Scalar source tracking in turbulent environments using deep reinforcement learning

By V. Spandan†, P. Bharadwaj, M. Bassenne AND L. Jofre

Sensory-guided navigation in turbulent environments is extremely challenging owing to the highly chaotic and intermittent nature of the underlying flow. We investigate the use of deep reinforcement learning in turbulent flow environments to train point swimmers to navigate and localize the source of dispersed scalars. The reinforcement learning algorithms are adapted from a deep Q-network implementation from the OpenAI Baselines framework. We find that the strategies learned by the swimmer within this framework to navigate unsteady vortex streets and jets in a turbulent channel flow remain robust with changes in initial conditions of the swimmer and perturbations during the testing phase. Such a framework can be useful for developing strategies for unmanned robots navigating complex flow environments and is a new pathway for extracting a new fundamental understanding of flow physics. The latter is challenging given the dependence of the learned strategies on the free parameters within reinforcement learning.

1. Introduction

Nature is filled with several examples of organisms navigating complex and chaotic flow environments over a wide range of length scales and timescales to achieve important life-supporting tasks. For example, swimming microorganisms use physical and chemical stimuli from their surrounding environment to guide their motion toward nutrient-rich or shear-free regions (Fenchel 2002); by aggregating imperfect information from individuals, schools of fish display traits of collective intelligence in navigation (Berdahl *et al.* 2013); migrating birds or gliders use thermal plumes in the atmosphere to gain height with minimal use of propulsion (Cone 1962). Although the final objectives of navigation/migration in the above examples might be different from each other, at a fundamental level the process involves continuously gathering local information (such as physical or chemical cues from the surrounding flow) and undertaking suitable actions that help accomplish the long-term goal. Of the examples discussed above, navigation/migration based on odor, i.e., olfaction, has received particular interest. Several challenges are associated with odor guided navigation. In many cases, odors or pheromones are dispersed in a highly turbulent environment and thus gradient-ascent often fails. An additional challenge is that a flow environment is typically seeded with several types of odors, thus making it a highly dimensional system. Furthermore, in particular for odor-guided navigation in animals (e.g., moths, dogs), owing to the chaotic nature of the underlying flow, the local odor information is highly intermittent.

One well-known search strategy that does not use information about gradients is termed infotaxis, where the underlying idea of the strategy is that the searcher tries to locally maximize the expected rate of information gain (Vergassola *et al.* 2007). Within a flow environment seeded with odor plume propagation, it has been shown that such

† School of Engineering and Applied Sciences, Harvard University

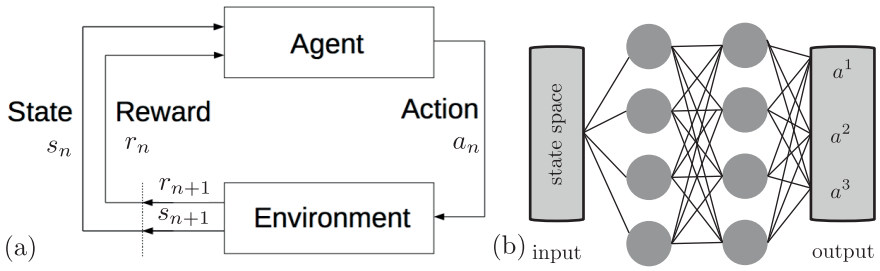

FIGURE 1. (a) A basic schematic of the reinforcement learning algorithm. Figure adapted from Sutton (1998). (b) A schematic of the neural network for the action-value map. The left most layer is the input layer, which contains a set of observations from the swimmer, and the output layer contains the value functions for each of the actions available to the swimmer.

