2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

# Simple Models, Real Swimming: Digital Twins for Tendon-Driven Underwater Robots

Mike Y. Michelis[1], Nana Obayashi[2,3], Josie Hughes[3], and Robert K. Katzschmann[1]

***Abstract*—Mimicking the graceful motion of swimming animals remains a core challenge in soft robotics due to the complexity of fluid-structure interaction and the difficulty of controlling soft, biomimetic bodies. Existing modeling approaches are often computationally expensive and impractical for complex control or reinforcement learning needed for realistic motions to emerge in robotic systems. In this work, we present a tendon-driven fish robot modeled in an efficient underwater swimmer environment using a simplified, stateless hydrodynamics formulation implemented in the widespread robotics framework MuJoCo. With just two real-world swimming trajectories, we identify five fluid parameters that allow a matching to experimental behavior and generalize across a range of actuation frequencies. We show that this stateless fluid model can generalize to unseen actuation and outperform classical analytical models such as the elongated body theory. This simulation environment runs faster than real-time and can easily enable downstream learning algorithms such as reinforcement learning for target tracking, reaching a 93% success rate. Due to the simplicity and ease of use of the model and our open-source simulation environment, our results show that even simple, stateless models — when carefully matched to physical data — can serve as effective digital twins for soft underwater robots, opening up new directions for scalable learning and control in aquatic environments.**

## I. Introduction

Animals move gracefully and efficiently in their natural habitats, and can perform a broad variety of tasks that present-day robots are still far away from being capable of, from agile locomotion to dexterous manipulation. These animals show controlled actions through a complex interplay between the rigid and soft materials in their bodies with the multiphysical environment around them.

One such environment that features a very dense and continuous interaction between animal and surrounding is the underwater environment. Unlike robots in locomotion or manipulation, where contact is sparse and discontinuous, underwater settings require a continuous exchange of forces between the robot surface and the surrounding fluid. Through this complex interaction we might find natural and efficient movements emerge in our robot. Several underwater robots have been built to mimic the elegant motions found in nature, such as fish [1], [2], jellyfish [3], and starfish [4]. However, modeling and control has remained a challenging topic for these soft robotic systems.

[1] Soft Robotics Lab and ETH AI Center, ETH Zurich, Switzerland
[2] Prema Lab, New York University, USA
[3] CREATE Lab, EPFL, Switzerland

Videos, code, and data are available on our website at: `srl-ethz.github.io/website_fishsim`

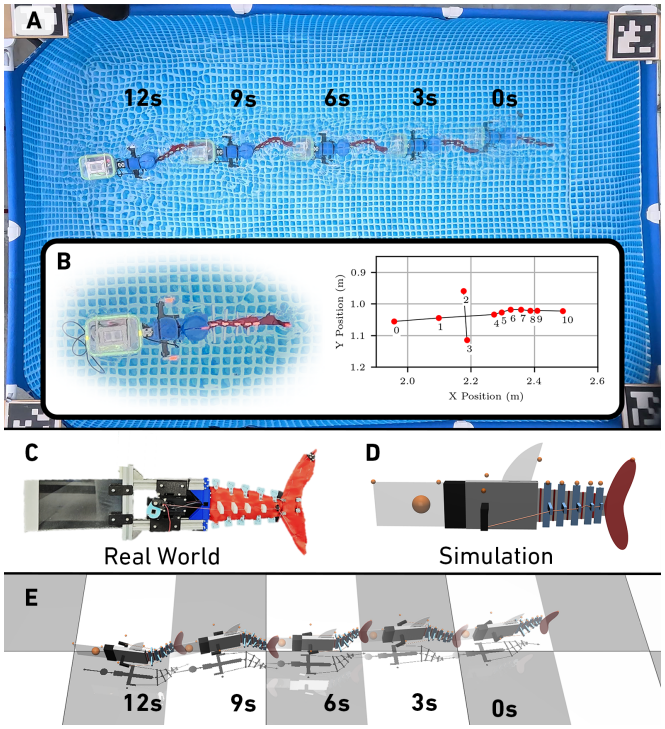


Fig. 1: Overview of the simulated and real swimmer robots. *A)* Robot swimming in the pool captured from a top-down view, images overlaid are 3 s apart. *B)* The 11 markers are tracked and extracted from the video. *C)* Side-view of the hardware with a single motor actuating the tendon-driven tail. *D)* Simulated robot with markers indicated by small orange spheres. Center of mass is located at the large orange sphere in the head. *E)* Simulated trajectory after matching fluid model to real experiment.

