}}$) and then adding two other locations that are $\pm0.5$ m outside the plume (i.e. $y_{\text{min}} - 0.5$ m and $y_{\text{max}} + 0.5$ m)
- Initial timestamp: 60.00s and 61.00s (58.00s and 59.00s for the ‘switch-once’ plume as it switches at exactly 60.00s)

**Agent selection:** We train 14 seeds per model type (RNNs, and MLPs with 2, 4, 6 ..., 12 timesteps of history) and select the top-5 best performing seeds for analysis. Performance here is measured by counting the number of successful episodes across ‘constant’, ‘switch-once’ and ‘switch-many’ plumes.

**Evaluation subset:** For our analyses in Sections 4.4 – 4.6, we use a randomly selected 120 episode evaluation subset of the 240 evaluation episodes for each of the constant, switch-once and switch-many plume configurations. The selected episodes are balanced to include an equal number (60 episodes each) of successful and unsuccessful plume tracking episode outcomes. Whenever there are fewer than 60 episodes of either outcome type (successful or unsuccessful) for any plume configuration, then the selection is trimmed to use an equal number of episodes of the smaller outcome type. PCAs tend to be sensitive to imbalances in the data, and this balancing process enables visualizations to be consistently compared across agents. In the analysis described in Section 4.3, we use all 240 evaluation episodes per agent.

![Figure A.11: Comparison of Vanilla RNNs and GRUs across 4 plume configurations. Vanilla RNN data is same as that in Figure 5](image)
Appendix B  Key parameters for simulation, agent, model, and training and evaluation

<table>
<thead>
<tr>
<th>Parameter description</th>
<th>Value/Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation integration time-step</td>
<td>0.01s</td>
</tr>
<tr>
<td>Wind speed</td>
<td>0.5 m/s</td>
</tr>
<tr>
<td>Wind speed crosswind noise</td>
<td>$\mathcal{N}(0, 0.005)$ m/s (per timestep)</td>
</tr>
<tr>
<td>Puff birth rate (Poisson mean)</td>
<td>1.0 pulls/timestep (at 100 FPS)</td>
</tr>
<tr>
<td>Puff initial radius</td>
<td>0.01m</td>
</tr>
<tr>
<td>Puff radius growth rate (diffusion)</td>
<td>0.01m/s</td>
</tr>
<tr>
<td>Maximum plume extent simulated (x, y)</td>
<td>(-2/+10m, ± 5m)</td>
</tr>
</tbody>
</table>

Table B.1: Plume parameters

<table>
<thead>
<tr>
<th>Parameter description</th>
<th>Value/Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment frame rate</td>
<td>25 FPS</td>
</tr>
<tr>
<td>Sensor sampling rate</td>
<td>25 Hz</td>
</tr>
<tr>
<td>Forward movement capacity ($\Delta_{max}$)</td>
<td>2.5 m/s</td>
</tr>
<tr>
<td>Turn capacity ($\theta_{max}$)</td>
<td>$\pm 6.25 \pi$ radians/s ($\pm 1125^\circ$/sec)</td>
</tr>
<tr>
<td>Homing radius</td>
<td>0.2 m</td>
</tr>
<tr>
<td>Max. stray from plume allowed</td>
<td>2 m</td>
</tr>
<tr>
<td>Odor sensing thresholds (minimum, maximum)</td>
<td>(0.0001, 1.0) (A.U.).</td>
</tr>
</tbody>
</table>

Table B.2: Agent and environment parameters

<table>
<thead>
<tr>
<th>Parameter description</th>
<th>Value/Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN hidden layer width</td>
<td>64 units</td>
</tr>
<tr>
<td>Feedforward hidden layer width(s)</td>
<td>64 units</td>
</tr>
<tr>
<td>Neural network nonlinearity</td>
<td>tanh</td>
</tr>
<tr>
<td>Layer initialization (Recurrent, Feedforward)</td>
<td>(Normal, Orthogonal)</td>
</tr>
</tbody>
</table>

