Learning to maximize the swimming speed with deep Reinforcement Learning

1st Israilov Sardor
I3S-CNRS, Université Côte d'Azur, France

***Abstract*—Aquatic animals such as fish evolved to undulate their bodies to efficiently navigate through water. Extensive research on natural and synthetic aquatic swimmers found the relationship between body oscillation kinematics and swimming speed. Yet an open and challenging inquiry remains: determining the optimal kinematics to maximize swimming speed within specific constraints and understanding how a system can modify its control parameters to achieve this optimality. In this study, we employ learning algorithms on a biomimetic robotic fish closely resembling real fish swimming and exhibiting undulatory motion. The obtained results indicate that there is an optimal frequency and mode of actuation of a robotic fish, explained by the interplay between the swimmer dynamics and the fluid-structure interaction with the surrounding fluid. We provide a novel methodology and framework for finding an optimal swimming gait for maximizing the speed of a robotic fish, using images from top-mounted camera via deep Reinforcement Learning. Additionally, we present a simplified numerical model that approximates experimental results, based on first physical principles.**

***Index Terms*—deep reinforcement learning, robotic fish swimming, optimization, biomimetics, fluid dynamics, computer vision**

I. Introduction

Fish locomotion remains a complex topic due to intersection of biology and fluid dynamics. Acquatic locomotion and control mechanisms have been refined through million of years due to Darwinian evolution [11]. Through years of evolution, acquatic animals have developed their biological mechanisms in order to survive, hunt and reproduce. Most of the swimming is undulatory, where the thrust is produced by the body undulation pushing water [5]. The kinematics of acquatic swimming demonstrate remarkable consistency across vertebrates, highlighting universal physical principles. Typically, the wavelength of body deformation is comparable to the swimmer's length, while the tail beat amplitude is about 0.2 times the swimmer's length. The tail beat frequency (f) varies with the individual, adjusting to control swimming speed. A higher frequency results in greater speed [1], [7], [31], [34], [43]. Swimmer's can adjust the frequency of tail undulation within a specific range limited by biological muscles and fluid dynamics of their interaction with the surrounding water. Muscles have limitations regarding contraction speed and tension [12]. The limits can also be imposed by the decision processes that are spontaneous, conscious or through proprioceptive reflexes. These considerations are reflected in the design of biomimetic robotic swimmers [21], [22], [32], [41], [59].

On the one hand, fish have evolved fabulous adaptations to their habitat, exhibiting diversity in shapes and physiologies that inspire the development of new technologies. On the other hand, integrating biomimetic features into aquatic robotic systems creates favorable opportunities for enhancing the understanding of fish locomotion. Maneuverability, efficiency and fish-like appearance distinguish robotic fish from other underwater robots. The applications of a robotic fish hold a big promise. Some prominent applications include marine life observation [18], where robotic fish exhibit fish-like behavior without disturbing aquatic world or navigating hardly-accessible places due to compact fish-like shape. Robotic fish can also lead real fish school from dangerous zones [29].

The development of efficient and fully autonomous robotic fish requires identifying control strategies achieving desired performance under given constraints. The most classical control approaches applied on a robotic fish include PI [54], PID [49], and robust controllers [53] have been employed to improve trajectory tracking performance. Other control strategies include artificial Central Pattern Generators (CPG) [44], [46], [50]–[52] to propel robotic fish, and fuzzy logic [14] to follow the target. Because of the complex nonlinearity of a robotic fish swimming, recently, there has been a growing interest in applying Reinforcement Learning (RL) for different control tasks [4], [30], [45], [55], [58]. Most of this research work had predefined patterns (harmonic movement or CPG) and optimized some parameters of the predefined control pattern via Reinforcement Learning. While certain research have examined the relationship between swimming speed and tail beat frequency, no conclusion has been made on the optimal control necessary to achieve the highest speed. In this article, we aim to find new insights on the optimal swimming gait of robotic fish to maximize the swimming speed.

Recently, a number of simulation models have been created to study fish locomotion. In [26], authors propose a set of physics-based environments for different modes of fish locomotion integrated in openai gym deep RL framework. In [17], the authors implemented a PPO agent to drive a simulated three-linked fish to a desired location in 2D in a potential flow, utilizing the sense of proprioception to reach the destination. In [47], deep RL was employed to propel the fish in CFD simulation. In [40], the authors present the general set of fish control benchmarks based on the popular physics-based simulation engine, mujoco [42]. Later, we present the modified "fish" environment from this control suite benchmark to accomplish our goal of maximizing the swimming speed in

1D.

In [56], [57] researchers implemented sim2real approach to achieve a path-following control: a policy trained within a data-driven/CFD simulation was applied on a robotic fish in a real-world setting. In [58], sim2real approach was applied on a robotic fish to adjust the attitude angle related to the water flow using RL. While sim2real approach is robust, it is sub-optimal due to the "model bias" experienced in simulation. Latest advances in model-free deep RL have significantly improved the speed of learning and permit to apply these methods directly on robots without any prior modelization [6], [9], [15], [36], [37]. Here are the key contributions of this paper:

1) physics-based simulation environment for thrust and speed maximization with an experimental setup
2) To the best of our knowledge, we present the first work applying deep RL successfully for fish locomotion task without prior modelisation. The RL session is capable of finding the best control frequency. Both simulation and experimental validation prove that bang-bang control is optimal.

In the following, we first present RL concepts and some algorithms in section II. Then, section III presents experimental setup and optimization configuration. We present simulation and experimental results in section IV. We then conclude with discussion.