Soft robotics in general has often required computationally expensive modeling approaches in the past, which have to be tuned through system identification [5], [6] to match real-world behavior. Simplified modeling has been applied to soft robots that can be approximated as 1D-splines [7] or as rods [8], but this severely limits the range of robot geometries that can be considered. Similarly, black-box neural networks can be used to approximate these complex dynamics [9], [10], but fail to be interpretable or generalize to unseen scenarios.

In this paper we present a computationally efficient approximation to modeling biomimetic tendon-driven swimmers, such that we can further investigate the natural motions that arise from a flexible spine that is actuated to interact with the environment. Previous work [11] has shown a simplified fluid model for analyzing fruit fly behavior. We verify this simplification can match real-world experimental data for

underwater swimmers with as little as 2 trajectories, and can also generalize to a wide range of unseen actuation frequencies. We provide the integration of this robot model into standard Reinforcement Learning (RL) pipelines for the broader robot learning community, and we showcase accurate RL agents for target tracking. In summary, we contribute:

1) Simulation environment for underwater tendon-driven robots: We test a stateless fluid model for underwater swimming.
2) Sim-to-real matching: We show the matching and generalization at different constant frequencies of actuation for this simplified fluid model.
3) Outperforming baseline: Analytical models such as elongated body theory struggle to describe the swimming of our robot, while our model can generalize to unseen actuation.
4) Downstream reinforcement learning: We demonstrate how our digital twin can be used for specific control tasks, such as learning to track targets.

## II. Related Work

Computational Fluid Dynamics (CFD) simulations are computationally expensive, but can provide accurate insights in fish swimming behavior in controlled environments [12]. These accurate simulations can be shown to work in an RL pipeline in 2D [13], where a surrogate neural network is trained on experimental data such that the RL agent first learns in this faster surrogate environment, then transitions to the accurate CFD when the policy is sufficiently precise. However, even though the training in the surrogate took less than an hour, the finetuning in CFD needed 16 days, hence such an approach remains infeasible to scale to 3D due to computational cost. An efficient approach leveraging GPU parallelization was proposed in [14] using Lattice Boltzmann Method for the fluid and a skinned, articulated rigid skeleton for the fish. This was transferred to a surface swimmer in [15] with careful tuning of the servo motors and fluid forces. The approach is presented for swimmers with controllable motors in each joint of the tail, whereas in this paper we explore the application to underactuated systems that rely on compliance, such as tendon-driven swimmers with a flexible spine.

The simplest model for fluid dynamics that is solving for a time-dependent fluid state would be particle-based methods such as the material point method [16], which can provide qualitatively good results, but struggle especially with boundary conditions. Compared to full CFD, the fluid-structure interaction between robot and fluid has shown to generate physically inaccurate fluid forces for the material point method [17], rendering it a poor choice for underwater robotics.

Simplifications can be made to these expensive CFD methods, such as added mass or slender body theorem [18], which consider the local fluid behavior around the submerged object, and lead to stateless thrust/drag formulations with only a few parameters to match to fish swimming experiments [4], [19]–[21]. These methods are stateless since no fluid pressure or velocity field is solved for; the fluid force on the fish is directly computed from the state of the fish and not the state of the fluid. This significantly speeds up the simulation time, but at the cost of not being able to propagate fluid state information over time, ignoring, for example, how fluids interact with wall boundaries or how multiple objects in the same fluid domain interact with one another.

To simulate the flying behavior of a fruit fly, a more expressive fluid model was developed including blunt drag, slender drag, angular drag, Kutta lift, and Magnus lift [11]. These stateless fluidic forces and torques are based on a quasi-steady-state approximation [22] with the added simplification that all bodies experiencing fluid forces are ellipsoids. We will expand this work to underwater robots to show its feasibility in matching real-world swimmer dynamics.

