Manipulation of Free-Floating Objects using Faraday Flows and Deep Reinforcement Learning

David Hardman*, **Thomas George Thuruthel**[1], **and Fumiya Iida**[1]

[1]All authors are with the Bio-Inspired Robotics Lab, Department of Engineering, Cambridge, CB2 1PZ, UK.
*dsh46@cam.ac.uk

ABSTRACT

The ability to remotely control a free-floating object through surface flows on a fluid medium can facilitate numerous applications[1–4]. Current studies on this problem have been limited to uni-directional motion control[4] due to the challenging nature of the control problem. Analytical modelling of the object dynamics is difficult due to the high-dimensionality and turbulent nature of the surface flows while the control problem is hard due to the nonlinear slow dynamics of the fluid medium, underactuation, and chaos. This study presents a general methodology for manipulation of free-floating objects using large-scale physical experimentation and recent advances in deep reinforcement learning. We show the effectiveness of our methodology by demonstrating the open-loop control of a free-floating objects in water using a robotic arm. Our learned control policy is relatively quick to obtain, highly data efficient, robust to the object morphology, and easily scalable to a higher-dimensional parameter space and/or experimental scenarios.

Introduction

Remote control of floating objects on a fluid surface has numerous applications ranging from manipulation of micro-particles[1–3] to macro-particles[4]. They are usually performed by transfer of energy through the surrounding medium[1,4,5] or through power sources that can be localized remotely such as magnetic fields and optical forces[6–8]. Controlled manipulation through the surrounding medium is highly desirable as it is potentially more scalable to larger areas and varying objects using simple control sources. The control objective is to generate surface flows on the fluid surface by parametrically excited high-dimensional local wave sources. These create two-dimensional horizontal velocity fields at the surface, also known as Faraday flows, that have attributes of two-dimensional turbulence; characterised by surface vortices and local Lagrangian coherent attractors and repellers[4,9–11]. The study of these surface flows and their use for object transport has been limited due to difficulties in modelling this nonlinear chaotic system and the high-dimensionality of the environment. Hence, current studies have been limited to empirical studies on uni-directional control of floating objects by one-dimensional wave sources[4]. Surface flows generated by multi-periodic wave sources are even limited to harmonic periods[12,13]. This works extends the state-of-the-art to the two-dimensional control of a floating object in water driven by multi-periodic wave sources using recent advances in deep reinforcement learning (DRL).

Developing optimal controllers directly from real-world experiments is appealing for fluid mechanics problems due to the difficulties in developing accurate mathematical models for them[14]. However, such controllers should be able to handle the high nonlinearity, high-dimensionality, non-convexity and the underactuated nature of the problem. Deep learning-based approaches[15] have been exceptional in handling the nonlinearity and dimensionality of the problem without falling into local minima[16–18]. Transferring these deep architectures to reinforcement learning have also shown promise in solving optimal control and design problems, particularly in shape optimization[19,20] and flow control[21,22]. Nevertheless, the application of DRL for fluid control problems are still limited to simulation studies, anticlimactic to their actual strengths[21].

This work presents the use of DRL for solving the problem of manipulating a free-floating object through surface flows. We structure the problem as an open-loop delivery task; given a randomised starting position of a object in a water tub, the controller drives the object to a target location at the edge of the tub. The floating object is driven by a complex multi-period oscillation of a robotic arm submerged at the centre of the container. The motion of the floating object follows its interaction with the surface flows generated in the fluid (Figure 1). By restricting our actions to an open-loop case, the physical setup is kept simple, requiring little sensory feedback, and lending itself to applications where delicate closed loop systems are undesirable and/or impractical. In order to select the right parametrization of the control space and the reward function, we first perform a global optimization routine selective targets. Then a DRL algorithm - deep deterministic policy gradient (DDPG) - is used to learn the global controller[23]. DDPG uses an deep actor-critic architecture to learn control policies for continuous action spaces. DDPG is an off-policy reinforcement learning method that concurrently learns a value function and a policy.

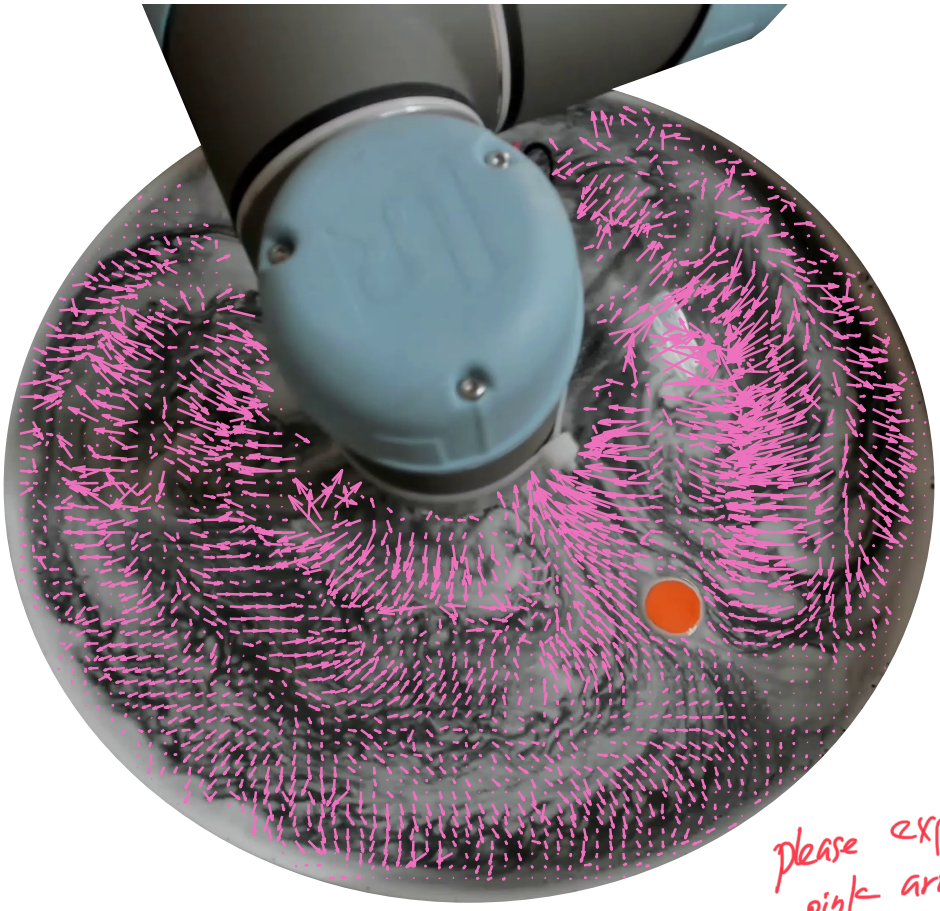

Figure 1. Manipulation of a free-floating object (in orange) by generating surface flows with a robotic arm. The free-floating object is guided across the water by the visualised complex surface flows.