a strategy can lead to zigzagging and casting paths similar to those seen in the flight of moths. However, the formulation of the strategy assumes an idealized statistical description of the odor plume propagation, whereas in reality, flows may present much more complex nature. Another class of studies have focused on building dynamical system models for the propagation of odor plumes or the flow itself and then computing the optimal trajectories to navigate such environments (Marques *et al.* 2006; Peng *et al.* 2015; Song & Mohseni 2015). Studies focused on understanding olfactory strategies in different animals are highly system dependent and do not span the wide phase space of the flow environment [e.g., crabs (Weissburg & Zimmer-Faust 1994), sharks (Rygg 2014), dogs (Craven *et al.* 2009), crustaceans (Koehl 2010)]. In this work, we explore the coupling of direct numerical simulations (DNS) of the Navier-Stokes (NS) equations and reinforcement learning (RL) and its suitability to study odor-based navigation strategies within a turbulent environment. Such a numerical framework allows the possibility to not only simulate realistic flows where the swimmers can actively modify the flow, but also possibly look for generalised search strategies based on cues from the underlying flow.

Reinforcement learning is based on continuous interaction between an agent (an active particle or a swimmer) and the environment (local flow conditions), during which the agent learns how to behave optimally (corresponding to a long-term goal such as maximum distance covered, minimum energy spent or finding source of an odor) by accumulating experience. A general framework of the RL algorithm is shown in Figure 1 (Sutton 1998). At any instant, the agent has the ability to sense a predefined set of quantities (s_n) from its environment and can choose from a predefined set of actions (a_n). The action leads to the agent being transported into a new state along with a given reward (r_{n+1}); the reward is a quantifiable measure of the local success of the previously chosen action to achieve the predefined globally set goal. Depending on the reward the agent receives, the policy of choosing a particular action for a given state is updated. When this loop is iterated several times, an approximately optimal policy is computed.

Although RL has been known and used for a long time in the fields of computer science and behaviorist psychology, its emergence in fluid dynamics is very recent. A series of recent studies demonstrated that RL can be used effectively in fluid dynamics to generate a framework of strategies for point-like microswimmers trying to maximize altitude, escape fluid traps, or navigate to specific regions in cellular flows such as Taylor-Green and Arnold-Beltrami-Childress flows (Colabrese *et al.* 2017; Gustavsson *et al.* 2017). In these studies, the RL framework employs a Q-learning approach where the training

phase involves constructing a quality matrix $Q(s, a)$ that quantifies the maximal return that can be achieved given a state s and action a. In such an approach, the state space (formally defined later) needs to be discretized in an ad hoc manner. More recently, high-fidelity simulations coupling NS and RL where the Q-matrix is now represented through a neural network have been used to show that in silico fishes can learn to follow a leader-fish while optimizing their thrust to cost ratio (Verma *et al.* 2018). In this work, we utilise the OpenAI RL framework (Brockman *et al.* 2016; Dhariwal *et al.* 2017) to explore whether the agent (swimmer) can train itself to navigate to the source of a scalar being dispersed in a turbulent environment using only locally available intermittent information. In Section 2, we briefly detail the numerical techniques used for the flow simulation and the learning algorithms. Finally, in section 3, we show results from training a swimmer in an unsteady vortex street and a cross-jet in a turbulent channel flow at $Re_\tau = 180$.

2. Formulation and computational setup

2.1. *Flow simulation*

For the fluid phase we solve the incompressible flow Navier-Stokes equations in a Cartesian box. A conservative second-order centered finite-difference scheme with velocities on a staggered grid is used for spatial discretization; explicit Adams-Bashforth scheme is used to discretize the nonlinear terms and an implicit Crank-Nicolson scheme is used for the viscous terms. Treating all the viscous terms implicitly results in a large sparse matrix that is avoided by an approximate factorization of the sparse matrix into three tridiagonal matrices (one for each direction), which are solved using the tridiagonal matrix algorithm. Time integration is performed via a self-starting fractional step third-order Runge-Kutta (RK3) scheme. The pressure required to enforce mass conservation is computed by solving a Poisson equation for a pressure correction. The code has already been tested extensively in previous studies for a variety of flow configurations, and additional details of the numerical scheme can be found in Rai & Moin (1991), Verzicco & Orlandi (1996), van der Poel *et al.* (2015), Spandan *et al.* (2017), and Spandan *et al.* (2018). Additionally, we solve an advection-diffusion equation for the transport of a passive scalar.

