Applying Reinforcement Learning to Navigation In Partially Observable Flows

Selim Mecanna
IRPHE, Aix Marseille Univ, CNRS
Centrale Méditerranée, Marseille, France
selim.mecanna@centrale-marseille.fr

Aurore Loisy
IRPHE, Aix Marseille Univ, CNRS
Centrale Méditerranée, Marseille, France
aurore.loisy@cnrs.fr

Christophe Eloy
IRPHE, Aix Marseille Univ, CNRS
Centrale Méditerranée, Marseille, France
christophe.eloy@centrale-marseille.fr

Abstract

We consider the problem of navigating in a fluid flow while being carried by it, using only information accessible from on-board sensors. This POMDP (partially observable Markov decision process) is particularly challenging because to behave optimally, the agent has to exploit coherent structures that exist in the flow without observing them directly, and while being subjected to chaotic dynamics. It is yet commonly faced by autonomous robots deployed in the oceans and drifting with the flow (for, e.g., environmental monitoring). While some attempts have been made to use reinforcement learning for navigation in partially observable flows, progress has been limited by the lack of well-defined benchmarks and baselines for this application. In this paper, we first introduce a well-posed navigation POMDP for which a near-optimal policy is known analytically, thereby allowing for a critical assessment of reinforcement learning methods applied to autonomous navigation in complex flows. We then evaluate the 'vanilla' learning algorithms commonly used in the fluid mechanics community (Advantage Actor Critic, Q-Learning) and report on their poor performance. Finally, we provide an implementation of PPO (Proximal Policy Optimization) able to match the theoretical near-optimal performance. This demonstrates the feasibility of learning autonomous navigation strategies in complex flows as encountered in the oceans.

1 Introduction

Developing artificial microswimmers with navigation capabilities has recently been an intense topic of research [S. Muiños-Landin et al., 2021, B. Dai et al., 2016, H. Huang et al., 2019]. When such robots with limited self-propulsion abilities are carried by a fluid flow, navigation becomes notoriously harder. This is the kind of challenge faced by robots deployed in the oceans for environmental monitoring purposes. Ideally, these drifting robots would be able to exploit background currents to travel more efficiently or more quickly while relying only on data from their on-board sensors. The very same problem is also faced by planktonic organisms, and it is believed that while they have only access to local flow measurement, these organisms manage to exploit this partial information to migrate efficiently in the oceans [R. Monthiller et al., 2022, J. Wheeler et al., 2019, T. Kiørboe et al., 1999].

If the agent had global information about the flow, optimal control theory could be used to find optimal trajectories (a problem known as Zermelo's navigation problem [E. Zermelo, 1931]). But

17th European Workshop on Reinforcement Learning (ewrl 2024).

when the agent can only sense the flow *locally* (that is, has only access to a *partial observation*), optimal control theory can no longer be used. This problem becomes a model-free partially observable Markov decision process (POMDP). Such model-free problems are usually well suited for reinforcement learning techniques, which are based on environment sampling.

Yet, navigation in partially observable flows fundamentally differs from standard POMDPs. It has an important stochastic element to it, even though the dynamics are governed by deterministic equations, the often chaotic nature of the background flow naturally includes uncertainty in all the signals observed by the agent. Unlike traditional stochastic POMDP environments where the randomness is manually incorporated in the dynamics [F. Stulp et al., 2012, Z. Sunberg and M. Kochenderfer, 2017] or the agent/environment interaction [P. Vamplew et al., 2021]. In this problem, the challenge is to filter through the noise to understand and benefit from some underlying structure of the flow.

In recent years, navigation in partially observable flows has attracted considerable attention in the fluid mechanics community [M. Gazzola et al., 2014, J. Qiu et al., 2021, J. Alageshan et al., 2020, S. Colabrese et al., 2017, K. Gustavsson et al., 2017, J. Qiu et al., 2022, Calascibetta et al., 2023], who started adopting reinforcement learning techniques to develop "smart" artificial microswimmers. Yet the lack of a proper benchmark with well-defined baselines has hindered progress on this front, and the majority of the latest studies still rely on tabular Q-Learning.

The contributions of this paper are the following:

- We define a well-posed benchmark navigation problem for which the near-optimal policy is known.
- We demonstrate that 'vanilla' learning algorithms, despite being still routinely used for this purpose, are actually not sufficient for navigation in partially observable flow.
- We show that our implementation of PPO (Proximal Policy Optimization [J. Schulman et al., 2017]) allows learning a policy matching the near-optimal performance, thereby demonstrating that reinforcement learning is indeed a promising path toward autonomous navigation in flows.
- We release a Gym-compatible environment implementing the navigation problem, together with baselines and reinforcement learning algorithms adapted to it.