Our experimental results show that even with the open-loop constraint and the turbulent nature of the surface flows, the control problem can be solved with reasonable accuracy. Using empirical studies we investigate the complexity of the problem and the strategy behind the optimal control solutions. We show that this global control policy can also be obtained through few days of training with reward shaping, even with the deep architectures we employ for our controller. Our results show how chaotic attractors and repellers in the surface flows can be used to generate repeatable trajectories. We also report the generation of multi-periodic surface flows and transiently stable flows based on the target location. As the motion of the floating object is determined by the surface flows, we also show that the control policies can be potentially transferable to object of various shapes and sizes.

Results

In order to study the feasibility and repeatability of the delivery task, we first use a global optimization method called Bayesian optimisation (BO). From a fixed starting position, three tasks are chosen, requiring the free-floating object to be driven to the container's edge after rotating 0°, 90°, & 180°. For each, a separate Bayesian minimisation is run to find the optimal control parameters (See Task Implementation) that drive the object closest to the desired target location. Figure 2 shows the progress of the three optimisations, where each iteration is assigned a cost function that decreases as the object approaches the target. After ∼650 iterations, the best action is repeated 5 times and plotted in Figure 2. Within the first 100 iterations low cost 0° actions are found and significantly improved upon, leading towards a highly repeatable method at the end of the optimisation. Though the 90° case takes longer to converge to a solution, a successful set of actions is still developed: only one of the 5 repetitions significantly deviates from the average path, but still reaches the edge close to the target location. We see more variance in the 180° paths: though all of the repetitions come close to the target, none hit it directly, with one attempt remaining uncompleted over the control period. The Bayesian optimization results provide an insight into the complexity of the delivery tasks. As expected, driving the object directly away from the oscillating plunger is an easier solution to find and more repeatable. As the

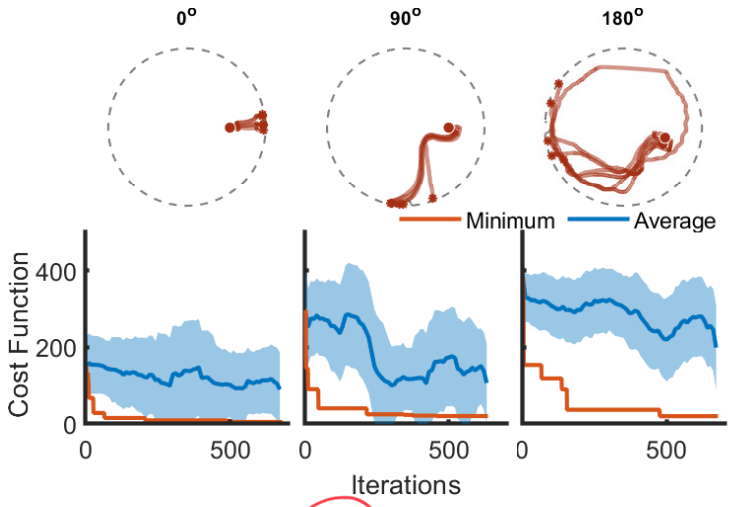

Figure 2. Bayesian optimization results for the 0°, 90°, & 180° open-loop delivery tasks. After ∼650 iterations, the best set of actions are repeated 5 times and the object's path plotted.

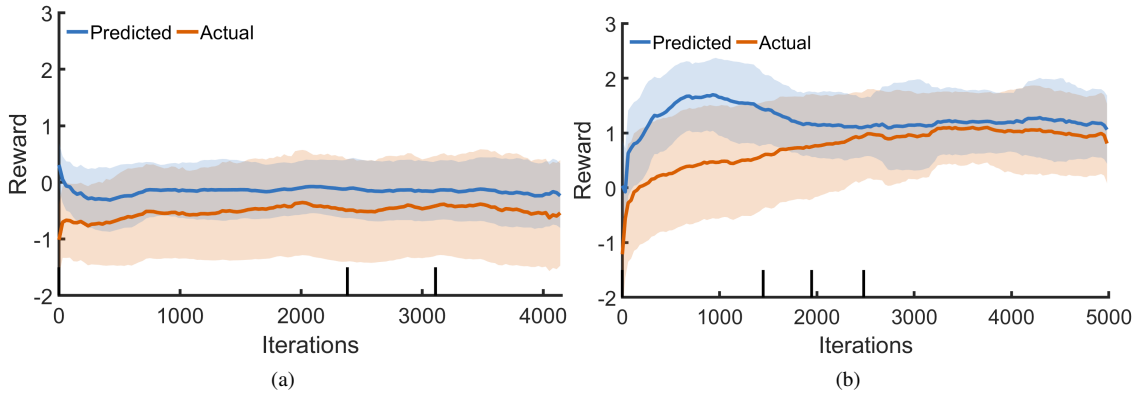

Figure 3. (a) DDPG training using a naive reward function that tries minimise the distance between target and object. (b) DDPG training with rewards shaped using prior knowledge.

traversed path increases, the observed trajectory is more unpredictable and hence more difficult to be obtained through any search algorithms. Nevertheless, with careful selection of the cost function, we can find control strategies with a high likelihood of success. The Bayesian optimization takes ∼500 iterations to guarantee a good solution for each task, which with an upper bound of two minutes for each control episode, amounts to around 16 hours of real-world experiments. Generating global controllers using these local solutions are hence infeasible for practical purposes. DRL, however, should be able to exploit any structure in the state space and control solutions to greatly reduce the data requirements.

For our study two DDPG agents were trained and analyzed. The first agent uses a naive reward function that is concerned only with minimising the distance between the object's centre and the target location. The second agent uses a reward function shaped by our observations on the BO tests (See "Methods"). The training progress of the two DDPG agents is shown in Figure 3. The naive reward function, although the most straightforward, is highly inefficient. From the BO results, it is apparent that it is easier to move radially than angularly and that near-optimal control actions can still lead to a large distance between the target and the object because of bifurcations and early edge contact. Therefore, the rewards are shaped such that outward motions are encouraged; with completed paths rewarded more highly than otherwise similar uncompleted paths. This leads to a sharp rise in reward near the start of training, seen in Figure 3(b), as the controller learns to find the easy and high reward actions. From here, the average reward rises more gradually, as completed solutions learn to approach the target locations on the container's edge. In addition, actions are rewarded for reaching the edge in short time, discouraging excessively long paths prone to bifurcations and non-repeatability. This reward shaping technique allows us to develop a global control policy in a relatively short time. Note that not only are we able to obtain a higher reward faster, but also the critic (predictor) converges faster for the latter reward function. For all furthers tests and analysis the second agent is selected from here on.