2.2. *Learning algorithm*

We now introduce the RL algorithm by first defining the variables used in the mathematical formulation. Boldface symbols represent sets, while lowercase symbols represent individual quantities.

An agent is the entity that is trying to achieve a certain goal. An action ($\mathbf{A}$) is a set of all possible moves the agent can make. An environment is the world sensed by the agent (flow environment here). A state ($\mathbf{S}$) is a set of all entities that the agent can sense. The reward ($\mathbf{R}$) defines the goal and is an entity provided by the environment to the agent each time step. The policy (Π) defines the agent's way of behaving at a given point of time. The exploration-factor (ϵ) sets the fraction of exploratory moves allowed. The discount factor (γ) determines the present value of future rewards, i.e. a reward received n time steps in the future is only γ^{n-1} times what it would be worth if it were received immediately. If $\gamma = 0$, the agent is concerned only with maximizing immediate rewards. The value of a state s, given a policy Π, is the expected return when starting in s and following Π.

As mentioned above, in RL an agent (in our case a smart swimmer) learns to earn rewards through trial and error within a simulated environment. The overall goal of

the agent is to learn an optimal policy, Π, which maximizes the action-value function, $Q^*(s_n, a_n)$, which is the sum of discounted future rewards defined as

$$Q^*(s_n, a_n) = \max \mathbf{E}(r_{n+1} + \gamma r_{n+2} + \gamma^2 r_{n+3} + ... | a_m = \Pi(s_m) \forall m \in [n+1, T]). \quad (2.1)$$

Here, T represents the end state of a training simulation; i.e., it can occur either if the agent has taken the maximum number of allocated steps or if it reaches the goal within the maximum allowable time steps. The discount factor γ is set to 0.9; this is an empirical value selected on the basis of previous works employing RL (Sutton 1998). Additional studies can be performed to study in detail the effect of the discount factor. In simple Q-networks, the state space and the action-value functions are discretized in an ad hoc manner in which the agent senses the environment in blocks. Additional complexity and generality can be achieved by representing the action-value function using a neural network with weights that are updated iteratively to minimize the temporal difference error.

In this work, we use the OpenAI implementation of the deep RL algorithms within the OpenAI gym and Baselines framework (Brockman *et al.* 2016; Dhariwal *et al.* 2017). The OpenAI gym is a toolkit for developing and comparing RL algorithms. On the other hand, Baselines is a set of high-quality implementations of various RL algorithms. Within Baselines, we use the DQN framework to train our agent in flow environments. Because we consider the agent to not have any effect on the flow, precomputed flow snapshots act as the environment in the gym and Baselines framework. Within DQN, we use two linear layers with 64 multilayer perceptrons each for the neural network representing the action value function. Every simulation is done in three phases: (i) the flow snapshots are generated; (ii) flow snapshots are used as a training environment for a given state space, action space, rewards; and a goal (iii) the agent (swimmer) is tested using the learned action-value map.

3. Results

3.1. *Unsteady laminar flow*

We first test the learning framework in an unsteady vortex street where the goal of a point swimmer is to reach the cylinders while gathering vorticity in its path. The swimmer does not differentiate between positive and negative vorticity, ω^2, and is rewarded in terms of its relative distance from the goal. The state space in these simulations is the local fluid velocity, and the action space available to the swimmer is to move unit distance in the directions shown in Figure 2. A single episode of the swimmer is defined in such a way that given an initial position, the swimmer is allowed to make a finite number of actions to achieve its predefined goal of reaching the cylinder while accumulating vorticity in its path. Once the episode ends, the swimmer is reinitialized to its starting position and the training continues.