2 Autonomous navigation in partially observable flows

2.1 The task

We consider a swimming agent tasked with the objective to go as far as possible in the upward $\hat{\boldsymbol{z}}$ direction. The agent has a constant swimming speed v and can only control its swimming direction $\hat{\boldsymbol{p}}(t)$. The agent is modeled as an inertialess point-like particle advected by the surrounding flow $\boldsymbol{u}(\boldsymbol{x}, t)$ without affecting it. It can instantly change its orientation to the desired one. The agent motion is governed by the following equation:

$$\frac{d\boldsymbol{X}(t)}{dt} = \boldsymbol{u}(\boldsymbol{X}(t), t) + v\hat{\boldsymbol{p}}(t), \quad \boldsymbol{X}(t_0) = \boldsymbol{X}_0 \tag{1}$$

where $\boldsymbol{X}(t)$ is the position of the agent at time t, and $\boldsymbol{u}(\boldsymbol{x}, t)$ is the flow velocity field. The agent initial position $\boldsymbol{X}_0$ is randomly initialized in the flow, and the starting time t_0 is also chosen randomly (when relevant).

At each time step the agent is required to pick an orientation $\hat{\boldsymbol{p}}(t)$ that depends only on the local velocity gradient tensor defined by $(\boldsymbol{\nabla u})_{ij} = \partial u_j / \partial x_i$ or the anti symmetric part of that tensor given by $\boldsymbol{\nabla u}^a = \frac{1}{2}\left(\boldsymbol{\nabla u} - \boldsymbol{\nabla u}^T\right)$, depending on the flow considered. These constitute the observations of the agent which will be denoted as $\boldsymbol{G}$ in what follows. This choice of observable is explained by the fact that only flow *gradients*, rather than the flow itself, can be measured by an agent drifting with the flow. The orientation picked based on $\boldsymbol{G}$ will then determine the next position of the agent by following equation 1. The goal of the agent is to maximize the total distance J travelled projected onto the target direction $\hat{\boldsymbol{z}}$ over an episode of duration $(t_f - t_0)$:

$$J = (\boldsymbol{X}(t_f) - \boldsymbol{X}_0) \cdot \hat{\boldsymbol{z}} \tag{2}$$

This task is representative of long-distance navigation to a target point (here moved to infinity), without the additional difficulties associated to sparse rewards in goal-oriented tasks. It is simple enough to have a known analytical near-optimal solution (Section 2.3) while retaining the complexity inherent to navigating while being carried by a flow. For these reasons it provides a well-posed benchmark problem for evaluating the capabilities of reinforcement learning applied to autonomous navigation in flows.

2.2 The flows

In this work we consider three flow environments of increasing difficulty and realism: TGV, ABC, and TURB. They are prototypical flows commonly used in fluid mechanics and they contain an increasing number of the key features of real flows: coherent structures (TGV, ABC, TURB), chaotic dynamics (ABC, TURB), and unsteadiness (TURB).

TGV stands for Taylor-Green vortex. It is 2D steady flow consisting of a lattice counter-rotating vortices (Figure 1). This simple flow contains well-defined coherent structures (vortices).

ABC stands for Arnold-Beltrami-Childress flow. It is a 3D steady flow characterized by coherent tube-like structures separated by a chaotic region (Figure 2).

TURB is a 'realistic' 2D unsteady turbulent flow obtained by numerical simulation of the Navier-Stokes equations. This multiscale chaotic flow features moving vortical structures that have a finite lifetime: they unpredictably appear, evolve and vanish (Figure 3).

All these flows are 2π-periodic in all directions: when the agent is at position $\boldsymbol{X}(t)$, the flow at this location is given by $\boldsymbol{u}(\boldsymbol{X}(t) \mod 2\pi, t)$. In TGV and TURB the agent has access to the local velocity gradient $\boldsymbol{G} = \boldsymbol{\nabla u}$ while in ABC, the agent can only access its anti-symmetric part $\boldsymbol{G} = \boldsymbol{\nabla u}^a$. This latter choice is motivated by consistency with prior work on ABC flow where this observable was chosen. More details about these flows are provided in Appendix A.2.

2.3 Analytical baselines

Two heuristic policies are considered as baselines.

The *naive* policy consists in always aiming upward: $\hat{\boldsymbol{p}} = \hat{\boldsymbol{z}}$. This naive policy is a weak baseline, representative of baselines used in prior work applying 'vanilla' reinforcement learning algorithms for navigation in flows.

The *surfing* policy is the near-optimal policy that can be derived analytically from optimal control theory (cf. derivation in Appendix A.1). It provides a very strong baseline that reinforcement learning should match in order to be considered, in our view, as a suitable method for autonomous navigation in flows. Its name comes from its physical interpretation [R. Monthiller et al., 2022], as the agent 'surfs' on beneficial upward currents. The surfing policy reads

$$\hat{\boldsymbol{p}} = \boldsymbol{\lambda}/\|\boldsymbol{\lambda}\|, \qquad \boldsymbol{\lambda} = [\exp(\tau \boldsymbol{G})] \cdot \hat{\boldsymbol{z}} \tag{3}$$

where $\exp$ is the matrix exponential. The parameter τ has a physical meaning: it quantifies the mean correlation time of $\boldsymbol{G}$ as observed by the agent along its trajectory. For all practical purposes τ can be seen as a free parameter of the surfing policy that can be manually optimized for each flow (cf. Appendix A.5).