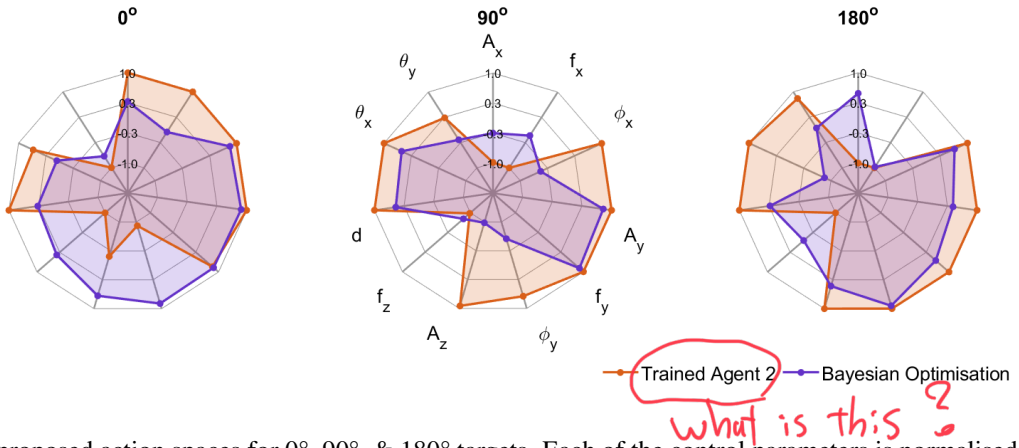

Figure 4. The proposed action spaces for 0°, 90°, & 180° targets. Each of the control parameters is normalised in the range (-1, 1).

Figure 4 compares the proposed action variables by the learned agent to the optimal action variables obtained through BO for the three control tasks mentioned previously in Figure 2. The control solutions obtained by both methods have certain dissimilarities showing the non-uniqueness of the solutions. It is also observable that the DDPG solutions tend to be nearer to the edges of the action ranges. With longer training times these solutions can be expected to converge to a more optimal one.

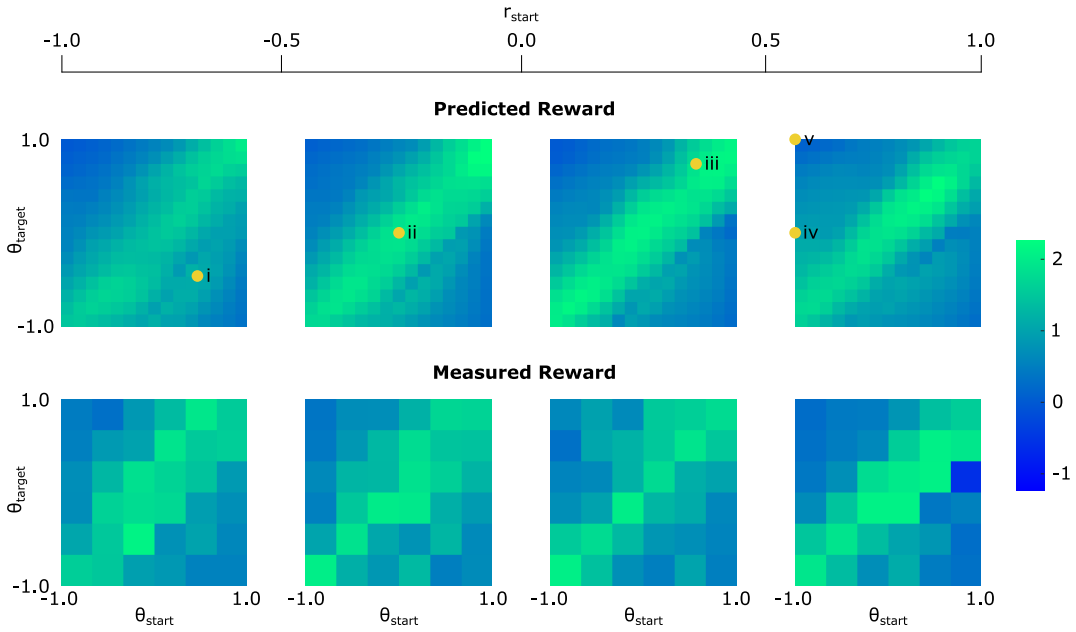

Figure 5. Predicted and measured rewards of the trained actor over the three dimensional state-space, target-space tuple; normalised over the range (-1, 1).

Due to the actor-critic architecture of DDPG, it is also possible to estimate the expected return of an action using the critic. The upper row of Figure 5 show the expected rewards from the agent's actor using the critic. Each plot's axes compare the normalised starting and target angles, whilst the starting radius is incremented between columns. Regardless of this radius, strong diagonal bands demonstrate the ease with which the controller learns to push the object radially with high repeatability. As the angles separate, the predicted reward decreases, reflecting the increased problem complexities already encountered in Figure 2. Only two cases are predicted negative rewards - the upper left cells of the first and third plots which require 180° rotations, though the predictions are still significantly higher than the -1.25 minimum. Interestingly, despite the radial symmetry of the setup, the lower right quadrants of the plots contain marginally higher rewards than those in the upper left. This is assumed due to the convergence location of the actor during training, though could be the manifestation of small asymmetries, such as fewer tracking errors in one half of the container. The predicted patterns are well reproduced in Figure 5's lower row,

which plots the measured rewards over 1000 tests of the trained agent with random starting states and targets. The matching columns suggest a well-trained critic, which is expected as critics can use all the sampled data for training updates, whereas the actor requires high reward data points for meaningful updates.

To systematically test the actor, 5 points (marked i-v) are chosen from Figure 5, covering a range of starting and target positions. For each corresponding state, the actions generated by the final agent are performed 10 times, and the tracked paths are shown in Figure 6. To analyze the progress of the agent over training, the development of the paths with the earlier agents are also shown, each repeated 5 times. In all 5 tasks, the average reward increases with training iterations, as more data is available to the agent for training. Note that in almost all cases the agent first learns to move out radially first and then adjusts its trajectory to improve its reward (except for Task v, where the the agent stumbled upon a good solution after only 1000 iterations).