Table B.3: Model (neural network) parameters

<table>
<thead>
<tr>
<th>Parameter description</th>
<th>Value/Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN training steps</td>
<td>5M</td>
</tr>
<tr>
<td>MLP training steps</td>
<td>2M</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0.0003 (with linear decay)</td>
</tr>
<tr>
<td>PPO Entropy Coefficient</td>
<td>0.05</td>
</tr>
<tr>
<td>PPO Value Loss Coefficient</td>
<td>0.5</td>
</tr>
<tr>
<td>PPO Epochs</td>
<td>10</td>
</tr>
<tr>
<td>PPO Gamma</td>
<td>0.99</td>
</tr>
<tr>
<td>PPO max. gradient norm</td>
<td>0.5</td>
</tr>
<tr>
<td>GAE Lambda</td>
<td>0.95</td>
</tr>
<tr>
<td>GAE steps</td>
<td>2048</td>
</tr>
</tbody>
</table>

Table B.4: Training algorithm, training curriculum and evaluation parameters
Appendix C Behavior module metadata distributions

<table>
<thead>
<tr>
<th>Agent</th>
<th>Agent ID</th>
<th>Lost module threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN 1</td>
<td>2760377</td>
<td>30 steps (1.2 s)</td>
</tr>
<tr>
<td>RNN 2</td>
<td>3199993</td>
<td>25 steps (1.0 s)</td>
</tr>
<tr>
<td>RNN 3</td>
<td>3307e9</td>
<td>35 steps (1.4 s)</td>
</tr>
<tr>
<td>RNN 4</td>
<td>541058</td>
<td>38 steps (1.52 s)</td>
</tr>
<tr>
<td>RNN 5</td>
<td>9781ba</td>
<td>25 steps (1.0 s)</td>
</tr>
</tbody>
</table>

Table C.5: Thresholds for defining when the lost behavior module kicks in i.e. duration (in timesteps or seconds) since the plume was last encountered.

Figure C.12: Behavior modules - Agent 1 (See Figure 3 for equivalent data on Agent 3 and figure details)

Figure C.13: Behavior modules - Agent 2
Figure C.14: Behavior modules - Agent 3 (same as Figure 3)

Figure C.15: Behavior modules - Agent 4

Figure C.16: Behavior modules - Agent 5
Appendix D Comparing reference frames for plume tracking

Figure D.17: Empirical course-direction (CD) distribution - Agent 1 (See Figure 4 for equivalent data on Agent 3 and figure details)

Figure D.18: Empirical course-direction (CD) distribution - Agent 2

Figure D.19: Empirical course-direction (CD) distribution - Agent 3 (Same as Figure 4)
Figure D.20: Empirical course-direction (CD) distribution - Agent 4

Figure D.21: Empirical course-direction (CD) distribution - Agent 5
Appendix E Neural activity dimensionality and neural representations

**Odor encounters:** Our definition of odor encounters is identical to that used in Demir et al. [2020]. The stream of odor inputs is discretized to be 1 at the first timestep of the stream where the odor is perceptible and 0 for the remaining contiguous steps where it is still perceptible. Other processing is as described in Section 4.4.

<table>
<thead>
<tr>
<th>Agent</th>
<th>Agent ID</th>
<th>odor\textsubscript{EWMA} window [steps]</th>
<th>odor\textsubscript{EWMA} (R^2)</th>
<th>odor\textsubscript{ENC} window [steps]</th>
<th>odor\textsubscript{ENC} (R^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN 1</td>
<td>2760377</td>
<td>8</td>
<td>0.91</td>
<td>62</td>
<td>0.57</td>
</tr>
<tr>
<td>RNN 2</td>
<td>3199993</td>
<td>10</td>
<td>0.86</td>
<td>44</td>
<td>0.71</td>
</tr>
<tr>
<td>RNN 3</td>
<td>3307e9</td>
<td>8</td>
<td>0.92</td>
<td>46</td>
<td>0.57</td>
</tr>
<tr>
<td>RNN 4</td>
<td>541058</td>
<td>6</td>
<td>0.88</td>
<td>40</td>
<td>0.51</td>
</tr>
<tr>
<td>RNN 5</td>
<td>9781ba</td>
<td>12</td>
<td>0.91</td>
<td>44</td>
<td>0.59</td>
</tr>
</tbody>
</table>

Table E.6: Moving window lengths and linear regression fit \(R^2\) for two represented variables: odor\textsubscript{EWMA} and odor\textsubscript{ENC}. See Section 4.4 for further details. (Recall that 25 timesteps = 1.0 second).