II. Deep Reinforcement Learning

Reinforcement Learning (RL) is a control framework for discretized dynamical systems. RL exploits the structure of Markov Decision Process (MDP), which is a discrete-time stochastic control process. At the discrete time step i, there are four components in MDP (S, A, P, R): a set of states $s_i \in \mathcal{S}$ of the dynamical system, a set of available actions $a_i \in \mathcal{A}$ that the system can actuate, unknown dynamics that define transition probability among the states $P \in [0, 1] : s_i \times a_i \rightarrow s_{i+1}$ and set of emitted rewards $r_i \in R$ on each transition (s_i, a_i, s_{i+1}). We refer to the internal decision maker i.e an RL algorithm as an agent, and the whole physical system as an environment. At each time step i, the agent (algorithm) chooses an action a_i according to the assessed current state s_i. After the actuation, the environment provides a new state s_{i+1} and a reward r_{i+1}. The reward r_{i+1} serves as a feedback signal to the agent, indicating the efficacy of the taken action a_i at state s_i.

During the learning process, the agent evolves in the environment and tries to maximize its cumulative reward R_τ during certain interaction time horizon $[0, T]$, which is referred as an episode or a trajectory τ:

$$\begin{aligned} \tau &= (s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}, s_T) \\ R_\tau &= \sum_{i=0}^{T-1} \gamma^i r\left(s_i, a_i\right), \end{aligned} \tag{1}$$

where $0 < \gamma < 1$ is the discount factor which measures the importance of the future unitary reward and serves as a bias-variance trade-off for R_τ.

We focus on actor-critic methods [28], [39] represented by neural networks, which alternate between policy evaluation and improvement. The actor represents the policy $\pi(a|s)$ and outputs an action a from state s. The critic estimates cumulative reward for different possible actions $(Q(s, a))$. During policy improvement, the critic is trained via the following loss function: $J(\theta) = \mathbb{E}\left[\left(r(s_i, a_i) + \gamma \max_{a_{i+1}} Q_{\theta'}(s_{i+1}, a_{i+1}) - Q_\theta(s_i, a_i)\right)^2\right]$. We use DROQ [13] for a continuous action setting and PPO [35] in a discrete action context.

Sample-efficient deep RL: DROQ

DROQ [13] is highly sample-efficient RL due to the high update-to-data (UTD) ratio of critic. Previous off-policy RL methods such as SAC [10] and DDPG [25] use UTD ratio of 1 because using high UTD ratio is prone to overfitting and requires regularization [23]. To become more sample-efficient, DROQ performs multiple gradient steps on a critic for every one environmental step taken. This is supported by [6] that demonstrates that increasing UTD ratio drastically improves sample-efficiency. DROQ is an extension of REDQ [3] that uses high UTD ratio alongside with ensemble of critics as from of regularization. To adress overfitting, DROQ incorporates dropout regularization [38] on the RL algorithm's critics. We implement an asynchronous actor-critic setup in for the agent to learn from off-policy replay buffer while interacting with the environment.

III. Experimental setup and design choices

Experimental setup description

In our experiments, we use a 3D printed robotic fish rear body described in [8], [32] and represented in fig. 1b and c. Fish body features a flexible skeleton with an attached fin at its end, fabricated through 3D printing with a flexible polymer. Controlled deformation is achieved using a waterproof servo-motor (Hitec HS-5086WP), which is linked to the end of the skeleton via two cables. The rigid head, fabricated from PLA via 3D printing, incorporates an additional mass of 70 grams to ensure stability and prevent oscillations of the robotic fish when moving the tail. The robotic fish is attached to plastic tube. The robotic fish is connected to a floating plastic tube (Fisherbrand^{TM} 500ml with 53mm diameter), within which all the electronic components are located. Having a floating tube on the water surface allows to communicate with the robot via wi-fi and alleviates the difficulty of the third dimension for fish swimming. A "Raspberry pi Zero WH" located in the tube receives the commands from a PC and controls the servomotor. We opt to control a robotic fish wirelessly, thus avoiding dry friction and hanging wires that could potentially impact the optimal swimming gait we aim to find. Stationary PC (Alienware-Aurora-R11 with Intel(R) Core(TM) i9-10900F CPU @ 2.80GHz and GeForce RTX 3080 GPU), performing gradient-based learning, communicates with a robotic fish via Wi-Fi protocol. To reduce the possibility of harmonic pollution, we use separate power supplies for the servo-motor and the main board. We employ a block of four (1.5 V,

3000mAh) rechargeable batteries to power the servo-motor, while a separate battery (3.7 V, 1200 mAH) powers the main card through the "PiJuice Zero" power module. All the experiments are conducted within the water tunnel of Rolling Hills Research Corporation.

RL learning setting

The objective is to swim against a water flow with the maximum speed possible. The water flow is regulated to maintain a velocity of 60 mm/s, opposing the fish's swimming direction. During the training phase, this specific water flow rate serves a dual purpose. Firstly, it provides a consistent opposing force for the robotic fish, effectively simulating the challenges of swimming upstream. The increased drag force ensures that only effective control policies are able to propulse the robotic fish forward. Secondly, the 60 mm/s flow rate facilitates a rapid and efficient RL environment reset to bring the fish to the initial position once a learning episode is terminated. The learning episode is ended by halting the fish and fish going backwards with the flow of the water tank. To prevent the fish from becoming immobilized during the way back due to friction along the reservoir's boundaries, fish undulates its tail slightly every three seconds. Once the camera detects a certain position threshold near the start is overcome, the fish restarts the new episode again. Small 3D-printed blue piece of a "triangular" shape is mounted at the beginning of the track, so that the robotic fish starts the episode in a consistent position relative to the reservoir.

An overhead web-camera (Razer Kiyo X) captures the robotic fish movements [see example image in fig. 1a]. Learning from raw images can be hard and time consuming due to the high-dimensionality of images [20]. To simplify and accelerate the learning process, we implement the marker-based approach of extracting useful features from images [27] by affixing yellow and red markers to the buoyant cylinder. One marker suffices to determine an object's position precisely; however, inferring the orientation of a robot requires at least two markers. This approach eliminates the use of high-dimensional input images and utilizes useful and latent information directly in RL algorithms. To detect the points, we convert the RGB image to HSV color space, and then color thresholding combined with noise filtering through image erosion followed by dilation. The servomotor consign angle ϕ_c is concatenated with extracted positions of two points, forming a 'base' observation vector. This adds the estimation of the fin deflection angle. To deal with the partial observability of the system, four consecutive 'base' vectors form a state for an RL agent. So, the resulting observation vector at time step i consists of $[\phi_{ci}, cx_i, cy_i, px_i, py_i] \times 4$, where (cx_i, cy_i) are the coordinates of one marker in 2-dimensional space at the time step i and (px_i, py_i) are the coordinates of the second point.