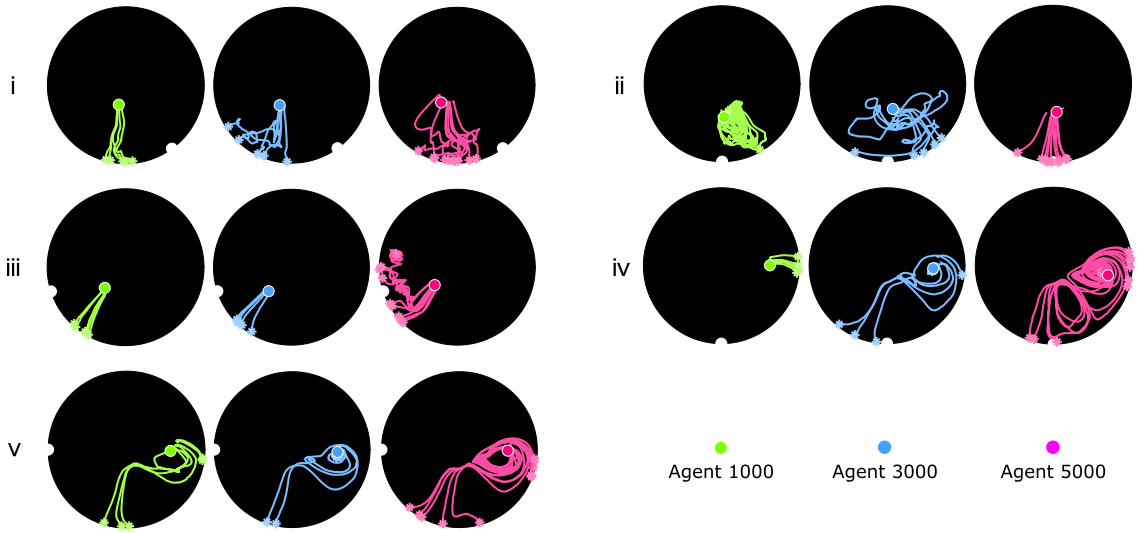

Figure 6. Development of the agent's solutions throughout the training process for the 5 selected tasks. Target locations are marked using notches on the container's perimeter.

For the two difficult targets (Tasks iv and v), the trajectories taken by the object are more complex and stochastic. The object performs a shoot-off strategy wherein the object is pushed towards the edge of a stationary vortex and rerouted to another flow field at the bifurcation of a chaotic attractor, making the strategy highly robust to changes in initial conditions (vital for open-loop strategies). Note that the stochastic nature of the final object state is heavily influenced by the strong capillary attraction towards the edge of the container. We can also observe that these two solutions differ from Tasks i-iii in that the surface flows which they generate - visualised in Figure 7(a) - are stationary, maintaining the same flow pattern indefinitely when the actions are run. Tasks i-iii instead show some periodic behaviours in the flow patterns, which shift over time with observed periods of 73s, 53s, & 3.5s respectively. Their corresponding average completion times are 15.2s, 4.2s, & 13.7s, indicating that only Task iii relies on the full effect of these periodic oscillations, which are marked in Figure 7(b). Comparing this with the paths of Figure 6iii, we see that this strategy provides a clever way to reach a high reward state. The initial radial 'push' at t = 0s imparts enough energy to the object that it reaches near the edge of the container. If the capillary forces are not strong enough, the reverse stroke then pulls the object back slightly while moving the object in the clockwise direction. This process is repeated at the 3.5 time period mentioned before, leading to an oscillatory motion. Even if the object reaches the edge of the container in the first push, the reward values are relatively high, leading to a very reliable control strategy.

The actions proposed for the first three tasks rely largely on oscillations in the x & y directions, setting z oscillation frequencies to be comparatively small. Assuming that the x & y vibrations dominate the controller's behaviour, the periods of oscillation may be predicted:

$$T_{pred} = \frac{1}{|f_x - f_y|} \tag{1}$$

giving predictions of 72.3s, 53.0s, and 3.4s for Tasks i, ii, and iii respectively. These align well with the observed periodicities, which were extracted from video footage. Discrepancies are most likely due to approximations made during these observations, though may be in part due to the two axis assumption and limited processor speed of the robotic arm.

Task i's paths experience, on average, 19% of a full period before completion, which is displayed by the somewhat erratic nature of the repeated paths: since the flow is not stationary in time, the object will not always experience the same flow at a given point, and small temporal differences between repetitions lead to differences in the final paths. Conversely, Task ii's paths experience only 8% of a full period, and we do not see such shifts in behaviour. The actions proposed for Tasks iv & v are both dominated by y oscillations, such that Equation 1 gives expected periods <1s, and the observed flows are stable.