In Figure 2, we show snapshots of the flow and the swimmer at different time instants of the testing phase. It is important to note that during the testing phase, the swimmer receives information on the local velocity and then performs actions purely on the basis of the action-value map learned during the training phase. When the swimmer is not rewarded using vorticity during the training phase, it quickly trains itself to move toward the goal in a straight path, which is a trivial solution learned during the training. On including vorticity in the reward function, the swimmer learns a different strategy wherein it oscillates according to the frequency of the incoming vortex street. Here, we also find

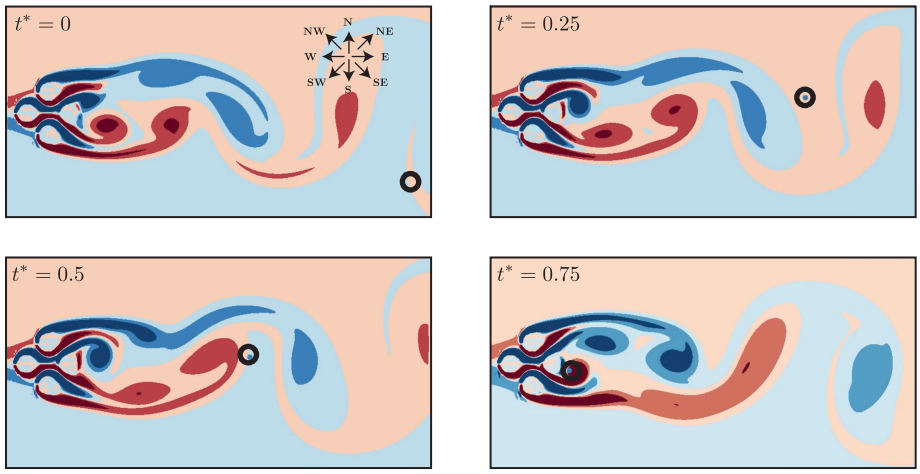

FIGURE 2. Instantaneous snapshots of the swimmer (black hollow circle) during the testing phase. Underlying environment is an uniform flow stream (left to right) at a cylinder Reynolds number of 100. The arrows in the top left panel show the actions available to the swimmer. $t*$ is the time since start of the testing episode normalized by the total time taken to reach the goal.

that the learned action-value function is robust to changes in initial position of the swimmer. An important parameter during the training phase is the exploration fraction, (ϵ), of the swimmer. By setting a finite value for the exploration fraction ($\epsilon = 0.4$), we find that the swimmer performs much better than a swimmer training with a perfectly greedy policy (this is well known in RL theory). Additionally during the testing phase, we force the swimmer to take a less optimal action by numerically kicking it out of its optimal path, and we find that the swimmer still manages to find its way back to its intended path toward the goal. This finding demonstrates the robustness of the learning. In Figure 3, we show how the accumulated reward increases with the number of episodes in the training phase. This shows that the swimmer is learning an increasingly optimal action-value map as the number of training episodes increases. As is observed, toward the end, the total accumulated rewards saturate when the training is stopped and the learned action-value map is saved for the testing phase.

3.2. *Jets in cross flow*

We now simulate a turbulent channel flow at $Re_\tau = 180$ with a jet in the wall-normal direction ($\hat{\mathbf{e}}_\mathbf{z}$) (Mahesh 2013). The ratio of the jet velocity to the mean stream-wise ($\hat{\mathbf{e}}_\mathbf{y}$) velocity is set to 4.0. Additionally, we initialize a passive scalar source at the location of the jet inlet. While the boundary conditions for velocity are periodic in the streamwise and spanwise directions, an outflow boundary condition is implemented for the passive scalar in the streamwise direction. In Figure 4(a), we show an instantaneous snapshot of the isosurface of the passive scalar in the turbulent channel. Figure 4(b) shows the dispersion of the scalar at different time instants in the mid-spanwise ($\hat{\mathbf{e}}_\mathbf{x}$) plane. In figure 4(c), we show a time-series of the fluctuations in the scalar concentration, wall-normal velocity (v_z), and streamwise velocity (v_y) where one can observe intense fluctuations in all quantities. The goal of these simulations is to train a swimmer that can access only intermittent local flow information, i.e., scalar concentration and flow velocity, and take actions in order to track down the source of the advected scalar.