2.4 Formulation as a POMDP

A partially observable Markov decision processes (POMDP) is defined as an eight tuple $(S, A, T, r, O, p_0, \gamma)$ where $s \in S$ is the state containing enough information so that the system has the Markovian property, $a \in A$ is the action performed by the agent at a given state, $T(s'|s, a)$ is the transition probability to the state s' given the current state and action $T : S \times A \times S \to [0, 1]$, the reward $r(s, a, s')$ is a function $r : S \times A \times S \to \mathbb{R}$ that is given to the agent upon reaching a new state, $o \in O : S \to O$ is the observation that constitutes the information that the agent can use to make an action, $p_0 : S \to [0, 1]$ is the probability to start in a given state s , and finally $\gamma \in [0, 1)$ is

the discount factor.

The policy $\pi(a|o)$ is the probability (discrete actions) or probability density function (continuous actions) of performing action a given observation o. It defines the strategy followed by the agent. The return, $R_n = \sum_{k=0}^{N-1} \gamma^k r_{n+k}$ is the discounted sum of future rewards. And finally, the value function over the observation $V^\pi(o_n) = \mathbb{E}^\pi [R_n|o_n]$ is the expected value of the return if a specific policy is followed. The goal is to maximize the return over an entire episodes which is achieved by finding the optimal policy $\pi^* = \arg\max_\pi \ V^\pi(o_0)$

In order to formulate our problem as a POMDP we need to discretize Equation (1) in time:

$$\boldsymbol{X}(t_{n+1}) = \boldsymbol{X}(t_n) + \int_{t_n}^{t_{n+1}} \left(\boldsymbol{u}(\boldsymbol{X}, t) + v\hat{\boldsymbol{p}}(t_n)\right) dt \tag{4}$$

Equation (4) represents the transition matrix $T(s'|s, a)$ and the dictionary of the other variables to the navigation problem is presented in Table 1

Table 1: POMDP applied to flow navigation.

POMDP VARIABLE	NAVIGATION VARIABLE
s_n	$\{\boldsymbol{X}(t_n), \boldsymbol{u}(\boldsymbol{X}, t) \, \forall t\}$
a_n	$\hat{\boldsymbol{p}}(t_n)$
o_n	$\boldsymbol{G}(\boldsymbol{X}(t_n), t_n)$
r_n	$[\boldsymbol{X}(t_{n+1}) - \boldsymbol{X}(t_n)] \cdot \hat{\boldsymbol{z}}$

2.5 Related work

Applying reinforcement learning for navigation in fluid flows recently started gaining popularity. In many of these studies the agent has global flow information, and to apply reinforcement learning on these problems people often rely on tabular Q-Learning [Schneider and Stark, 2019, Yoo and Kim, 2016], or in some cases A2C [M. Nasiri and B. Liebchen, 2022, L. Biferale et al., 2019]. With global information, optimal control provides a baseline to assess the performance of the learned algorithm.

In the case where the agent can only access local flow information, Q-Learning is also widely used [M. Gazzola et al., 2014, J. Qiu et al., 2021, J. Alageshan et al., 2020, S. Colabrese et al., 2017, K. Gustavsson et al., 2017, J. Qiu et al., 2022, Calascibetta et al., 2023]. In rare cases, new algorithms have been developed to tackle similar problems, such as V-RACER [P. Gunnarson et al., 2021]. It is difficult to assess the ability of learning algorithms at solving the task since no theoretical solution is known, and no comparison of performance across different reinforcement learning algorithms is provided. Besides, actions (and often states) are discretized which will necessarily hinder performance.

In this work, we formulate a navigation task for which we known a near-optimal policy (known as "Surf"), such that performance of learned strategies can be meaningfully assessed. The difficulty of the task can be adjusted by choosing different background flows. We compare the performance of the two 'vanilla' learning algorithms (Q-Learning, A2C) used in the fluid mechanics community to our own implementation of PPO, a popular algorithm which has demonstrated its capabilities across a wide range of domains.

3 Methods

We introduce the three baseline algorithms considered in this work: Q-Learning, A2C, PPO. For each environment and algorithm we train five times (five random seeds) over 10^6 episodes. In the TURB environment, we use 80% of the simulation time for training. Hyper-parameters for each algorithm were obtained manually to achieve best performance check Appendix A.3 for the hyper-parameters of every algorithm in each domain.