The agent-environment interaction during learning occurs in episodes spanning 128 time steps, each with a sampling interval of 50 ms. The episode time of 128 time steps ensures that the robotic fish always stays in the camera vision field and the 50 ms sampling time guarantees consistent Wi-Fi data exchanges. The episode stops upon reaching the maximum time step count. To be robust to white noise and perturbations, the reward at each time step is defined as a moving-average velocity ($\bar{\dot{x}}$) over $1s$ time interval in the x direction [see fig. 1a].

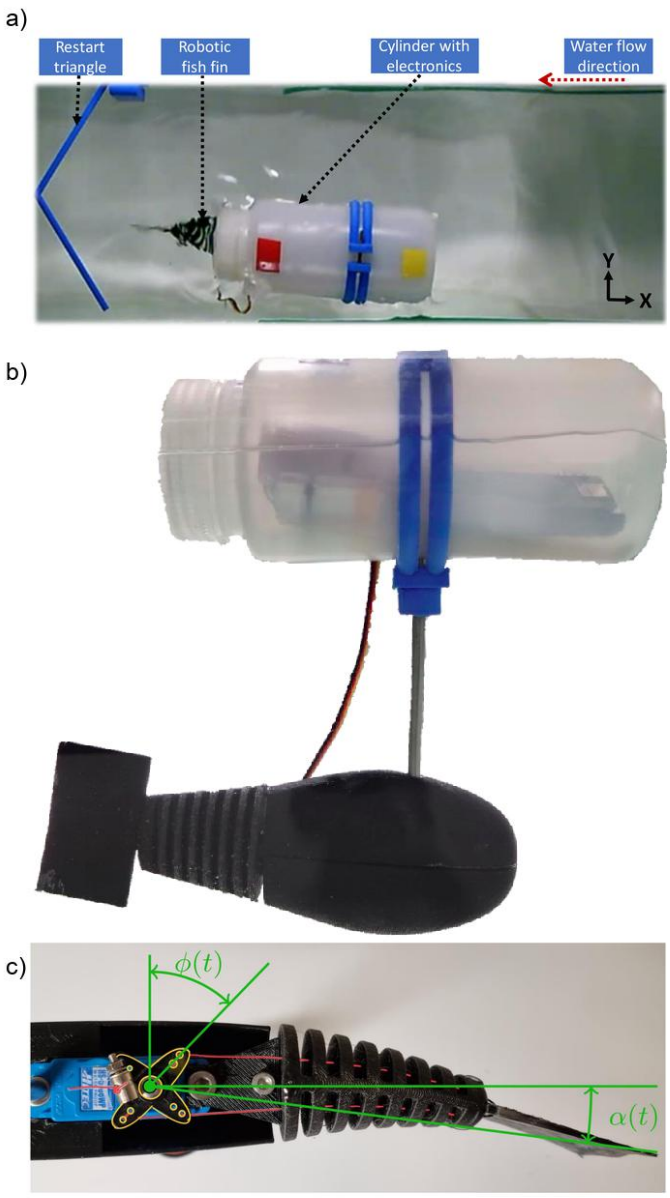

Fig. 1. Experimental setup for a robotic fish swimming in a water tunnel. a) View on the fish from above. The use of two markers permit to define the position of the swimmer, as well as its body angle orientation with respect to the flowing water direction. b) Side view of the setup: floater with electronics, servomotor and elastic fin. c) Zoom on inside of a robotic fish. Servomotor actuates the elastic fin of a robot fish via the fishing wires, which in turn propels the entire system against the direction of the flow.

IV. Results

A. Simulation

There is another way to fit ALL parameters through bayesian optimization. same principle for fitting hyperparameters of deep learning algorithms [look optuna example]. it may fit better, but the resulting parameters may not have a physical sense. Before moving with experiments, we construct an approximate physical model for testing different modes of learning and tuning hyperparameters. We hereby aim to optimize the simulated robotic fish swimming speed in 1D and follow OpenAI Gym framework for learning [2].

Simulation model of fish swimming: In this simulated environment, the equations governing the motion dynamics of the robotic fish rely on the models of fish locomotion described in [32], [33]. Servomotor control is defined as a consign angular position of a servomotor $\phi_c \in [-\Phi, \Phi]$, where Φ is a maximum instruction angle. A servomotor tries to adjust its real angle ϕ to the consign ϕ_c and its internal dynamics are described in eq. (2). The servomotor head is attached tightly to the fish fin via two elastic threads. This attachment is characterized by linear coefficient $\lambda = 0.46$. The dynamics of the fish tail flapping is modeled as a damped harmonic oscillator in eq. (4) with ξ a dissipation coefficient and ω_0 a proper frequency. The full derivation of these equations is available in [32], [33]. When fish undulates its body, it pushes water and produces thrust driving itself forward. The fish movement is governed by eq. (5), where the fish body is affected by the thrust force $(-C_t\ddot{\alpha}\alpha)$ and the drag $(-C_d|\dot{x}|\dot{x})$.