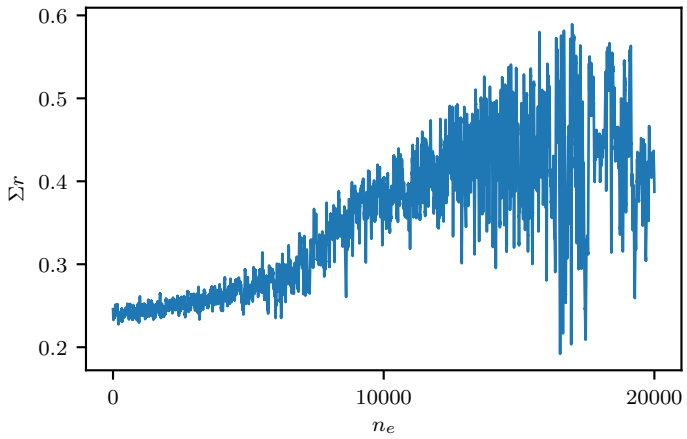

FIGURE 3. Total accumulated reward per episode versus the number of episodes during the training phase.

To reduce computational costs, we restrict the swimmer to remain in the mid-spanwise plane and freely navigate in the streamwise and wallnormal directions. Given this constraint, we test the learning performance of the swimmer with different state space, action space and exploration fraction and its effect on the ability of the swimmer to navigate successfully to the source of the jet. In Figure 5, we show the different trajectories that the swimmer undertakes during the testing phase; the difference among all cases is the strategy $Q(s, a)$ learned during training. We first look at the effect of different actions on the performance of the swimmer. In Figure 5(a), the swimmer is allowed to take steps in directions orthogonal to the wall-normal and stream-wise directions (N, S, E, W c.f. Figure 2) while in Figure 5(b), the swimmer is also allowed to move across diagonals (NE, SE, NW, SW c.f. Figure 2). From Figure 5(a), we observe that the swimmer fails to reach the source of the jet and gets stuck wandering near its initial position. On also allowing diagonal motion, we find that the swimmer is now able to navigate itself toward the source of the jet. The difference between the left panels and right panels in Figure 5 is the exploration fraction that the swimmer is allowed; left panels correspond to $\epsilon = 0.1$, while right panels correspond to $\epsilon = 0.4$. In Figure 5(a), orthogonal steps turn out to be too slow in comparison to the timescale of the advection of the flow in the stream-wise direction which leads to deteriorated learning. Individual diagonal steps are slightly longer than individual orthogonal steps and thus the relative timescales of motion in comparison to the advection timescale turns out to be a relevant parameter in the learning framework. In Figure 5(c), we also include the wall-normal velocity into the state space of the swimmer and the swimmer seems to perform even better in this configuration. This is a consequence of the availability of additional flow information in comparison with previous cases which helps the swimmer sample the flow in a more efficient manner.

We have attempted to train the swimmer in three-dimensional flow environments; however, given the huge phase space that needs to be covered to achieve a statistically significant action-value map, the swimmer could not learn successful strategies within reasonable time. Large-scale dedicated simulations will help to improve the learning in three-dimensional environments. Here, it is important to note that when the state-space,