**Agent action classifier:** To quantify how important these represented variables are to actual task performance, we train a Random Forest (RF) [Breiman, 2001] classifier to predict actions taken by the agent over successful trajectories. We uniformly partition the Turn and Move action variable, which are continuous valued, into domains of 3 and 2 discrete classes respectively. These classes correspond roughly to ‘left’, ‘center’ and ‘right’ turns, and to ‘fast’ and ‘slow’ forward movements. These are concatenated to form a 6-class independent variable. The classifier receives instantaneous sensory observations (egocentric wind speed \(w_X, w_Y\) and odor concentration) and the four aforementioned encoded features as inputs. Training and test sets are a randomized non-overlapping 80%-20% split of evaluation episodes, balanced across plume configuration and episode outcomes. We do a 20-trial 3-fold cross-validated randomized search over the number-of-estimators (range: [10,50]) hyperparameter, and then train a classifier using the best hyperparameter on the whole training set. We next estimate the relative importance of each input feature by calculating its permutation importance score [Breiman, 2001; Strobl et al., 2008], which is an estimate of the reduction in the classifier’s accuracy across several (\(N=30\)) randomized permutations of that feature. Note again that the estimates provided by this analysis are approximate due to the discretization of the action data and correlations between features.

<table>
<thead>
<tr>
<th>Agent</th>
<th>Agent ID</th>
<th>Test set accuracy (All features)</th>
<th>Test set accuracy (Instantaneous only)</th>
<th>Test set accuracy (Most freq. class)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN 1</td>
<td>2760377</td>
<td>0.84</td>
<td>0.74 (0.10)</td>
<td>0.33 (0.51)</td>
</tr>
<tr>
<td>RNN 2</td>
<td>3199993</td>
<td>0.67</td>
<td>0.49 (0.18)</td>
<td>0.28 (0.39)</td>
</tr>
<tr>
<td>RNN 3</td>
<td>3307e9</td>
<td>0.82</td>
<td>0.69 (0.13)</td>
<td>0.39 (0.43)</td>
</tr>
<tr>
<td>RNN 4</td>
<td>541058</td>
<td>0.70</td>
<td>0.53 (0.17)</td>
<td>0.44 (0.26)</td>
</tr>
<tr>
<td>RNN 5</td>
<td>9781ba</td>
<td>0.84</td>
<td>0.74 (0.10)</td>
<td>0.40 (0.44)</td>
</tr>
</tbody>
</table>

Table E.7: Classifier based quantification of contribution of represented features: In last two columns, quantity in parentheses is the difference in accuracy with respect to classifier that has all features (4 represented features and instantaneous features). Represented features contribute to higher test accuracy. See Section 4.4 for further details.
Figure E.22: Neural representations – Agent 1 (See Figure 6 for equivalent data on Agent 3 and figure details)
Figure E.23: Neural representations – Agent 2
Figure E.24: Neural representations – Agent 3 (Same agent as in Figure 6)
Figure E.25: Neural representations – Agent 4
Figure E.26: Neural representations – Agent 5
Appendix F  Structured neural dynamics

<table>
<thead>
<tr>
<th>Agent</th>
<th>Agent ID</th>
<th>Limit-cycle period</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN 1</td>
<td>2760377</td>
<td>19 steps (0.76 s)</td>
</tr>
<tr>
<td>RNN 2</td>
<td>3199993</td>
<td>NA (clear periodic structure not observed)</td>
</tr>
<tr>
<td>RNN 3</td>
<td>3307e9</td>
<td>17 steps (0.68 s)</td>
</tr>
<tr>
<td>RNN 4</td>
<td>541058</td>
<td>28 steps (1.12 s)</td>
</tr>
<tr>
<td>RNN 5</td>
<td>9781ba</td>
<td>18 steps (0.72 s)</td>
</tr>
</tbody>
</table>

Table F.8: Limit cycle periods for each RNN agent

Figure F.27: Neural dynamics – Agent 1 (See Figure 7 for equivalent data on Agent 3 and figure details)
Figure F.28: Neural dynamics – Agent 2
Figure F.29: Neural dynamics – Agent 3 (same as Figure 7)
Figure F.30: Neural dynamics – Agent 4
Appendix G Transitions between neural clusters