Given enough experimental data, learning-based approaches can be applied to fish swimming as well. A black-box neural network was trained to predict fluid thrust forces of a fixed, pneumatically-actuated sillicone fish tail [9], but was shown to struggle while generalizing to the stiffer sillicone tails. Another approach is to learn the fluid dynamics using a physics-informed surrogate model, and combine this into a differentiable simulation pipeline for optimizing the control frequency [23]. Providing more structure in the learning architecture, [24] present a data-driven reduced-order model for direct numerical simulations of an undulating swimmer and [10] use a Koopman operator to predict the time-dependent dynamics of a real-world motor-driven fish while accounting for the background flow using a very coarse 2D CFD to reduce computational cost. These applications remain rather task-specific and data-hungry, hence in this paper we propose a more general learning framework that closely follows the learning-based robotics community using simulators such as MuJoCo [25] for bridging the sim-to-real gap for soft underwater robotics. We demonstrate that our approach can achieve comparable fidelity to learned models, while remaining highly interpretable and efficient, and requiring far less data to calibrate a model matching to reality.

## III. Experimental Setup

### A. Hardware Setup

The robotic fish design is based on a parametrically length-scalable framework, developed through work [26], which allows the robot to be fabricated at different lengths. The robot's physical frame is split into two distinct sections: the passive front-end and the active, compliant tail. The passive front-end is constructed with acrylic sheets fixed with 3D-printed pieces to aluminum extrusions. Acrylic sheets are also used to construct the pectoral and dorsal fins for stability. A single dual-shaft, waterproof Dynamixel XW-540-T140-R motor is centrally mounted within this front-end. This motor actuates two fishing wire tendons through a crank-slider-inspired mechanism. These tendons run along either side of the tail, with their routing transitioning at the tail's midpoint to create an antagonistic S-bend motion, mimicking aquatic creature kinematics.

The active tail is constructed from four straight $1\,\mathrm{mm}$ diameter nitinol rods that run its entire length. These rods provide the structural stiffness and elasticity for propulsion and are rigidly secured to 3D-printed PLA structural supports placed at five discrete, equidistant points along the tail. These supports have guide holes for the actuating tendons, which terminate at the caudal fin. The caudal fin itself is formed by two additional nitinol rods, mounted vertically perpendicular to the tail body with 3D-printed supports. All non-waterproof sensing, computing, and power electronics are housed within a custom waterproof container atop the front-end frame. A 3S LiPo battery powers a DC-DC converter for a Raspberry Pi Zero, and a Dynamixel U2D2 Motor Controller. A waterproof resistant fabric covers the tail and part of the body to provide surface area for propulsion, and polyethylene foam is added to make the swimmer buoyant.

The tail of the fish is divided in five spine segments (the PLA structural supports), each of which has a marker placed on top which can be tracked along with the markers on the rigid body, head, and tail fin using a CSRT tracker [27] from OpenCV. The bulk size of the robot is roughly $57\,\mathrm{cm}$ long, $25\,\mathrm{cm}$ tall, and $17\,\mathrm{cm}$ wide. We place the swimmer in a $2\,\mathrm{m} \times 3\,\mathrm{m}$ pool, capture the camera footage from the top down using a GoPro Hero 8, at a resolution of $2160 \times 3840$, 60 frames per second, and a linear field of view. This video data can be seen in Figure 1, where the extracted markers are shown as well. The tracked markers are processed by rotating all markers into their local frame where the heading direction of the fish defines the negative x-axis.

### B. Simulation Setup

We make several simplifications to efficiently simulate the robotic swimmer, these can be seen in Figure 2. Firstly, we assume the continuously bending tail is discretized in rigid segments connected by hinge joints, which has been shown to be a valid approximation for continuum robots [28].