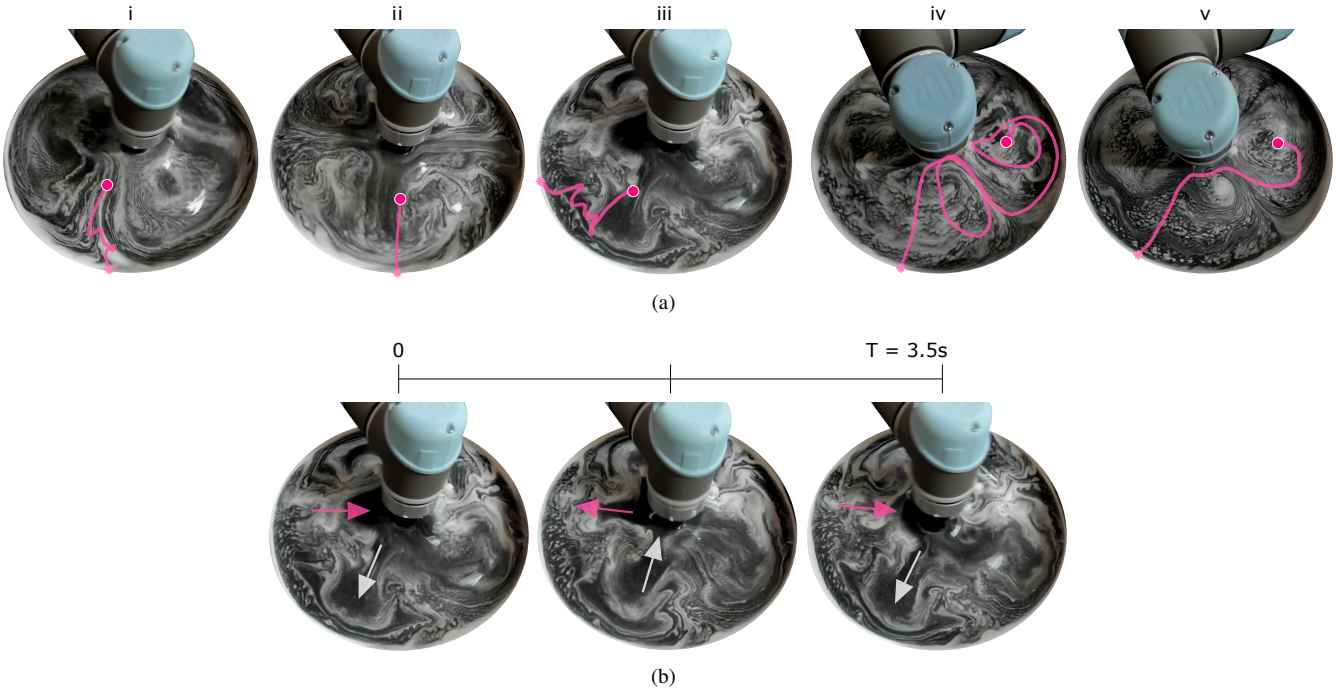

Figure 7. (a) Visualisations of the surface flows generated for each of the five tasks using glass microsphere tracer particles, each overlain with their characteristic paths. (b) Task iii's periodic flow pattern, with the directions of two major flow oscillations indicated with arrows.

Finally, we evaluate the transferability of the DRL solutions to different geometries of the floating object, introducing 6 new shapes: 8mm, 46mm, & 72mm circles, 22mm & 35mm squares, and a 25mm quatrefoil. Each is tested with Figure 6's tasks & final actions, with no additional training to account for the change from the original 20mm circle. As the surface flows generated by the robotic arm are not significantly affected by the free-floating object, one can expect the transferability of the control policy to different objects to be reasonably good. As such, the tests provide an insight into the morphological attributes which determine their behaviours in these surface flows. No shape performs as well as the 20mm circle over all tasks, with changes in size affecting the physical area which can be navigated (unsurprisingly, the 72mm circle's motions are dominated by early edge hits), and changes in shape affecting the relative effect of the capillary and flow forces: the flat edged squares often remain in a stable boundary offset by approximately 15mm from the container's perimeter, with a flat edge aligned with the wall. The quatrefoil is chosen as an intermediate between the square and circular shapes, and can often be seen behaving as such: though it is capable of settling into stationary points offset from the container's edge, it does so much less often than the 35mm square. The 8mm circle's behaviour appears comparatively erratic throughout Tasks i & iii, which we suggest is caused by its tendency to be caught in local surface flows otherwise overcome by nearby forces in the larger shapes. The simplicity of the radial flow in Task ii does not provide much of a chance for such deviations to occur, whilst the stationary flows of Tasks iv & v make the object's behaviour much more repeatable, even if low scoring. Though Task i shows the disadvantage of reliance on the periodic flow - if a differently shaped object does not hit the edge after following the expected path, it has the potential to be carried far from the target in the shifting flow - clear evidence of Figure 6's paths can be seen in all of the newly introduced morphologies, suggesting that their controllers could be quickly adapted by using another morphology's agent as a starting point for training. A further dimensional analysis of the solution's transferability between problem geometries is presented in the supplementary material, where clear dependence of the solutions on the container radius and invariance to the fluid depth is observed for our problem. Such post-learning analyses can further help understand the role of the independent variables and device better experimental protocols.

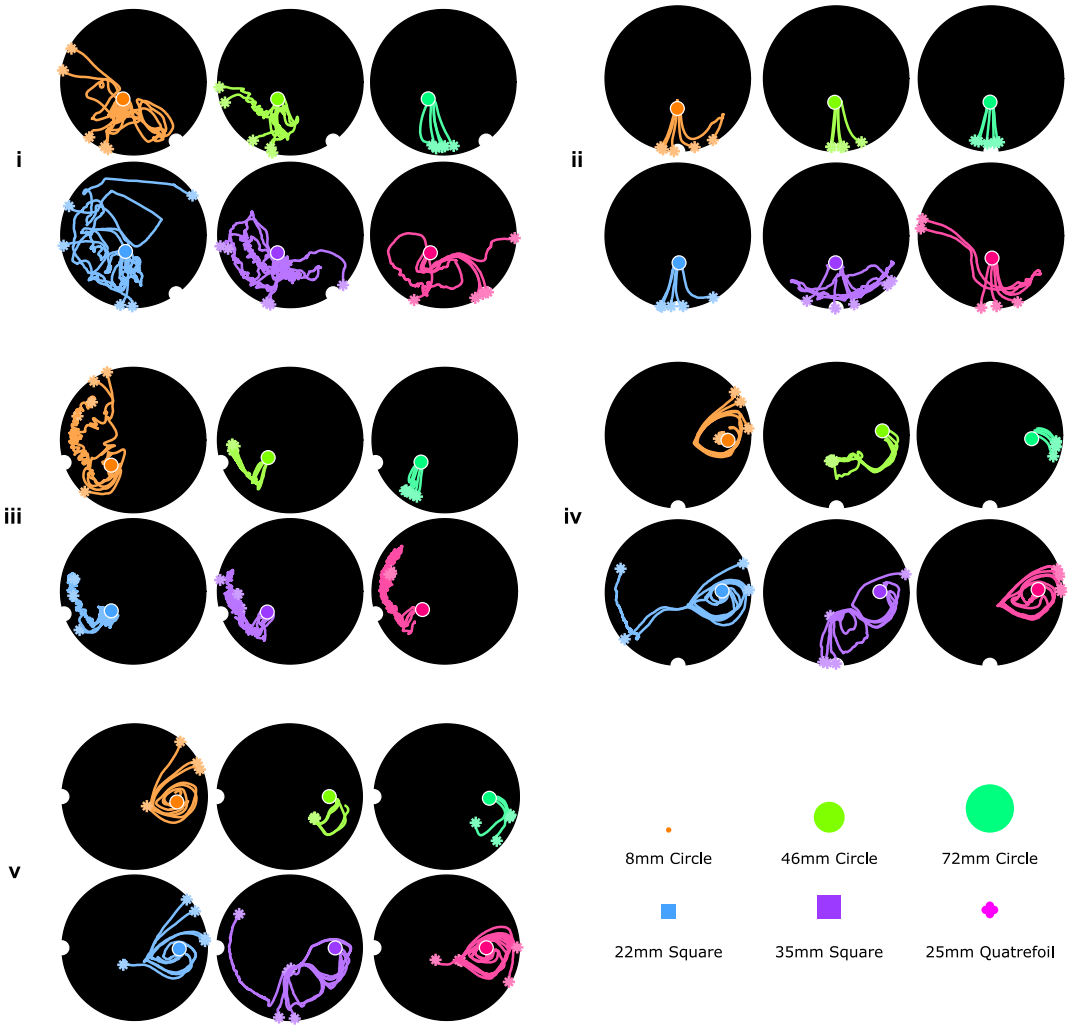

Figure 8. Transferability of the learned agent to a new set of shapes for the 5 example tasks. Each shape is tested 5 times without any additional training.