$$\dot{\phi}(t) = \Omega \tanh\left(\frac{1}{\Delta}\left(\phi_c(t) - \phi(t)\right)\right), \tag{2}$$

$$\alpha_c(t) = \lambda\phi, \tag{3}$$

$$\ddot{\alpha}(t) + \xi\omega_0\dot{\alpha}(t) + \omega_0^2\left(\alpha(t) - \alpha_c(t)\right) = 0, \tag{4}$$

$$\ddot{x} = (-C_d|\dot{x}|\dot{x} - C_t\ddot{\alpha}\alpha)/m. \tag{5}$$

The process of determining parameters through fitting is carried out using both linear regression and nonlinear fitting techniques, specifically employing the *lmfit*[1] library for the latter. Initially, the robotic fish is actuated using different control angles: $\phi_c = 20°, 40°, 50°$ and at various frequencies in the interval $f = [0.2, 1.2]$. The procedure is divided into following steps:

1) The proportionality factor (between α and ϕ) $\lambda = 0.46$ is determined in quasi-static experiments i.e. for experiments conducted at low frequency square wave forcing.
2) From [32], the mean thrust force is defined as $\overline{F_x} = -\frac{K}{T}\int_0^T \alpha(t)\ddot{\alpha}(t)dt = \frac{K}{T}\int_0^T \dot{\alpha}(t)^2 dt$. It is proportional to $\omega^2\alpha_{\max}^{\star\,2}$. We employ ridge linear regression to fit the coefficient C_t in eq. (5), setting $\overline{F_x}$ as a function of $\omega^2\alpha_{\max}^{\star\,2}$ [see fig. 2b].
3) The corresponding maximum fin deflection angle $\alpha_{\max}^{\star}$ is measured from the videos recorded of fin oscillating at these different angle amplitudes and frequencies [see fig. 3]. This maximum angle $\alpha_{\max}^{\star}$ is then used in the next step.
4) With a pre-defined range for physically plausible values of different parameters, we proceed with nonlinear fit of the parameters Ω, Δ of eq. (2) and ξ, ω_0 of eq. (4) on the experimental data. The result of the fit can be visualized in fig. 2b.
5) We fit C_d from the second order polynomial fit.

[1]https://lmfit.github.io/lmfit-py/

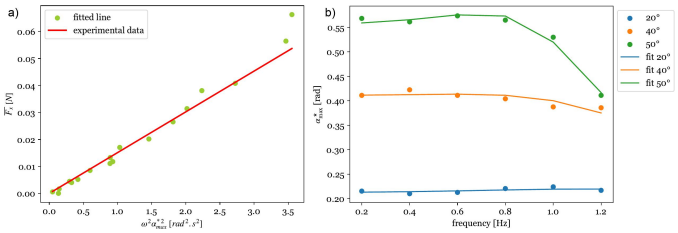

Fig. 2. Parameters fitting results comparison with actual data. a) Fit of parameter $C_t = 12.9\ 10^{-3}$ N.rad^{-2}.s^2 in equation eq. (4) via linear regression b) Superposition of measured angle of the fish fin undulation and prediction by the model. The data from three servo-motor actuation angles ($\phi_c = 20°, 40°, 50°$) and 6 different frequencies ($f \in [0.2, 1.2]$) were used to find the parameters $\Omega = 5.8$ rad.s^{-1}, $\Delta = 0.29$ rad, $\omega_0 = 12.5$ rad.s^{-1} and $\xi = 1.2$.

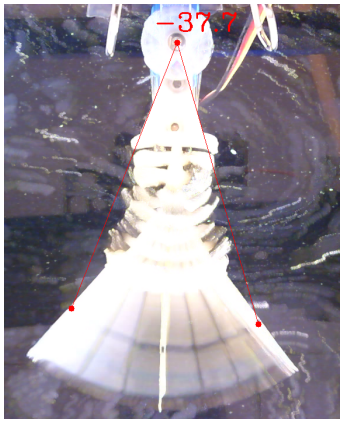

Fig. 3. DRAFTMean image obtained from averaging of images from video recording of several tail undulations. Example of determining two times $\alpha_{\max}^{\star}$ from mean image. The angle is measured from the static marker near the end of caudal fin.

Simulation results: After constructing the simulator with the above dynamics, we add a simple visualization of movement that depends on both linear movement x and fin deflection angle α [see fig. 4a and b]. Here, we distinguish between two distinct observation modalities for learning. The first category includes the actual true sensor state, denoted by $[x, \dot{x}, \alpha, \dot{\alpha}]$, while the second one consists of the processed image of the simulator [see fig. 4a]. The state of the system is updated using the backward Euler integration scheme with a sampling time step of 20 ms.

All the following learning simulation results are based on the continuous action space $\phi_c \in [-60°, 60°]$, with all results being shown in dimensionless units.

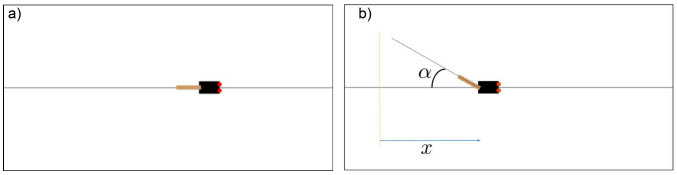

Fig. 4. Visualization from the simulator with size (400, 800). The thin brown rectangle represents the tail, the black rectangle with two red circles represents the fish body. For visual control, this input is sampled every sampling time and is then rescaled to (84, 84). a) raw image b) annotated for visualization purposes of x and α.

Before applying deep RL in simulation, we enforced three types of classical periodic control functions (sinusoidal, triangular and square wave) for a given Φ and measured the average speed $\dot{x}$ as a function of frequency f during one episode. Results are shown in fig. 5. Square wave forcing consistently outperforms other functions for speed maximization. We then perform deep RL learning using either images or true state for observations. Both modalities were suitable for learning and inference results are presented in fig. 6.

In our simulations, we applied PPO [35] on learning with from true state, while we used DrQv2 [48] for learning from images. DrQv2 is a SOTA model-free deep RL algorithm to learn from images incorporating data augmentation and enhanced actor-critic update scheme based on [25].

Learning from true state converges rapidly to the optimal policy [see fig. 6a]. The resulting swimming gait in a square wave control also known as "bang-bang" control which arises as an optimal control in many mechatronic systems [19]. We can also confirm that PPO was able to find the optimal frequency, while DrQv2 a near optimal one. We now proceed with another simulation based on a popular physics engine, mujoco [42].