Transition duration calculations: To define entry into a neural activity regime, we first define neural activity ‘centroids’ associated with the tracking and lost neural activity regimes. These are the average of the last 1-second’s neural activity from successful evaluation episodes that home in on odor source (HOME), and unsuccessful evaluation episodes where the agent straying Out Of (arena) Bounds (OOB), respectively (see Figure 8). We then define neural activity clusters associated with the HOME and OOB centroids as being comprised of all neural activity with a distance $D/2$ units from the respective centroid, where $D$ is the distance between centroids. Finally, for any unsuccessful tracking episode, we calculate a ‘time to lost’ (TTL) as the duration between the agent leaving the plume and entering the OOB cluster. Similarly, for successfully homing episodes, we calculate a ‘time to track’ (TTT) as the time taken to enter the HOME cluster after entering the plume. In calculating TTT, we exclude small excursions outside the plume where the agent is skimming the boundary of the plume and only consider excursions where the agent has entered the recovering or lost behavioral module. We split TTT into two types, labeling it ‘time to track not lost’ (TTT-NL) if the agent was in recovering or ‘time to track after lost’ (TTT-L) if the agent was in lost before entering the plume.

Statistical significance calculations: All plots use the Mann-Whitney-Wilcoxon test two-sided with Bonferroni correction, where p-value annotations indicate:

- ns: $5.00e-02 < p \leq 1.00e+00$ (not significant)
- *: $1.00e-02 < p \leq 5.00e-02$
- **: $1.00e-03 < p \leq 1.00e-02$
- ***: $1.00e-04 < p \leq 1.00e-03$
- ****: $p \leq 1.00e-04$

Figure G.32: Transitions between neural activity regimes – Agent 1 (compare with Agent 3 in Figure 8)

Figure G.33: Transitions between neural activity regimes – Agent 2 (NB: this agent does not follow the trend)
Figure G.34: Transitions between neural activity regimes – Agent 3 (same data as Figure 8)

Figure G.35: Transitions between neural activity regimes – Agent 4

Figure G.36: Transitions between neural activity regimes – Agent 5
Appendix H  RNN connectivity and stimulus integration timescales

Stimulus integration timescale $\tau_i$ calculation: (First see background on RNNs provided in Section 4.7). Prior literature has looked at the eigenvalues and eigenvectors of the recurrence Jacobian (and recurrence matrix) to investigate how connectivity affects the dynamics of the network [Maheswaranathan et al., 2019a; Rajan and Abbott, 2006]. Specifically [Maheswaranathan et al., 2019a] obtains the stimulus integration timescale $\tau_i$ associated with a stable eigenvalue $\lambda_i$ (i.e. $|\lambda_i| \leq 1$), by looking at the discrete-time iteration $h_i(t) = \lambda_i^t h_i(0)$ that governs the integration of stimulus in the direction of eigenvector $v_i$ associated with $\lambda_i$. They then compare this with the equivalent continuous time equation $h_i(t) = h_i(0)e^{-t/\tau_i}$, to get $\tau_i = \left| \frac{1}{\ln|\lambda_i|} \right|$.

<table>
<thead>
<tr>
<th>Agent</th>
<th>Agent ID</th>
<th>Top 5 $\tau$s</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN 1</td>
<td>2760377</td>
<td>116.5, 81.5, 16.9, 13.5, 8.3</td>
</tr>
<tr>
<td>RNN 2</td>
<td>3199993</td>
<td>95.7, 61.7, 16.6, 12.0, 9.6</td>
</tr>
<tr>
<td>RNN 3</td>
<td>3307e9</td>
<td>56.5, 13.0, 7.7, 6.8, 5.8</td>
</tr>
<tr>
<td>RNN 4</td>
<td>541058</td>
<td>86.4, 51.8, 15.1, 12.4, 9.7</td>
</tr>
<tr>
<td>RNN 5</td>
<td>9781ba</td>
<td>86.2, 27.4, 8.6, 6.6, 5.6</td>
</tr>
</tbody>
</table>

Table H.9: Top 5 $\tau$s (stimulus integration timescales) for each RNN seed

Figure H.37: Eigenspectra of $W_h$ before and after training, and stimulus integration timescales – Agent 1 (compare with Agent 3 in Figure 9)

Figure H.38: Eigenspectra of $W_h$ before and after training, and stimulus integration timescales – Agent 2
Figure H.39: Eigenspectra of $W_h$ before and after training, and stimulus integration timescales – Agent 3 (same as Figure 9)

Figure H.40: Eigenspectra of $W_h$ before and after training, and stimulus integration timescales – Agent 4

Figure H.41: Eigenspectra of $W_h$ before and after training, and stimulus integration timescales – Agent 5