Discussion

This study shows that deep learning techniques can indeed be used to develop an effective controller in the face of nonlinearities, stochasticity, and non-convexity. The delivery task's open-loop implementation and underactuation can be addressed through the large scale experimentation made possible by the autonomous robotic system. We show how reward shaping based on the problem characteristics can be used to reduce the learning period significantly. More importantly, we show that this seemingly complex problem can be solved with reasonable robustness in spite of non-stationary and turbulent surface flows. The generated control solutions rely on attracting or repelling Lagrangian coherent structures to develop repeatable object trajectories even with uncertainities in the system state and the turbulent nature of the surface flows. We show that post-learning, the control solutions and environmental parameters can be further analyzed and studied leading to a better understanding of the underlying physics and the role of the independent variables for the optimal control solutions.

The proposed DRL architecture is highly desirable for this control task as even unsuccessful trials can be used to improve the critic and to aid exploration. Using insights from the physical setup, we further reduce the training time by reward shaping. We do this by encouraging outward radial motions in the early stages of training, avoiding bifurcations and non-repeatable behaviors. Fine tuning of the control policy is then done to improve the overall behavior. A high performing agent was obtained with our proposed methodology only after 5000 iterations. The total training time has upper bound of 3 minutes per iteration (2 minutes for uncompleted behaviours plus one for the reset), giving an upper bound of 10.4 days of training for 5000 iterations. However, because of our reward's shaping, the majority of attempts would hit the edge and stop early, thus not rely on the temporal stopping criterion. In the later stages of training, more than 1000 iterations could be averaged every day, thereby

providing decent solutions to the problem in reasonable times, a challenge with current physical experimentation platforms[14]. Multiple setups would permit parallel optimisations and further reduction in training time. This could also aid in finding policies that are more robust due to unavoidable errors incurred while replicating the setup[24]. Note that the number of iterations taken for the DDPG agent to learn a global controller is only an order higher than the iterations used by the Bayesian Optimization for a single target and starting location. This is because the DRL agent can utilize even information from unsuccessful trials to update its internal models through the critic. Furthermore, the DRL agent can exploit patterns in the value function and control solutions to extrapolate between unvisited states and targets.

Techniques for object manipulation using localized energy sources would still provide better controllability and accuracy over the proposed remote control strategy. However, remote manipulation is still the only available solution for certain applications. Manipulation by propagating waves are highly desirable in real-world scenarios where there are restrictions on the environment and target objects. On-surface and underwater robotic manipulation is the most straightforward application similar to what is observed in nature, where current robotic technologies rely on direct contact with the target object[25–27]. The open-loop implementation greatly simplifies the sensory and on-board processing requirements. The transferability of the learned agent promises a practical approach to translate pre-learned controllers to changing environments and objects with minimal adaptation.

Methods

Experimental Setup

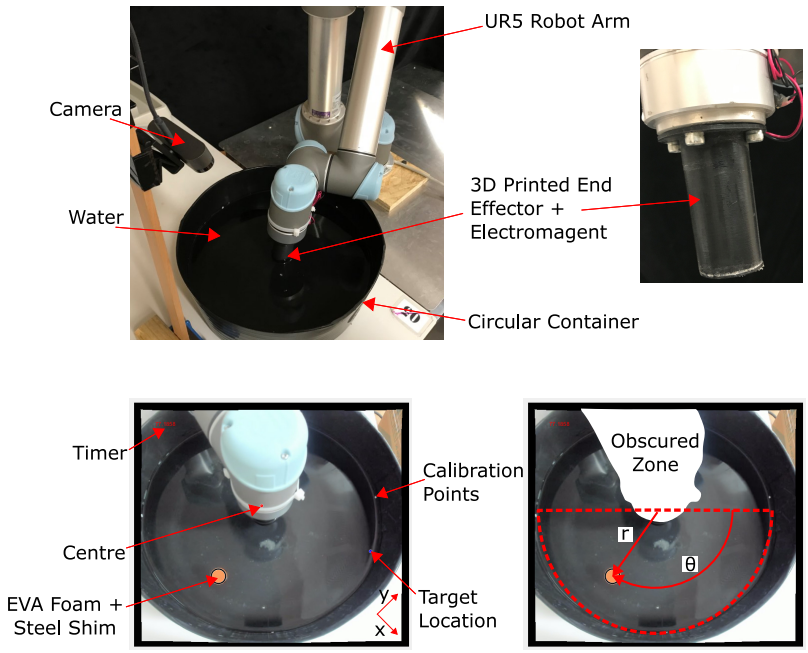

Figure 9. Top: Experimental setup. Bottom: A typical view returned from the camera, indicating the obscured zone and the operating semicircle used by the BO and DRL methods.

The experimental setup consists of a tapered circular container filled with 8L of water, with 85mm water depth and 360mm surface diameter (Figure 9). Surface waves are generated from the container's centre using a UR5 robotic arm equipped with a submersible end effector: a hollow PLA cylinder with 40mm diameter and 80mm length. An internal 12V electromagnet is configured as a digital tool output of the arm such that, by holding the end effector above the water's surface and powering the electromagnet, local magnetic objects on the water's surface can be attracted and moved to a desired position. This forms the basis of the reset mechanism which allows the setup to autonomously run for extended periods of time: the floating object is comprised of two layers of buoyant EVA foam sandwiching a circular steel shim, and can be retrieved and repositioned from any location in container using the electromagnet. To prevent the object from adhering to the end effector's underside during this procedure, hot melt adhesive is used to texture this surface, limiting the contact area of the two.