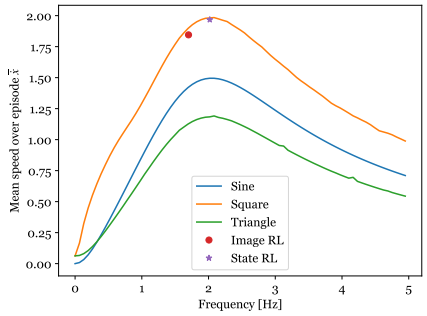

Fig. 5. Influence of frequency on episodic return in a custom fish swimming environment. The enforced swimming gaits consist of predefined functions: square wave (orange), sinusoidal (blue) and triangular (green) functions with frequency range $f \in [0.1, 5]$ Hz. The red circle is the learning results from DrQv2 on the visual data (PPO could not learn from visual data in a reasonable time). The star signifies the best inference return from the PPO training on the true state $[x, \dot{x}, \alpha, \dot{\alpha}]$.

Physics-engine based simulation: We adapted a fish environment from [40] based on a mujoco engine to be constrained in 1D. The modified version can be visualized in fig. 7.

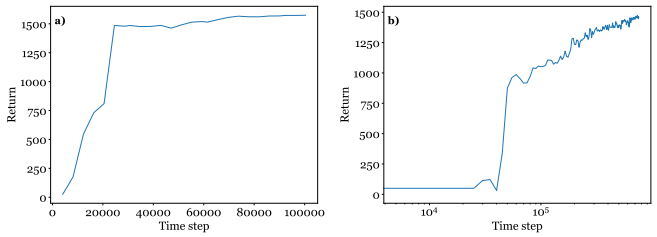

Fig. 6. Evaluation curve of learning via a) PPO algorithm trained on true state $[x, \dot{x}, \alpha, \dot{\alpha}]$ b) DrQv2 algorithm trained on processed images of a simulator. The curve represents the algorithm's evaluation score every 5000 time steps.

This fish is a rigid body with a two-link tail. One part of the tail (small oval) is actuated and another is compliant. This closely approximates the dynamics of a robot fish in fig. 5, when the motor actuates the upper part of the tail and another one remaining compliant. As before, to get the first insight onto the control of this fish, we apply three classical functions for actuation and record mean forward speed over a specified duration. These velocities are plotted as a function of frequency in fig. 8, revealing a consistent trend with earlier simulations based on custom dynamics. Namely, the square wave control outperforms other functions and a peak frequency in forcing exist which can be rationalized due to the fin elasticity (compliant part).

Partial conclusion: We have got useful insights into expected behavior of the system and we now proceed with experiments. We found the "bang-bang" control optimality with the peak forcing frequency rationalized by the body elasticity. The hyperparameters used in experimental part were tuned through hyperparameter search in simulation. Finally, simulations with true state of the system tend to be faster, with a slightly better performance.

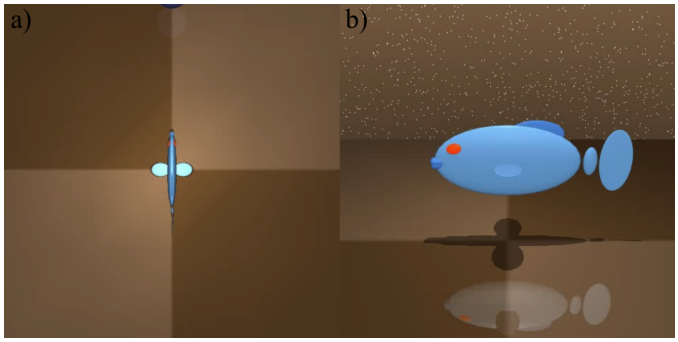

Fig. 7. Example image of physics-based fish environment using mujoco engine [42]: a) top view (used for RL training) b) side-view.

B. Experimental results: speed maximization

We now impose predefined policies (sinusoidal, triangular and square wave) on the robot in fig. 5b and plot mean speed $\dot{x}$ during 1 episode as a function of frequency f. The result is represented in fig. 9 for $\Phi = 30°, 20°$. In the conducted experiments, sinusoidal and triangular waveforms are inferior

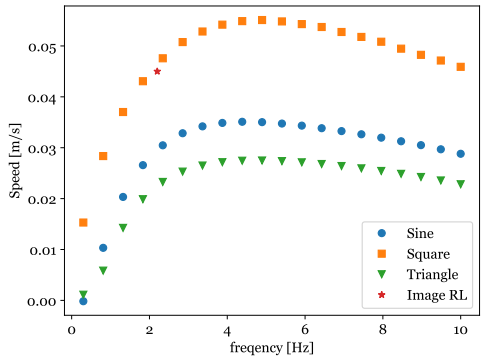

Fig. 8. Simulation results of the fish 1D swimming environment using mujoco engine. It can be deduced that the square wave control is the most effective, surpassing both the sinusoidal and the triangular forcings. DrQ-v2 was able to learn the square wave forcing of the fish from visual data, but failed to optimize the frequency, which is rationalized by the low visibility of the fin from the top view.

to square wave control for most of the cases. Yet, for $\Phi = 30°$ the maximum speed is achieved independent of the signal form due to the hardware limitations of servomotor to follow large Φ at high frequencies.

Learning with continuous action space: We use DROQ actor-critic algorithm to optimize the swimming speed with 20 UTD ratio for critic and 0.05 dropout rate. The action space consists of $\phi_c \in [-\Phi, \Phi]$. The learning curve increases rapidly and the agent was able to rapidly learn how to oscillate the fin harmonically and swim in a straight line. The learning curve and inference evaluation are represented in fig. 10. We performed the RL training for different $\Phi = \{20°, 30°\}$ that are compared to predefined control policies in fig. 9. The control frequency was found through Fourier frequency analysis.