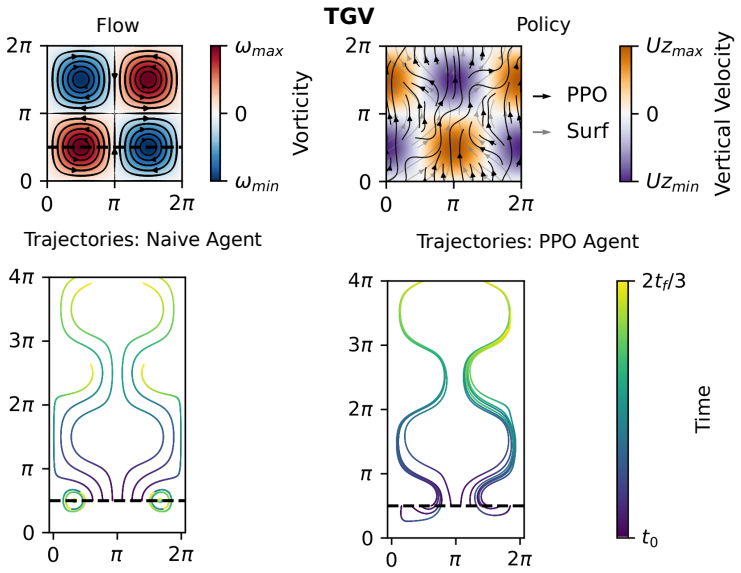

Figure 1: The TGV flow is represented in the upper left corner by showing the vorticity ($\boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{u}$) along with the streamlines. The dashed line represents the initialization of particles in the trajectories plots shown in the second row. These plots show the trajectories of particles following the Naive strategy (left), and the learned PPO strategy (right). It can be seen that PPO tends to reach the top more reliably than naive achieving higher performances, and that its trajectories all converge. The reason for that is highlighted by the upper right plot showing the policies of both PPO and Surf where it is clear that both tend to diverge from negative upward velocities (violet) and converge to positive upward velocities (orange).

Q-Learning: For Q-Learning it is required to discretize both the observation O and action A spaces. Every component of the observation vector o is divided into three possible categories $[\overline{o}-, \overline{o}, \overline{o}+]$ such that one third of the data sampled from each environment belonged to each category. The actions are discretized in the following way $A : [\hat{\boldsymbol{z}} \perp, \hat{\boldsymbol{z}}, -\hat{\boldsymbol{z}} \perp, -\hat{\boldsymbol{z}}]$ with four perpendicular directions oriented with $\hat{\boldsymbol{z}}$ in 2D, and the two additional perpendicular directions in 3D. We used an optimistic initialization of the Q-matrix to enhance exploration.

A2C: A2C is an actor-critic method that utilizes the advantage function to update the policy, it is an on-policy and online algorithm that updates the value and policy while interacting with the environment. For A2C we use continuous observation and action spaces, where the output of the actor is a von Mises-Fisher distribution that represents the orientation of the agent in 2D and 3D. The main reference used in the implementation of A2C is [R. Sutton and A. Barto, 1998].

PPO: PPO also belongs to the actor-critic on-policy family of algorithms, it is more sample-efficient than A2C. In our implementation we use a vectorized architecture, generalized advantage estimation, advantage normalization, and separate MLP networks for policy and value estimation. The observation and actions spaces are also continuous here, with the same von Mises-Fisher distribution as the output of the actor. Our implementation of PPO is heavily influenced by [H. Shengyi et al., 2022].

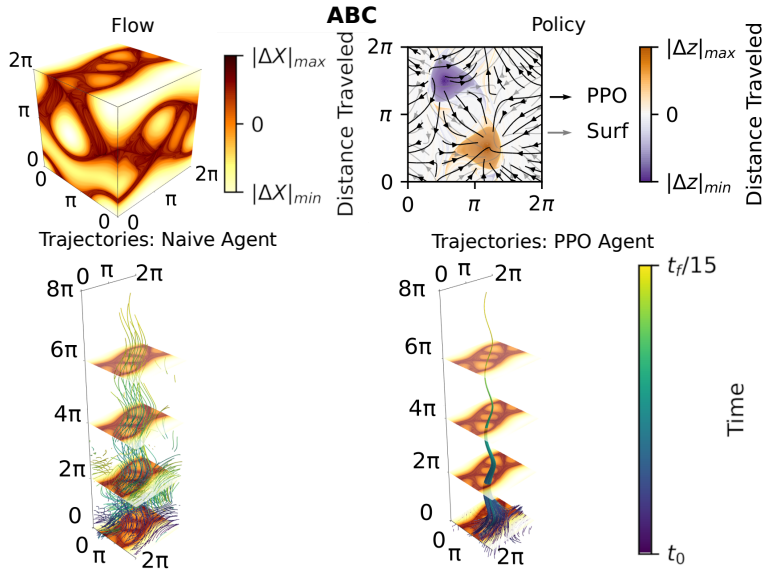

Figure 2: The ABC flow is represented in the upper left corner by showing the distance travelled by tracers advected by the flow (such quantity, called Lagrangian descriptor [Agaoglou et al., 2020], highlights flow regions with qualitatively different dynamics). This flow contains many coherent tube-like structure (light yellow) where tracers tend to cover large distances, these areas are also associated with preferential directions. They are separated by a chaotic region (dark red). In the bottom panels where trajectories are shown, agents are initialized at the $z = 0$ plane, following the Naive strategy (left) and PPO (right). The same behavior is observed when comparing to TGV, PPO reaches the top more reliably, and trajectories converge. This is because PPO benefits from the structure with preferential direction aligned with the target direction $\hat{z}$. This is also highlighted in the policy plot where both PPO and Surf diverge from downward currents (violet) and converge to upward currents (orange).

Table 2: Performance of the policies (with 95% confidence intervals) in the three flow environments.