To facilitate these resets, a webcam is positioned above the container, feeding live data to the PC which controls the robotic arm via Python. The bright orange of the EVA object against the dark background of the container makes it straightforward to apply a binary mask and to identify the circular object using *MATLAB*'s image processing toolbox. Prior to this, the camera

is calibrated to eliminate distortions in the water surface plane. Small white calibration points marked on the inside of the container are used to locate the surface's centre and return locations in polar coordinates, where $\theta = 0$ is aligned with the camera's horizontal orientation and the z direction is defined as perpendicular to the plane of the water.

Figure 1 & 7's flow visualisations are achieved by coating the surface of the water with a layer of K20 glass microspheres before the vibrations begin. The webcam's output is then fed into *MATLAB*'s `estimateFlow` function to create Figure 1's vector field.

Control Space

The DDPG agent takes three inputs: the object's starting position in 2D polar coordinates, and the target periphery position, which is expressed using its polar angle. Given this input state, an 11 dimensional action is returned, defining the vibrations of the end effector for each iteration. Table 1 describes the 11 control parameters and their minimum/maximum limits. A sine wave's mean (depth of submersion), amplitude, and frequency are defined in the z direction, perpendicular to the plane of the water. The x and y axes are are each associated with a vibration frequency, amplitude, and phase (relative to the z vibrations), with zero mean. Using d to represent the depth of submersion, the position $\mathbf{p}$ of the end effector's tip can then be described as:

$$\mathbf{p} = \begin{bmatrix} A_x \sin(2\pi f_x t + \phi_x) & A_y \sin(2\pi f_y t + \phi_y) & A_z \sin(2\pi f_z t) - d \end{bmatrix}^T \tag{2}$$

Where, $\mathbf{p} = \mathbf{0}$ corresponds to the center position at the surface of the water. In addition, small rotations of the end effector around its tip are defined along the x and y axes, described in Table 1 by parameters θ_x & θ_y respectively.

Table 1. The 11 control variables which parameterise the end effector's vibrations.

	$\mathbf{A_x}(mm)$	$\mathbf{f_x}(Hz)$	$\phi_\mathbf{x}(^\circ)$	$\theta_\mathbf{x}(^\circ)$	$\mathbf{A_y}(mm)$	$\mathbf{f_y}(Hz)$	$\phi_\mathbf{y}(^\circ)$	$\theta_\mathbf{y}(^\circ)$	$\mathbf{A_z}(mm)$	$\mathbf{f_z}(Hz)$	$\mathbf{d}(mm)$
Min.	0	0	0	-10	0	0	0	-10	0	0	5
Max.	4	5	360	10	4	5	360	10	4	5	45

The lower half of Figure 9 depicts a typical view returned from the camera during the training stages. The interface displayed during each iteration marks the container's centre, the object's current position and target position, and displays the time elapsed since motion began. Given the development of an open-loop controller, this data is not fed back to the agent during each iteration, but is used after its completion to evaluate performance and allocate a suitable reward function. The only exceptions are the stopping conditions: an iteration is ended if the identified location reports that the object has hit the container edge, where capillary forces will retain it, or if two minutes have elapsed and the edge has not been reached. Before each iteration, a starting position and target location within Figure 9's semicircle are randomly selected, and the object is positioned using the end effector's electromagnet. The polar coordinate frame, common to the UR5 and vision system, is also marked. The ranges of the starting radius ($40 \rightarrow 100$mm) and starting/target angles ($0^\circ \rightarrow 180^\circ$) are linearly normalised to the range (-1,1) throughout experimentation.

BO & DDPG Implementation

The input states do not change between Bayesian iterations, with each optimisation process focusing on developing the best 11 vibration parameters for one of the three rotation cases. Following the completion of each iteration, a cost function uses the tracked path to sum the distance from the target and the time elapsed:

$$f = l_{min} + t_f \tag{3}$$

where t_f indicates the time in seconds before either stopping criterion is fulfilled, and l_{min} is the shortest distance recorded in mm between the object's position and the target location over the whole trajectory.

For training of the DDPG agent, the naive reward function focuses only on minimising the distance:

$$R(l_{min}) = 3.45 \left[\exp\left(\frac{-l_{min}}{k_l} \right) \right] - 1.15 \tag{4}$$

where k_l = 165mm. Perfect alignment of the two corresponds to a maximum reward of 2.3, decaying exponentially with l_{min} towards a minimum of -1.15. The training is halted after 4151 iterations, when little improvement is seen in the controller.

The shaped reward function consists of the sum of two Gaussian functions and a small time dependence:

$$R(l_{min}, \Delta\theta, t_f) = \exp\left(-\frac{\Delta\theta^2}{2{\sigma_\theta}^2} \right) + \exp\left(-\frac{{l_{min}}^2}{2{\sigma_d}^2} \right) - \frac{t_f}{k_t} + 0.5 \tag{5a}$$

in which $\sigma_\theta = \pi/3$, $\sigma_d = 50$mm, & $k_t = 120$s. $\Delta\theta$ is the angle between the target location and the object's final location, allocated only if the positional stopping criterion is fulfilled. If the temporal stopping criterion has instead been fulfilled, then the first Gaussian function is instead replaced with a negative reward:

$$R\left(l_{min}, t_f\right) = -1 + \exp\left(-\frac{{l_{min}}^2}{2{\sigma_d}^2}\right) - \frac{t_f}{k_t} + 0.5 \tag{5b}$$

The combination of the reward described in 5a & 5b specifically discourages behaviours of the object which do not reach the container's edge within the 2 minute time period (referred to as 'uncompleted' behaviours), whilst the function tends smoothly to a maximum of 2.25 amongst behaviours which reach the edge ('completed'). The minimum reward tends to -1.25, for uncompleted behaviours which remain far from the target location throughout. Training was terminated after 5000 iterations.

The passing of the object into Figure 9's obscured zone is not explicitly discouraged by the cost and reward functions. Paths which pass through the zone and regain camera visibility are not penalised, but those which reach the edge in this region are considered uncompleted during the reward allocation. In this way, these paths pose a risk to the agent which does not exist when the zone is avoided. Coupled with the initialisation of all starting positions and target locations into Figure 9's operating semicircle, behaviours rarely rely on motion within this zone.

The controller's optimisation and learning processes are implemented using functionalities provided by *MATLAB*'s Statistics and Machine Learning and Reinforcement Learning toolboxes. The sequential Bayesian optimisations of Figure 2 all use the same parameters, and are halted after 680, 635, & 677 iterations respectively. A 0.5 exploration ratio, Gaussian active set size of 300, and expected-improvement-per-second-plus acquisition function are used. The optimisations begin with no prior knowledge of the system's setup or nonlinear dynamics, and only receive updates through the action-in/cost-out black box implementation.