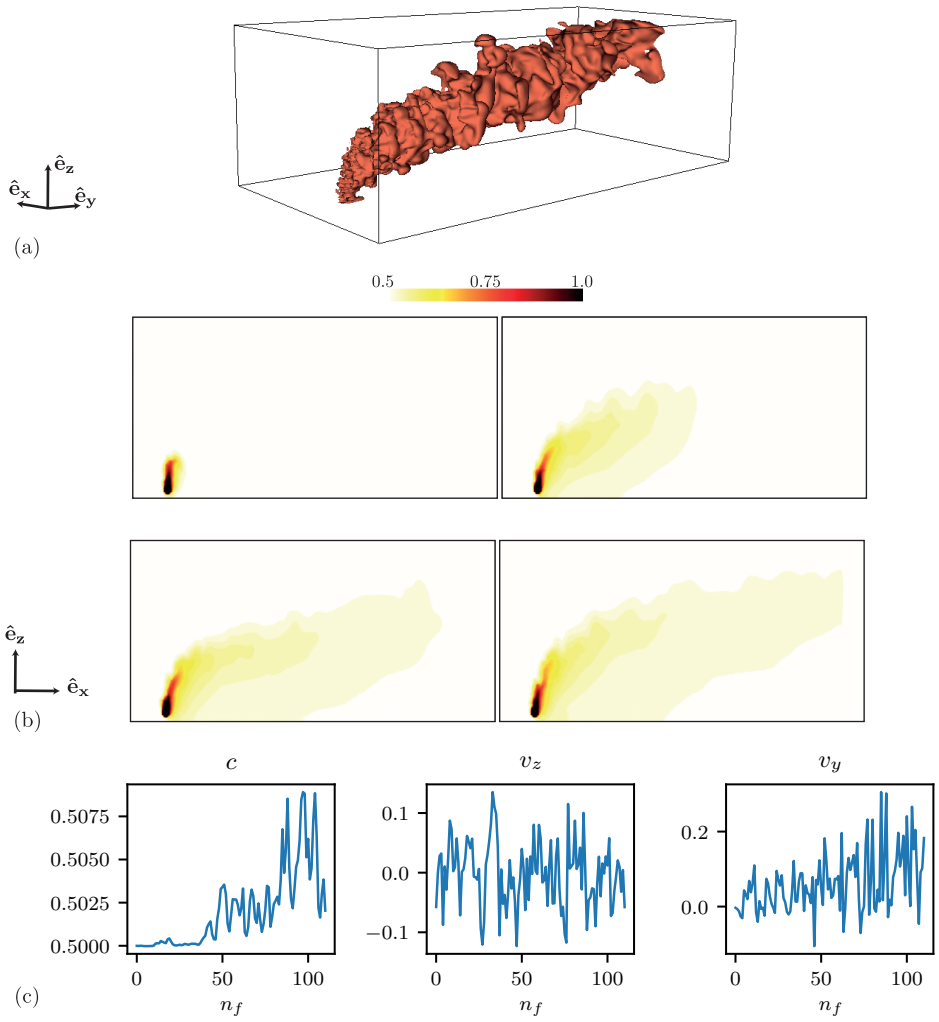

FIGURE 4. (a) Isosurface of a passive scalar being dispersed in a jet in cross flow. Friction Reynolds number of the underlying cross flow is set to $Re_\tau = 180$. (b) Instantaneous snapshots of the scalar concentration in the mid-spanwise place at different time instants. (c) Time series of the scalar concentration, wall-normal velocity, and streamwise velocity. n_f is the number of unique frames used for training and testing.

action-space and episode length are restricted, reasonable training can still be performed in three-dimensional environments. In the current work, although the swimmer does learn to navigate to the source of the jet in a highly turbulent environment with intermittent information, analyzing underlying flow physics from the learned strategies seems to be far from trivial in these simulations. Free parameters such as the exploration fraction, learning rate and discount factor influence the learned strategies, and these have to be taken into account in any analysis of the flow physics. On the other hand, coupling NS with RL seems to provide a robust framework for designing unmanned robots that can actively sense and navigate flow environments. The successful strategies also turn out