Discrete action space: We employ PPO [35] actor-critic algorithm with $\phi_c \in \{-\Phi, 0, \Phi\}$, with $\Phi = 40°$. Because actions are discrete, only triangular or square wave forcing are possible. The inference results are shown in fig. 11, where PPO was successful to find square wave policy. To validate RL findings, we compare RL inference frequency with square wave forcing depicted in fig. 12. RL is able to find near optimal frequency that is limited by the coarse granularity of the sampling interval.

We also compare the found optimal control with thrust force maximization following the protocol described in [16]. When we conducted forward thrust optimization experiments using the same robotic fish in a static condition with the bi-directional force sensor, we observed that the optimal frequencies around 2 Hz. While maximizing the thrust force is equivalent to maximizing the speed from the physical perspective, this simplified perspective neglects a nuanced physical phenomena that can come into play in experiments. Primarily, the undulatory motion of a fish's tail generates waves that propagate through the water, altering flow conditions and potentially influencing the fish's movement [24]. The wave

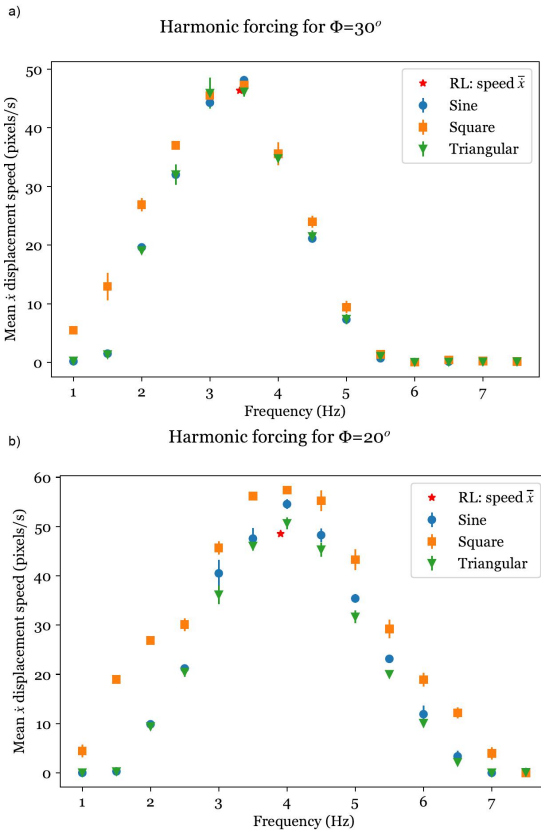

Fig. 9. Influence of the frequency on the speed $\dot{x}$ by some periodic control functions: square (orange squares), sinusoidal (blue circles) and triangular (green triangles) waves. We plot mean and standard deviation on three test episodes.

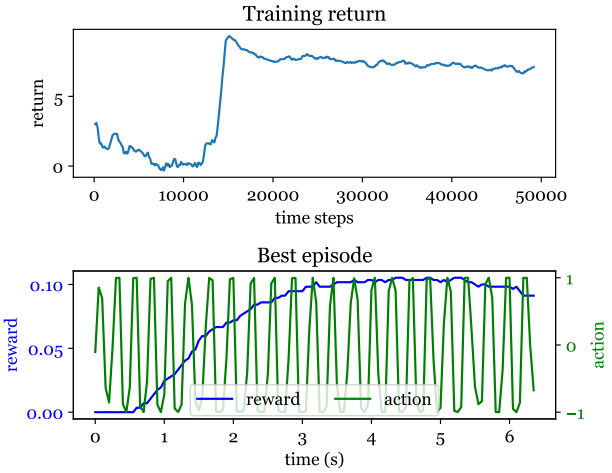

Fig. 10. Data for $\Phi = 30°$ a) Learning curve. b) Highest return actions and rewards.

propagation (referred as sloshing) in the water tunnel is highly dependent on the geometry and the size of a fish and the tank. While this physical phenomena is well present in practice, it is difficult to model it in simulation.

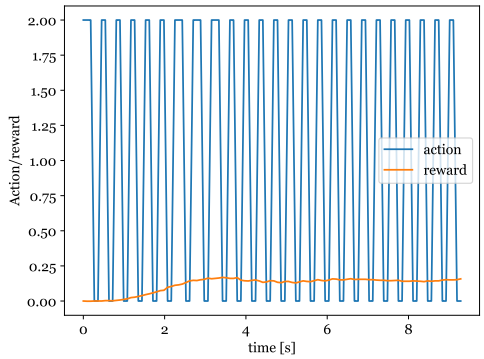

Fig. 11. Experimental results of PPO inference for a robotic fish swimming in a water tunnel after 120000 time steps with sampling time of 50 ms. The base frequency is 4 sampling times, which translates to the actuation frequency of f=2.74 Hz.

Discussions

We have successfully demonstrated that the mean speed of a robotic fish can be optimized using the visual data and predetermined markers on a fish body. The main purpose of putting the markers is to indicate the useful features from high-dimensional images without training a feature extractor. Having already processed information from images should boost the training speed. To add the information about fish fin undulation, we fuse the command ϕ_c with the features extracted from the image to compose the observation. We apply three preset control policies to propulse the robotic fish and clearly observe the optimal forcing frequency.

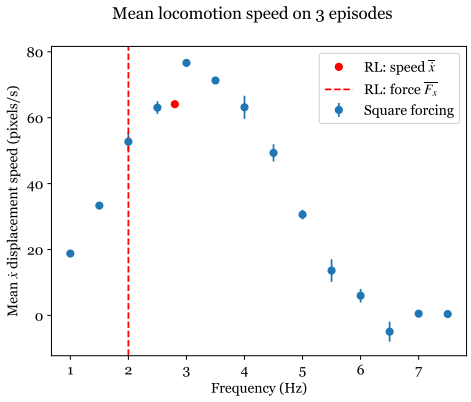

Fig. 12. done with other camera in december, I will normalize to match current results. Comparison of different approaches for finding an optimal swimming gait. Blue symbols corresponds to forcing with square wave policy, the discontinuous red line to the optimal frequency found of thrust optimization and red symbol to the speed maximization via RL.