	PPO	A2C	QL	Surf	Naive
TGV	**16.23** $\pm$ 0.1	11.32 $\pm$ 0.1	12.16 $\pm$ 0.1	14.84 $\pm$ 0.1	9.96 $\pm$ 0.1
ABC	**70.6** $\pm$ 1.0	69.6 $\pm$ 1.0	56.6 $\pm$ 1.0	62.5 $\pm$ 1.0	30.3 $\pm$ 1.0
TURB	**15.05** $\pm$ 0.1	12.02 $\pm$ 0.1	11.65 $\pm$ 0.1	**15.09** $\pm$ 0.1	10.04 $\pm$ 0.1

4 Results

4.1 Robustness over training trials

Figure 4 shows the beginning of the learning curves (over 10^5 episodes) for the ten trials over all the environments and algorithms. We use this curve to judge the robustness of each algorithm.

In all of the flows, PPO reliably attains the same solution with very little variance when compared to the other algorithms. In general, PPO converges quickly to the final solution, and the final policy is deterministic. The reason for this is that PPO can handle high learning rates for both the actor and the critic networks, and even multiple epoch training because the updated policy is guaranteed not to differ so much from the original one by clipping the loss function.

A2C is shown to be robust in TURB, but it was less repeatable in the other environments where one trial in TGV did not learn anything. A2C tend to converge much slower than the other algorithms, this is due to the fact that both the actor and critic networks need to have small learning rates to

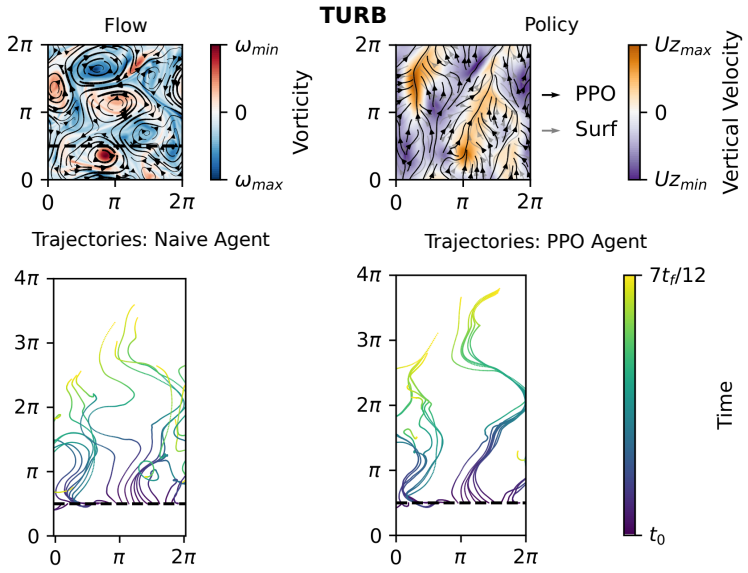

Figure 3: A snapshot of the time-dependent turbulence simulation is represented in the upper left along with the streamlines, the snapshot corresponds to the time $t = t_0$ which is used with the dashed line to initialize the trajectories in the lower plots. Just as in TGV and ABC, following the PPO strategy achieves higher performances more reliably, and the trajectories tend to clump together. This is again shown in the policy plot by noting that PPO and Surf tend to diverge from negative upward velocities (violet) and converge to positive upward velocities (orange). In this case we can see a clear resemblance between the policies of PPO and Surf.

ensure stability for this algorithm. The absence of control over the policy change in each update limits the algorithm to small learning rates and prevents it from doing multiple epoch training, the learning rates found in A.3.2 are the largest learning rates that produce convergence .

Q-Learning has the largest variance across environments, and it often unlearns good strategies which is shown by the sudden decreases in performance. Additionally, in both ABC and TURB one trial outperformed the rest rendering it non repeatable. In general Q-Learning learns fast, but it is very unstable, its performance and robustness are also hindered by the fact that it is the only algorithm here that uses discrete sets of observations and actions.

4.2 Performance

Table 2 is used to judge the performances of each algorithm across the different environments. We pick the best out of the ten trials, and the neural networks and Q-matrices are saved at the point where they achieve highest average returns on the learning curves, this is done to ensure that we include the portions where performance decreases.

To analyze the performances, all the policies are transformed to greedy policies by taking the mean of the von Mises-Fischer distributions in the case of continuous actions, and the action corresponding to the highest action value function for the discrete case.

These policies are then tested in their corresponding environments, and the average returns are reported in Table 2. In the TURB environment, we use the 20% part of the simulation that was not used in learning to test the performance.

PPO is the best performing algorithm in both ABC and TGV, it additionally outperforms the analytical strategies in these flows. The returns in ABC are generally much higher than the other