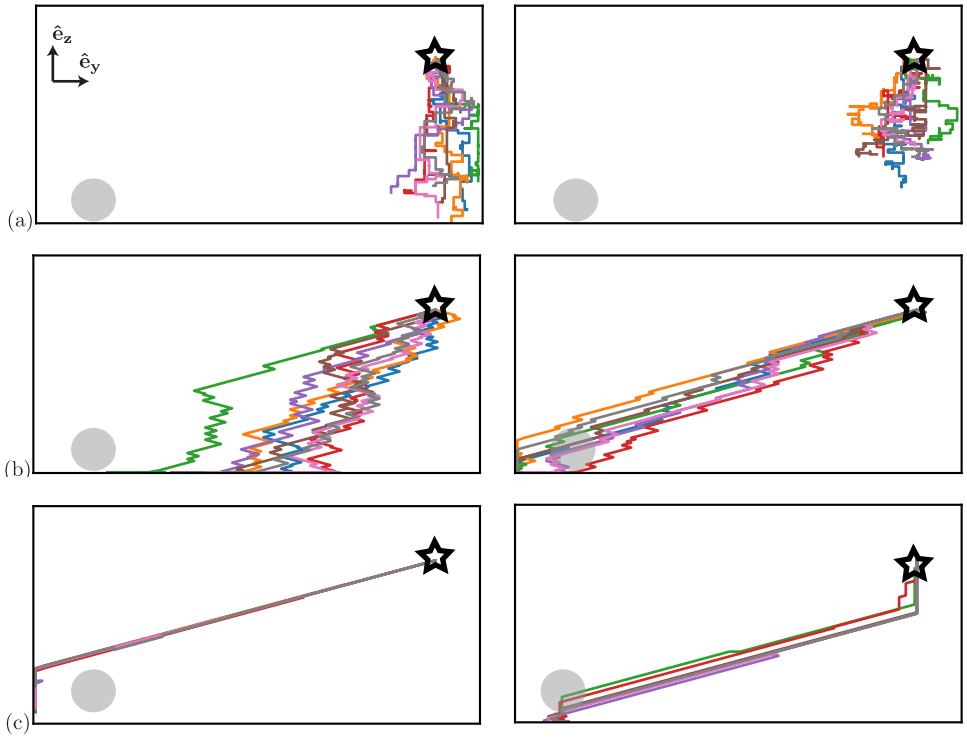

FIGURE 5. Trajectories of the swimmer during testing phase; lines represent different test cases using the same learned strategy but different perturbations. Initial position of the swimmer is marked by a start and the goal of the swimmer (source of the scalar) is marked by the gray circle.

to be robust to changes in the initial conditions of the swimmer. Flow environments at scales much larger to those considered in this work can be generated by large-eddy simulations and coupling them with RL can prove beneficial in field studies of underwater localization.

4. Conclusions

We have used the OpenAI Baselines framework for RL to train a point swimmer to navigate complex unsteady flows. In particular, we look at two specific flow environments: (i) unsteady vortex street, where the goal of the swimmer is to navigate towards upstream obstacles while passing through regions of intense vorticity; and (ii) jet in cross-flow with a passive scalar, where the goal is to navigate toward the source of the scalar on only local flow information and an arbitrary initial position downstream of the source. For the neural network representing the action-value map, we use two linear layers with 64 multilayer perceptrons each. We find that the strategy learned during the training is robust to noise during the testing phase and changes in the initial condition. In the case of the swimmer navigating to the source of the scalar, we find that the timescale of motion of the swimmer relative to the advection timescale is important during training to learn robust strategies. Here, it is important to note that the swimmer is able to learn multiple strategies during

training depending on the learning parameters which eventually lead to achieving the set goal. Thus, gaining insight into the underlying flow physics of navigation and tracking based on the learned action-value map is highly nontrivial, and more work is warranted along this front. Combining flow simulations at different scales with deep reinforcement learning may provide a framework for source localization for robotic applications.