V. CONCLUSION

In this article, reinforcement learning approach of fish swimming gait optimization is presented. We discuss different methods and their applicability both in simulation and experiments. The proposed simulation model closely approximates the dynamics of a real robot fish movements from physical perspective. This model served as an algorithm validation and hyperparameter optimization tool before proceeding with experiments. Fish locomotion and robot fish control are challenging topics and we believe that this work sheds new insights on optimal swimming gaits for designing more efficient robotic fish movements.

ACKNOWLEDGMENT

REFERENCES

[1] Richard Bainbridge. The speed of swimming of fish as related to size and to the frequency and amplitude of the tail beat. Journal of experimental biology, 35(1):109–133, 1958.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. arXiv preprint arXiv:1606.01540, 2016.

[3] Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. arXiv preprint arXiv:2101.05982, 2021.

[4] Zhiwei Chen, Jiahui Wang, Xin Wang, Yufei Zhao, Xu Li, and Yang Liu. Design and control of soft biomimetic pangasius fish robot using fin ray effect and reinforcement learning. Scientific Reports, 12(1):21861, 2022.

[5] Stephen Childress. Mechanics ofswimming and flying. Number 2. Cambridge University Press, 1981.

[6] Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare, and Aaron Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In Deep Reinforcement Learning Workshop NeurIPS 2022, 2022.

[7] Mattia Gazzola, Médéric Argentina, and Lakshminarayanan Mahadevan. Scaling macroscopic aquatic locomotion. Nature Physics, 10(10):758–761, 2014.

[8] Florence Gibouin, Christophe Raufaste, Yann Bouret, and Médéric Argentina. Study of the thrust–drag balance with a swimming robotic fish. Physics of Fluids, 30(9):091901, 2018.

[9] Abhishek Gupta, Justin Yu, Tony Z Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 6664–6671. IEEE, 2021.

[10] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In 35th International Conference on Machine Learning, ICML 2018, 2018.

[11] Kevin Healy, Thomas HG Ezard, Owen R Jones, Roberto Salguero-Gómez, and Yvonne M Buckley. Animal life history is shaped by the pace of life and the distribution of age-specific mortality and reproduction. Nature ecology & evolution, 3(8):1217–1224, 2019.

[12] Archibald Vivian Hill. The heat of shortening and the dynamic constants of muscle. Proceedings of the Royal Society of London. Series B-Biological Sciences, 126(843):136–195, 1938.

[13] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. arXiv preprint arXiv:2110.02034, 2021.

[14] Yonghui Hu, Wei Zhao, Long Wang, and Yingmin Jia. Underwater target following with a vision-based autonomous robotic fish, 2009.

[15] Zheyuan Hu, Aaron Rovinsky, Jianlan Luo, Vikash Kumar, Abhishek Gupta, and Sergey Levine. Reboot: Reuse data for bootstrapping efficient real-world dexterous manipulation. arXiv preprint arXiv:2309.03322, 2023.

[16] S Israilov, J Sanchez Rodriguez, C Brouzet, G Allibert, C Raufaste, M Argentina, et al. Optimum control strategies for maximum thrust production in underwater undulatory swimming. arXiv preprint arXiv:2309.14025, 2023.

[17] Yusheng Jiao, Feng Ling, Sina Heydari, Nicolas Heess, Josh Merel, and Eva Kanso. Learning to swim in potential flow. Physical Review Fluids, 6(5):050505, 2021.

[18] Robert K Katzschmann, Joseph DelPreto, Robert MacCurdy, and Daniela Rus. Exploration of underwater life with an acoustically controlled soft robotic fish. Sci. Robot., 3(16):1–13, 2018.

[19] Donald E Kirk. Optimal control theory: an introduction. Courier Corporation, 2004.

[20] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In International Conference on Machine Learning, pages 5639–5650. PMLR, 2020.

[21] Vincent Lebastard, Frédéric Boyer, and Sylvain Lanneau. Reactive underwater object inspection based on artificial electric sense. Bioinspir. Biomim., 11(4):045003, 2016.

[22] Keel Yong Lee, Sung-Jin Park, David G. Matthews, Sean L. Kim, Carlos Antonio Marquez, John F. Zimmerman, Herdeline Ann M. Ardoña, Andre G. Kleber, George V. Lauder, and Kevin Kit Parker. An autonomously swimming biohybrid fish designed with human cardiac biophysics. Science, 375(6581):639–647, 2022.

[23] Qiyang Li, Aviral Kumar, Ilya Kostrikov, and Sergey Levine. Efficient deep reinforcement learning requires regulating overfitting. arXiv preprint arXiv:2304.10466, 2023.

[24] James C Liao. A review of fish swimming mechanics and behaviour in altered flows. Philosophical Transactions of the Royal Society B: Biological Sciences, 362(1487):1973–1993, 2007.

[25] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In 4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc., 2016.

[26] Wenji Liu, Kai Bai, Xuming He, Shuran Song, Changxi Zheng, and Xiaopei Liu. Fishgym: A high-performance physics-based simulation framework for underwater robot learning. In 2022 International Conference on Robotics and Automation (ICRA), pages 6268–6275, 2022.

[27] A. Mathis, S. Schneider, J. Lauer, and M. W. Mathis. A primer on motion capture with deep learning: Principles, pitfalls, and perspectives. Neuron, 108:44–65, 2020.

[28] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. Neural networks, 21(4):682–697, 2008.

[29] Giovanni Polverino, Paul Phamduy, and Maurizio Porfiri. Fish and Robots Swimming Together in a Water Tunnel: Robot Color and Tail-Beat Frequency Influence Fish Behavior. PLoS One, 8(10):47–50, 2013.