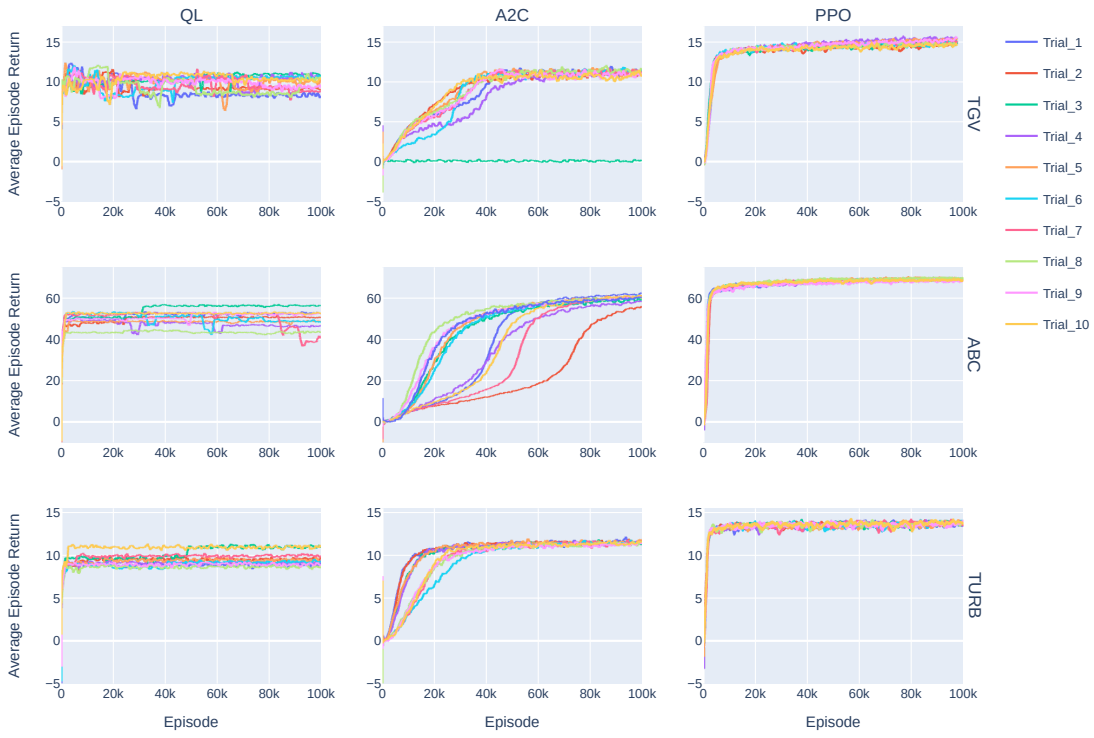

Figure 4: Learning curves of each algorithm (Columns) and environment (Rows).

environments, this is due to the structures highlighted in Figure 2 that have preferred directions aligned with $\hat{z}$. In TURB, PPO matches the performance of Surf, and they both outperform all the other learning algorithms.

5 Discussion and Conclusion

We have introduced a POMDP that describes a navigation task relevant to autonomous robotic microswimmers and planktonic organisms. This problem is challenging because it combines chaotic state dynamics with the difficulties inherent to partial observability. This benchmark problem also comes with an near-optimal solution useful to assess performance. We have shown that on this benchmark, A2C and Q-Learning, implemented the way they are in almost all prior studies on navigation in flows, actually performs rather poorly. Nevertheless, we demonstrate through our implementation of PPO that reinforcement learning methods can achieve the near-optimal theoretical performance on this task. This implies the ability of learning algorithms to discover and fully exploit hidden flow structures without having direct access to them. We expect this version of PPO to be a good starting point for solving possibly more challenging (e.g., goal-oriented) navigation tasks in partially observable flows.

In the most difficult environment (unsteady turbulence TURB), we highlighted the fact that PPO achieves near-optimal performance with a learned policy that is similar, but not identical to the analytically derived one. This hints to the existence of a family of policies that are different but equally performing. It remains to be elucidated whether these policies are truly equivalent.

The proposed navigation POMDP can be easily extended to incorporate more complexity and realism: increasing the difficulty of the background flow by using a 3D simulation of turbulence, including a response time in the agent dynamics such that the chosen direction is not instantly followed, or increasing the amount of information accessible to the agent by giving it memory or multiple points of measurement. In all of these cases the analytical strategy (Surf) can be used as is and provides a strong baseline to beat. As difficulty increases, standard learning algorithm may become inefficient. We hope this POMDP will encourage the development of new learning algorithms able to address the particular challenges associated to navigation in partially observable flows.

Acknowledgments and Disclosure of Funding

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 834238).

References

M. Agaoglou, B. Aguilar Sanjuan, V. J. García-Garrido, F. Gonzalez Montoya, M. Katsanikas, V. Krajňák, S. Naik, and S. R. Wiggins. *Lagrangian Descriptors: Discovery and Quantification of Phase Space Structure and Transport*. Zenodo, 2020. doi: 10.5281/zenodo.3958985.

B. Dai, J. Wang, Z. Xiong, X. Zhan, W. Dai, C. Li, S. Feng, and J. Tang. Programmable artificial phototactic microswimmer. *Nature Nanotechnology*, 11:1087–1092, 2016.

G. Boffetta and R. E. Ecke. Two-Dimensional Turbulence. *Annual Review of Fluid Mechanics*, 44(1): 427–451, 2012. doi: 10.1146/annurev-fluid-120710-101240.

C. Calascibetta, L. Biferale, F. Borra, A. Celani, and M. Cencini. Taming Lagrangian chaos with multi-objective reinforcement learning. *EPJ E*, 46(3):9, 2023. ISSN 1292-895X. doi: 10.1140/epje/s10189-023-00271-0.

E. Zermelo. Über das Navigationsproblem bei ruhender oder veränderlicher Windverteilung. *Journal of Applied Mathematics and Mechanics*, 11:114–124, 1931.