REFERENCES

Berdahl, A., Torney, C. J., Ioannou, C. C., Faria, J. J. & Couzin, I. D. 2013 Emergent sensing of complex environments by mobile animal groups. *Science* **339**, 574–576.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. & Zaremba, W. 2016 OpenAI Gym. *ArXiv preprint* 1606.01540.

Colabrese, S., Gustavsson, K., Celani, A. & Biferale, L. 2017 Flow navigation by smart microswimmers via reinforcement learning. *Phys. Rev. Lett.* **118**, 158004.

Cone, C. D. 1962 Thermal soaring of birds. *Sci. Am.* **50**, 180–209.

Craven, B. A., Paterson, E. G. & Settles, G. S. 2009 The fluid dynamics of canine olfaction: unique nasal airflow patterns as an explanation of macrosmia. *J. R. Soc. Int.* pp. 933–943.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S. & Wu, Y. 2017 OpenAI Baselines.

Fenchel, T. 2002 Microbial behavior in a heterogeneous world. *Science* **296**, 1068–1071.

Gustavsson, K., Biferale, L., Celani, A. & Colabrese, S. 2017 Finding efficient swimming strategies in a three dimensional chaotic flow by reinforcement rearning. *Eur. Phys. J. E* **40**, 110.

Koehl, M. A. R. 2010 Hydrodynamics of sniffing by crustaceans. *Chemical Communication in Crustaceans* pp. 85–102.

Mahesh, K. 2013 The interaction of jets with crossflow. *Annu. Rev. Fluid Mech.* **45**, 379–407.

Marques, L., Nunes, U. & de Almeida, A. T. 2006 Particle swarm-based olfactory guided search. *Auton. Robot.* **20**, 277–287.

Peng, L., Silic, M. & Mohseni, K. 2015 A DDDAS plume monitoring system with reduced Kalman filter. *Procedia Comput. Sci.* **51**, 2533–2542.

van der Poel, E. P., Ostilla-Mónico, R., Donners, J. & Verzicco, R. 2015 A pencil distributed finite difference code for strongly turbulent wall-bounded flows. *Comput. Fluids* **116**, 10–16.

Rai, M. M. & Moin, P. 1991 Direct simulations of turbulent flow using finite-difference schemes. *J. Comput. Phys.* **96**, 15–53.

Rygg, A. D. 2014 *The hydrodynamics and transport phenomena of olfaction in the hammerhead shark (Sphyrna tudes): implications for bio-inspired chemical sensing*. Ph.D. Thesis, Pennsylvania State University.

Song, Z. & Mohseni, K. 2015 Anisotropic active Lagrangian particle swarm control in a meandering jet. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference*, pp. 240–245.

Spandan, V., Lohse, D., de Tullio, M. D. & Verzicco, R. 2018 A fast moving least squares approximation with adaptive Lagrangian mesh refinement for large scale immersed boundary simulations. *J. Comput. Phys.* **375**, 228–239.

Spandan, V., Meschini, V., Ostilla-Mónico, R., Lohse, D., Querzoli, G., de Tullio, M. D. & Verzicco, R. 2017 A parallel interaction potential approach coupled with the immersed boundary method for fully resolved simulations of deformable interfaces and membranes. *J. Comput. Phys.* **348**, 567–590.

Sutton, R. S. 1998 *Reinforcement learning: An introduction*. Kluwer Academic Publishers.

Vergassola, M., Villermaux, E. & Shraiman, B. I. 2007 'Infotaxis' as a strategy for searching without gradients. *Nature* **445**, 406409.

Verma, S., Novati, G. & Koumoutsakos, P. 2018 Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proc. Natl. Acad. Sciences U.S.A.* **115**, 5849–5854.

Verzicco, R. & Orlandi, P. 1996 A finite-difference scheme for three-dimensional incompressible flows in cylindrical coordinates. *J. Comput. Phys.* **123**, 402–414.

Weissburg, M. J. & Zimmer-Faust, R. K. 1994 Odor plumes and how blue crabs use them in finding prey. *J. Exp. Biol.* **197**, 349–375.