[30] Sunil Kumar Rajendran and Feitian Zhang. Design, Modeling, and Visual Learning-Based Control of Soft Robotic Fish Driven by Super-Coiled Polymers. Frontiers in Robotics and AI, 8, 2022.

[31] M Saadat, Frank E Fish, AG Domel, V Di Santo, GV Lauder, and H Haj-Hariri. On the rules for aquatic locomotion. Physical Review Fluids, 2(8):083102, 2017.

[32] J. Sánchez-Rodríguez, F. Celestini, C. Raufaste, and M. Argentina. Proprioceptive mechanism for bioinspired fish swimming. Phys. Rev. Lett., 126:234501, Jun 2021.

[33] Jesús Sánchez-Rodríguez, Christophe Raufaste, and Médéric Argentina. A minimal model of self propelled locomotion. Journal of Fluids and Structures, 97:103071, 2020.

[34] Jesús Sánchez-Rodríguez, Christophe Raufaste, and Médéric Argentina. Scaling the tail beat frequency and swimming speed in underwater undulatory swimming. Nature Communications, 14(1):5569, 2023.

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. arXiv:1707.06347, 2017.

[36] Laura Smith, Yunhao Cao, and Sergey Levine. Grow your limits: Continuous improvement with real-world rl for robotic locomotion. arXiv preprint arXiv:2310.17634, 2023.

[37] Laura Smith, Ilya Kostrikov, and Sergey Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. arXiv preprint arXiv:2208.07860, 2022.

[38] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958, 2014.

[39] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.

[40] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. Deepmind control suite. CoRR, abs/1801.00690, 2018.

[41] Robin Thandiackal, Kamilo Melo, Laura Paez, Johann Herault, Takeshi Kano, Kyoichi Akiyama, Frédéric Boyer, Dimitri Ryczko, Akio Ishiguro, and Auke J. Ijspeert. Emergence of robust self-organized undulatory swimming based on local hydrodynamic force sensing. Science Robotics, 6(57), 2021.

[42] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In 2012 IEEE/RSJ international conference on intelligent robots and systems, pages 5026–5033, 2012.

[43] George S Triantafyllou, Michael S Triantafyllou, and Mark A Grosenbaugh. Optimal thrust development in oscillating foils with application to fish propulsion. Journal of Fluids and Structures, 7(2):205–224, 1993.

[44] Ming Wang, Huifang Dong, Xu Li, Yanlu Zhang, and Junzhi Yu. Control and Optimization of a Bionic Robotic Fish Through a Combination of CPG model and PSO. Neurocomputing, 337:144–152, 2019.

[45] Wei Wang, Dongbing Gu, and Guangming Xie. Autonomous optimization of swimming gait in a fish robot with multiple onboard sensors. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 49(5):891–903, 2017.

[46] Wei Wang, Jiajie Guo, Zijian Wang, and Guangming Xie. Neural controller for swimming modes and gait transition on an ostraciiform fish robot. In 2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pages 1564–1569. IEEE, 2013.

[47] Lang Yan, Xinghua Chang, Runyu Tian, Nianhua Wang, Laiping Zhang, and Wei Liu. A numerical simulation method for bionic fish self-propelled swimming under control based on deep reinforcement learning. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 234(17):3397–3415, 2020.

[48] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 10674–10681, 2021.

[49] J. Yu, M. Tan, S. Wang, and E. Chen. Development of a Biomimetic Robotic Fish and Its Control Algorithm. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 34(4):1798–1810, 2004.

[50] Junzhi Yu, Min Tan, Jian Chen, and Jianwei Zhang. A survey on CPG-inspired control models and system implementation. IEEE Trans. Neural Networks Learn. Syst., 25(3):441–456, 2014.

[51] Junzhi Yu, Chen Wang, and Guangming Xie. Coordination of multiple robotic fish with applications to underwater robot competition. IEEE Transactions on Industrial Electronics, 63(2):1280–1288, 2015.

[52] Junzhi Yu, Kai Wang, Min Tan, and Jianwei Zhang. Design and control of an embedded vision guided robotic fish with multiple control surfaces. ScientificWorldJournal, 2014:631296, 2014.

[53] Feitian Zhang, Osama Ennasr, Elena Litchman, and Xiaobo Tan. Autonomous Sampling of Water Columns Using Gliding Robotic Fish: Algorithms and Harmful-Algae-Sampling Experiments. IEEE Systems Journal, 10(3):1271–1281, 2016.

[54] Feitian Zhang, Francis D Lagor, Derrick Yeo, Patrick Washington, and Derek A Paley. Distributed flow sensing for closed-loop speed control of a flexible fish robot. Bioinspiration & Biomimetics, 10(6), 2015.

[55] Tianhao Zhang, Runyu Tian, Chen Wang, and Guangming Xie. Path-following control of fish-like robots: A deep reinforcement learning approach. IFAC-PapersOnLine, 53(2):8163–8168, 2020.

[56] Tianhao Zhang, Runyu Tian, Chen Wang, and Guangming Xie. Path-following control of fish-like robots: A deep reinforcement learning approach. IFAC-PapersOnLine, 53(2):8163–8168, 2020.

[57] Tianhao Zhang, Runyu Tian, Hongqi Yang, Chen Wang, Jinan Sun, Shikun Zhang, and Guangming Xie. From simulation to reality: A learning framework for fish-like robots to perform control tasks. IEEE Transactions on Robotics, 38(6):3861–3878, 2022.

[58] Junzheng Zheng, Tianhao Zhang, Chen Wang, Minglei Xiong, and Guangming Xie. Learning for Attitude Holding of a Robotic Fish: An End-to-End Approach With Sim-to-Real Transfer. IEEE Trans. Robot., PP:1–17, 2021.

[59] J. Zhu, C. White, D. K. Wainwright, V. Di Santo, G. V. Lauder, and H. Bart-Smith. Tuna robotics: A high-frequency experimental platform exploring the performance space of swimming fishes. Science Robotics, 4(34), 2019.