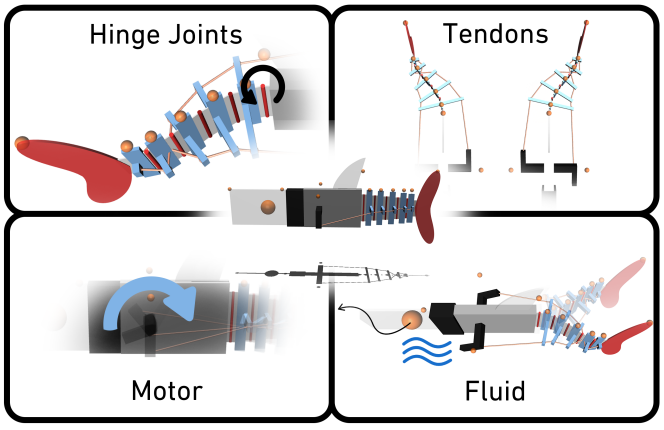


Fig. 2: Overview of the main mechanisms for the swimmer simulation environment. We simplify the deformable spine with multiple hinge joints as an articulated rigid body, we use stiff elastic tendons for the tendon-driven actuation, a velocity-controlled motor that pulls the tendons, and a simplified fluid model to mimic real-world swimming behavior.

And secondly, a simplified, stateless fluid dynamics model [11] is used to model thrust generation of the fish. This was previously applied on a fruit fly, where we extend its application area to real-time underwater robots as well. Since this fluid dynamics model is implemented in Mujoco [25], we implement our robotic fish in this simulation framework as well.

For the articulated rigid body, we choose to discretize based on the spine segments of the hardware setup, as seen in Figure 1. The stiffness of the hinge joints are identified based on experimental data, as explained in Section IV, and we use the same stiffness for all tail joints. We model the tendons as stiff springs instead of length constraints, since this results in more computationally stable simulations. From our experiments the length of the tendons stretches roughly $3\,\%$ during actuation. Since the buoyant foam on the hardware robot results in a surface swimmer, we mimic this through a soft position constraint in z-direction on the simulated swimmer.

Initially developed for modeling the fluttering of playing cards falling in quiescent and incompressible air, [11] extended this to self-propelling flight of a fruit fly. In this paper, we apply this model to hydrodynamics of a swimmer. The same assumption is made that all bodies that experience the fluid forces are approximated as ellipsoids. The fluid force is defined with the following components:

*Blunt and Slender Drag:*

$$\vec{f}_D = -\rho \left( c_{blunt}\ A_v + c_{slender}\ \left(A_{max} - A_v\right)\right) \|\vec{v}\|\ \vec{v}$$

*Angular Drag:*

$$\vec{\tau}_D = -\rho \left( \left( c_{angular}\ \vec{I}_D + c_{slender}\ \left( \vec{I}_{max} - \vec{I}_D \right) \right) \cdot \vec{\omega} \right) \vec{\omega}$$

*Viscous Drag:*

$$\vec{f}_V = -6\pi\mu\tilde{r}\ \vec{v}$$

$$\vec{\tau}_V = -8\pi\mu\tilde{r}^3\ \vec{\omega}$$

*Kutta Lift:*

$$\vec{f}_K = \rho \frac{c_{kutta}\ A_v}{\|\vec{v}\|} \left( \vec{v} \times \vec{v}_{\|} \right) \times \vec{v}$$

*Magnus Lift:*

$$\vec{f}_M = \rho\ c_{magnus} V\ \vec{\omega} \times \vec{v}$$

*Added Mass:*

$$\vec{f}_A = -\mathbf{M}_A \dot{\vec{v}} + (\mathbf{M}_A \vec{v}) \times \vec{\omega}$$

$$\vec{\tau}_A = -\mathbf{I}_A \dot{\vec{\omega}} + (\mathbf{M}_A \vec{v}) \times \vec{v} + (\mathbf{I}_A \vec{\omega}) \times \vec{\omega}$$

where $\rho$ is the fluid density, $\vec{v}, \vec{\omega}$ the linear and angular velocity of the body respectively, $A_v$ the projected surface of the ellipsoid normal to the flow, $A_{max}$ the maximum projected surface area, $\vec{I}_D$ the diagonal entries of the moment of inertia of the ellipsoid, $\vec{I}_{max}$ a vector with each component equal to the maximum entry in the moment of inertia, $\tilde{r}$ is the average of the ellipsoid radii, $\mu$ the dynamic viscosity, $\vec{v}_{\|}$ the velocity parallel to the body, $V$ the body volume, $\mathbf{M}_A$ and $\mathbf{I}_A$ the diagonal virtual added-mass and virtual-added moment of inertia. Here $[c_{blunt}, c_{slender}, c_{angular}, c_{kutta}, c_{magnus}]$ are the five fluid coefficients we identify. We ran the simulation

on our computer with an Intel i9-11900K CPU, where the fish runs at 14.58x realtime, *i.e.*, we can run $14.58\,\mathrm{s}$ of fish motions after $1\,\mathrm{s}$ of computation time.