The trained DDPG agent makes use of two deep neural networks for its actor and critic architectures. Between the actor's 3-node input layer and its 11-node output, 5 hidden layers are implemented: a fully connected (FC) layer with 50 nodes is followed by a rectified linear unit (ReLU), another 50-node FC layer and ReLU, and a final FC layer of the same size as the action space dimensionality. When Reward Function 2 is used, this size is 11 (Table 1), whilst Reward Function 1 uses a 12th decay parameter D (ranging from 2 to 502) to linearly reduce the amplitude parameters at each time step:

$$A_x(t) = \max\left[0, \left(1 - \frac{t}{D}\right)A_x(0)\right]; \quad A_y(t) = \max\left[0, \left(1 - \frac{t}{D}\right)A_y(0)\right]; \quad A_z(t) = \max\left[0, \left(1 - \frac{t}{D}\right)A_z(0)\right] \tag{6}$$

The decay parameter was not used for the second agent as it did not seem to improve the performance of the controller. The critic has two input layers, for the state and proposed action. Each is initially processed separately: the state is passed through a 50-node FC layer, ReLU & a second 50-node FC layer, whilst the action is fed into a single 50-node FC layer. An addition layer then combines the two, before a ReLU & FC layer reduce the output dimensionality to 1: the reward estimate (See supplementary for more details).

References

1. Ding, X. *et al.* On-chip manipulation of single microparticles, cells, and organisms using surface acoustic waves. *Proc. Natl. Acad. Sci.* **109**, 11105–11109 (2012).
2. Collins, D. J. *et al.* Two-dimensional single-cell patterning with one cell per well driven by surface acoustic waves. *Nat. communications* **6**, 1–11 (2015).
3. Zhang, P. *et al.* Generation of acoustic self-bending and bottle beams by phase engineering. *Nat. communications* **5**, 1–9 (2014).
4. Punzmann, H., Francois, N., Xia, H., Falkovich, G. & Shats, M. Generation and reversal of surface flows by propagating waves. *Nat. Phys.* **10**, 658–663 (2014).
5. Zhou, Q., Sariola, V., Latifi, K. & Liimatainen, V. Controlling the motion of multiple objects on a chladni plate. *Nat. communications* **7**, 1–10 (2016).
6. Ahmed, D. *et al.* Neutrophil-inspired propulsion in a combined acoustic and magnetic field. *Nat. communications* **8**, 1–8 (2017).
7. Moffitt, J. R., Chemla, Y. R., Smith, S. B. & Bustamante, C. Recent advances in optical tweezers. *Annu. Rev. Biochem.* **77**, 205–228 (2008).
8. Tasoglu, S., Diller, E., Guven, S., Sitti, M. & Demirci, U. Untethered micro-robotic coding of three-dimensional material composition. *Nat. communications* **5**, 1–9 (2014).

9. Colombi, R., Schlüter, M. & von Kameke, A. Three dimensional flows beneath a thin layer of 2d turbulence induced by faraday waves. *Exp. Fluids* **62**, 1–13 (2021).
10. Francois, N., Xia, H., Punzmann, H., Ramsden, S. & Shats, M. Three-dimensional fluid motion in faraday waves: creation of vorticity and generation of two-dimensional turbulence. *Phys. Rev. X* **4**, 021021 (2014).
11. Francois, N., Xia, H., Punzmann, H. & Shats, M. Rectification of chaotic fluid motion in two-dimensional turbulence. *Phys. Rev. Fluids* **3**, 124602 (2018).
12. Lifshitz, R. & Petrich, D. M. Theoretical model for faraday waves with multiple-frequency forcing. *Phys. review letters* **79**, 1261 (1997).
13. Li, X., Li, X. & Liao, S. Observation of two coupled faraday waves in a vertically vibrating hele-shaw cell with one of them oscillating horizontally. *Phys. Fluids* **30**, 012108 (2018).
14. Fan, D. *et al.* A robotic intelligent towing tank for learning complex fluid-structure dynamics. *Sci. Robotics* **4** (2019).
15. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *nature* **521**, 436–444 (2015).
16. Brunton, S. L., Noack, B. R. & Koumoutsakos, P. Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech.* **52**, 477–508 (2020).
17. Mohan, A. T. & Gaitonde, D. V. A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks. *arXiv preprint arXiv:1804.09269* (2018).
18. Kawaguchi, K. & Bengio, Y. Depth with nonlinearity creates no bad local minima in resnets. *Neural Networks* **118**, 167–174 (2019).
19. Viquerat, J. *et al.* Direct shape optimization through deep reinforcement learning. *J. Comput. Phys.* **428**, 110080 (2021).
20. Rabault, J., Ren, F., Zhang, W., Tang, H. & Xu, H. Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape optimization. *J. Hydrodyn.* **32**, 234–246 (2020).
21. Garnier, P. *et al.* A review on deep reinforcement learning for fluid mechanics. *arXiv preprint arXiv:1908.04127* (2019).
22. Rabault, J., Kuchta, M., Jensen, A., Réglade, U. & Cerardi, N. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *J. fluid mechanics* **865**, 281–302 (2019).
23. Lillicrap, T. P. *et al.* Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).
24. Tobin, J. *et al.* Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 23–30 (IEEE, 2017).
25. Stuart, H., Wang, S., Khatib, O. & Cutkosky, M. R. The ocean one hands: An adaptive design for robust marine manipulation. *The Int. J. Robotics Res.* **36**, 150–166 (2017).
26. Han, D. *et al.* Soft robotic manipulation and locomotion with a 3d printed electroactive hydrogel. *ACS applied materials & interfaces* **10**, 17512–17518 (2018).
27. Lane, D. M. *et al.* Amadeus: advanced manipulation for deep underwater sampling. *IEEE Robotics & Autom. Mag.* **4**, 34–45 (1997).

Acknowledgements

This work was supported by the SHERO project, a Future and Emerging Technologies (FET) programme of the European Commission (grant agreement ID 828818), and by EPSRC DTP EP/R513180/1.

Author contributions statement

Must include all authors, identified by initials, for example: A.A. conceived the experiment(s), A.A. and B.A. conducted the experiment(s), C.A. and D.A. analysed the results. All authors reviewed the manuscript.

Additional information

The authors declare no competing interests.