F. Stulp, J. Buchli, A. Ellmer, M. Mistry, E. Theodorou, and S. Schaal. Model-free reinforcement learning of impedance control in stochastic environments. *IEEE*, 4, 2012.

H. Huang, F. Uslu, P. Katsamba, E. Lauga, M. Sakar, and B. Nelson. Adaptive locomotion of artificial microswimmers. *Science Advances*, 5, 2019.

H. Shengyi, D. Julien, R. Atonin, K. Anssi, and W. Weixun / ICLR Blog Track. The 37 implementation details of proximal policy optimization, 2022. URL `https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/`.

J. Alageshan, A. Verma, J. Bec, and R. Pandit. Machine learning strategies for path-planning microswimmers in turbulent flows. *Phys. Rev. E*, 101, 2020.

J. Qiu, N. Mousavi, K. Gustavsson, C. Xu, B. Mehlig, and L. Zhao. Navigation of micro-swimmers in steady flow: the importance of symmetries. *J. Fluid Mech.*, 932, 2021.

J. Qiu, N. Mousavi, L. Zhao, and K. Gustavsson. Active gyrotactic stability of microswimmers using hydromechanical signals. *Phys. Rev. Fluids*, 7, 2022.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *Arxiv*, 1707, 2017.

J. Wheeler, E. Secchi, R. Rusconi, and R. Stocker. Not just going with the flow: The effects of fluid flow on bacteria and plankton. *Annual Review of Cell and Developmental Biology*, 35:213–237, 2019.

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

L. Biferale, F. Bonaccorso, M. Buzzicotti, P. Clark Di Leoni, and K. Gustavsson. Zermelo's problem: Optimal point-to-point navigation in 2d turbulent flows using reinforcement learning. *Chaos*, 29, 2019.

M. Gazzola, B. Hejazialhosseini, and P. Koumoutsakos. Reinforcement learning and wavelet adapted vortex methods for simulations of self-propelled swimmers. *SIAM J. Sci. Comput*, 36, 2014.

M. Nasiri and B. Liebchen. Reinforcement learning of optimal active particle navigation. *New J. Phys.*, 24:9, 2022.

P. Gunnarson, I. Mandralis, G. Novati, P. Koumoutsakos, and J. Dabiri. Learning efficient navigation in vortical flow fields. *Nat. Commun*, 12, 2021.

P. Vamplew, C. Foale, and R. Dazeley. The impact of environmental stochasticity on value-based multiobjective reinforcement learning. *Neural. Comput. Appl.*, 34:1783–1799, 2021.

R. Monthiller, A. Loisy, M. Koehl, B. Favier, and C. Eloy. Surfing on turbulence: A strategy for planktonic navigation. *Phys. Rev. Lett.*, 129, 2022.

R. Sutton and A. Barto. *Reinforcement Learning An Intorduction*. the MIT Press, 1998.

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

S. Muiños-Landin, A. Fischer, V. Holubec, and F. Cichos. Reinforcement learning with artificial microswimmers. *Science Robotics*, 6:8, 2021.

E. Schneider and H. Stark. Optimal steering of a smart active particle. 127(6):64003, 2019. ISSN 0295-5075. doi: 10.1209/0295-5075/127/64003.

T. Kiørboe, E. Saiz, and A. Visser. Hydrodynamic signal perception in the copepod Acartia tonsa. *Marine Ecology Progress Series*, 179:97–111, 1999.

B. Yoo and J. Kim. Path optimization for marine vehicles in ocean currents using reinforcement learning. 21(2):334–343, 2016. ISSN 1437-8213. doi: 10.1007/s00773-015-0355-9.

Z. Sunberg and M. Kochenderfer. Online algorithms for pomdps with continuous state, action, and observation spaces. *ICAPS*, 10, 2017.

A Supplementary Material

A.1 Derivation of the surfing policy (approximate optimal control)

Using Euler-Lagrange formalism, the optimal control problem can be reformulated as an ordinary differential equation for the adjoint $\boldsymbol{\lambda}$:

$$\frac{d\boldsymbol{\lambda}(t)}{dt} = -\boldsymbol{\nabla u}(\boldsymbol{X}(t), t) \cdot \boldsymbol{\lambda}(t), \quad \boldsymbol{\lambda}(t_f) = \hat{\boldsymbol{z}} \tag{5}$$

which solution is

$$\boldsymbol{\lambda}(t) = \left[\exp\left(\int_0^{t_f - t} \boldsymbol{\nabla u}(\boldsymbol{X}(t+\tau), t+\tau) d\tau\right)\right] \cdot \hat{\boldsymbol{z}} \tag{6}$$

where $\boldsymbol{X}(t)$ is the solution of Equation (1). In fluid flows, $\boldsymbol{\nabla u}$ is generally time-correlated over a finite time τ. This allows us to approximate the integral by $\tau \boldsymbol{\nabla u}(\boldsymbol{X}, t)$. The surfing policy, given by Equation (3), immediately follows after replacing $\boldsymbol{\nabla u}$ by $\boldsymbol{G}$. Note that neglecting the existence of correlations in the flow amounts to setting $\tau = 0$, which gives the naive policy.