As a baseline we compare with a classical, also stateless, formulation from Lighthill's Elongated Body Theory (EBT) model [18]. This model for thrust and drag on a swimmer assumes that the lateral motion is small compared to the body length, which is valid for most undulating swimmers, including ours.

$$T = m \left[ \overline{\left(\frac{\partial h(t,x)}{\partial t}\right)^2} - U^2 \overline{\left(\frac{\partial h(t,x)}{\partial x}\right)^2} \right]_{x=L} \qquad (1)$$
$$D = \frac{1}{2}\rho\, c_D\, S\, U^2$$

where $\overline{h(t,x)}$ implies the average lateral displacement over time, $U$ the cruising speed of the swimmer, $S$ is the total submerged surface area, $c_D$ is the drag coefficient, and the thrust expression is evaluated at the tip of the tail $x = L$. We consider the same local added mass per unit length from [29] for $m = \frac{1}{4}\rho\pi\beta d^2$ evaluated at the tail tip with vertical submerged depth $d$ and a non-dimensional parameter $\beta$ that should be close to 1. During constant cruising, which we consider for our data, the thrust and drag forces balance eachother, which results in a mapping from the tail oscillations $h(t,x)$ to the cruising speed $U$. We compute the spatial derivative of the lateral displacements $h(t,x)$, evaluated at the tip of the tail, using finite difference between the last two tail markers. The results can be found in Section IV-D.

## IV. System Identification

### A. Tail Stiffness

Since we have discretized the continuous tail as articulated rigid bodies, we identify the stiffness of the hinge joints based on the natural frequency of the system. We bend both the simulated and real system to a certain angle and release the tail to record a natural oscillation of the markers we track. We remove the tendons on both systems before conducting this experiment. Performing a Fast Fourier Transform (FFT) on both oscillations reveals a dominant frequency which we match by running a simple search over the scalar stiffness variable, which we assume to be the same value for all hinge joints in the discretized tail. The oscillations and frequencies can be seen in Figure 3. The dominant natural frequency we find is $3.5\,\mathrm{Hz}$.

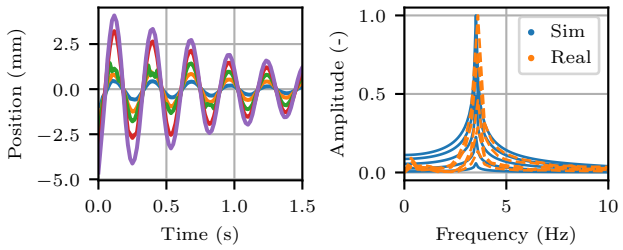


Fig. 3: Natural frequency in both time and frequency domain of the fish tail measured on the real robot tail (left) and of the simulated system after matching the main amplitude frequency with reality (right). The different colors in the left plot indicate the 5 spine segments that are tracked during the oscillation.

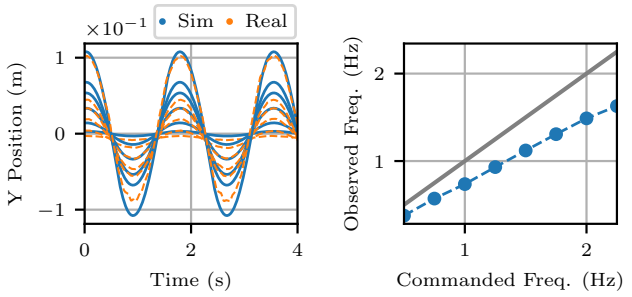


Fig. 4: System identification of motor angular frequency and phase offset to minimize the distance between simulated and real markers, with a minimal distance of $0.026\,\mathrm{m}$ for the $0.60\,\mathrm{Hz}$ swimming shown on the left.