A.2 Environments Details

A.2.1 TGV

The following are the equations representing the Taylor-Green vortices in 2D:

$$\begin{aligned}\dot{x} =& 0.5\cos(x)\sin(y),\\ \dot{y} =& 0.5\sin(x)\cos(y),\end{aligned}$$

Due to symmetries only two components of the velocity gradient $\boldsymbol{\nabla u}$ are independent, so the observation space is two-dimensional. In TGV the swimming speed of the agent is taken to be half the maximum velocity of the flow $v = \frac{V_{max}}{2}$.

A.2.2 ABC

The following are the equations representing the Arnold-Beltrami-Childress flow in 3D:

$$\begin{aligned}\dot{x} =& A\sin(z) + C\cos(y),\\ \dot{y} =& B\sin(x) + A\cos(z),\\ \dot{z} =& C\sin(y) + B\cos(x),\end{aligned}$$

Where we took $A = \sqrt{3}$, $B = \sqrt{2}$ and $C = 1$. ABC flow has the property that the velocity is equal to the vorticity $\boldsymbol{u} = \boldsymbol{\omega}$ which has three independent components making the observation space three dimensional. In ABC, the swimming speed of the agent is taken to be half the maximum velocity of the flow $v = \frac{V_{max}}{2}$.

A.2.3 TURB

Turbulent flow fields have been generated numerically by DNS (direct numerical simulation) using standard techniques for statistically stationary, homogeneous isotropic turbulence Boffetta and Ecke [2012]: the Navier-Stokes equations (which describe the spatiotemporal evolution of flows) are solved in a 2D periodic domain using a standard pseudo-spectral method on 256x256 points, and turbulence is sustained by applying a stochastic forcing at large scales. In 2D turbulence the three components of the velocity gradients are independent making the observation space three dimensional. In TURB the swimming velocity of the agent is taken to be half the RMS value of the flow $v = \frac{V_{rms}}{2}$.

A.3 Hyper-parameters

A.3.1 PPO

Table 3: Hyper-parameters used in PPO for the different environments.

Parameter	TGV	ABC	TURB
Learning-rate-actor	10^{-4}	10^{-4}	10^{-4}
Learning rate critic	10^{-3}	10^{-3}	10^{-3}
Anneal learning rates	True	True	True
Discount factor	0.99	0.99	0.99
Number of environments	100	10	10
Steps per update	10	10	10
GAE lambda	1.0	1.0	1.0
Number of mini-batches	5	5	5
Epochs	4	4	4
Clip coefficient	0.1	0.1	0.1
Entropy coefficient	0.0	0.0	0.0
Target kl divergence	0.02	0.02	0.02

A.3.2 A2C

Table 4: Hyper-parameters used in A2C for the different environments.

Parameter	TGV	ABC	TURB
Learning rate actor	10^{-6}	10^{-6}	10^{-6}
Learning rate critic	10^{-4}	10^{-4}	10^{-4}
Anneal learning rates	False	False	False
Discount factor	0.95	0.99	0.99

A.3.3 QL

Table 5: Hyper-parameters used in QL for the different environments.

Parameter	TGV	ABC	TURB
Learning rate	0.8	0.8	0.8
Anneal learning rates	True	True	True
discrete obs per input	3	3	3
epsilon	0.1	0.1	0.1
Discount factor	0.95	0.95	0.99

A.4 Neural Networks Parameters

A.4.1 Actor

Table 6: Parameters of the actor neural network in both A2C and PPO.

network parameters	value
number hidden layers	2
nodes per layer	40
layers	Dense
initialization	glorot-uniform
activation	elu
use feature normalization	True
optimizer	Adam
output	von Mises-Fisher

A.4.2 Critic

Table 7: Parameters of the critic neural network in both A2C and PPO.

network parameters	value
number hidden layers	2
nodes per layer	100
layers	Dense
initialization	glorot-uniform
activation	elu
use feature normalization	True
optimizer	Adam

A.5 Surf Tau Optimization

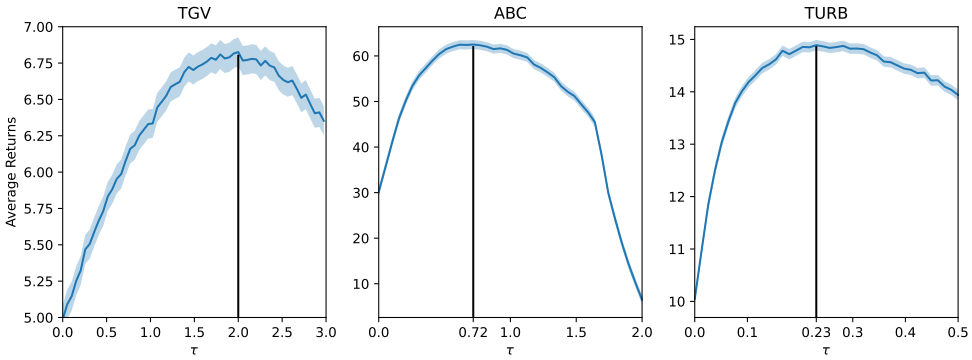

Figure 5: Optimization over τ in the different flows. In TGV $\tau^* = 2.0$, in ABC $\tau^* = 0.72$, and in TURB $\tau^* = 0.23$