### B. Tail Actuation

Since we have made several simplifications to the robot geometry, we perform another system identification to match the tail motion. We fix the head of the fish and track the markers of the spine segments outside of water while the motor is actuated at a constant angular velocity. By adjusting the motor arm length of the crank $c_m$ in simulation we try to match the actuated tail oscillations as best as possible. We observed that the real motor angular velocity is slightly slower than the commanded velocity, especially at the higher frequencies (rotations per second) likely due to reaching its torque limit, hence we additionally allow the simulated motor angular velocity $\omega$ and the phase offset $\phi$ to be optimized to match the real experiment. The phase offset is only needed because the sinusoidal movement of the tail, i.e. the motor position, is not guaranteed to start at the resting state in the experimental data. Since this is still a low dimensional search space we can sample the entire space to find an optimum that minimizes the mean distance of all markers between simulation and reality for a motion of a fixed $4\,\mathrm{s}$. The minimization problem is defined as:

$$\min_{c_m,\omega,\phi} \frac{1}{T \cdot N} \sum_{t=1}^{T} \sum_{i=1}^{N} \left\| \mathbf{s}_t^i - \bar{\mathbf{s}}_t^i \right\|_2 \qquad (2)$$

where $\bar{\mathbf{s}}_t^i$ is the ground truth marker location $i$ at timestep $t$. We use a total of $N = 11$ markers, of which 5 belong to the rigid head and the remaining 6 are the tail spine segments and the tail fin itself. The marker data, both simulated and real, is processed by rotating all markers into the local frame of the fish head, where the first marker on the head is defined as the origin and the vector from the first to the second marker is the positive x-axis. Each timestep we translate the local frame by the head displacement along the head vector of that timestep. The search space and the final result of the optimization is shown in Figure 4, where the markers for the head are not included in the visualization.

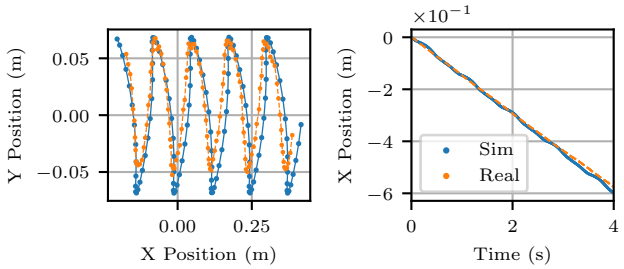


Fig. 5: Optimized results of sim-to-real in-water experiment for 1.19 Hz data. We show the tail fin marker position (left) and forward swimming position (right) to validate the thrust generated by the hydrodynamics model. The average distance error on all markers is computed to be 0.016 m.

We perform this out-of-water experiment on two angular velocities, one at a lower frequency of 0.60 Hz and another at a higher frequency of 1.19 Hz, of which the former is shown in Figure 4. The optimized distance errors for these two frequencies are 26.0 mm and 26.5 mm respectively. These optimized non-zero errors are mainly due to tracking inaccuracy, such as drifting markers, or even fabrication errors, where not all markers can be matched perfectly. This experiment is used to inform us on the motor arm length, which we choose to be 0.0395 m as an average of the two optimization results, close to the measured real motor arm at 0.04 m. For any following experiment, we always optimize the angular velocity of the motor in the real data to find the "real" frequency to instruct our simulated motor.

### C. Fluid Coefficients

Once we have found the motor arm length and angular velocity of swimming outside of water, we conduct sim-to-real for swimming experiments inside of water, including the fluid model in our simulated environment. Following the fluid model from [11], there are five fluid coefficients to optimize: blunt drag, slender drag, angular drag, Kutta lift, and Magnus lift. Since this search space is nonconvex and higher dimensional, we combine a Bayesian Optimization method [30] for global search with a Nelder-Mead algorithm [31] for more local refinement. Our objective is the same as Equation (2), but now we optimize the fluid coefficients as well as a phase offset of the actuation. We set a reasonable bound for the fluid coefficients of $[0, 10]$.

We collected two sets of swimming trajectories on the hardware with constant tail oscillation frequency, the same 0.60 Hz and 1.19 Hz as before outside of water. The fluid coefficients were optimized using these two trajectories. For both the real and simulated data, we let the swimmer warmup until they reach their cruising state before we try to match simulation and reality, we choose this warmup time to be 1 s. The optimization result for the higher frequency can be seen in Figure 5 where we observe an accurate forward swimming velocity throughout the entire trajectory with a final average distance error of 28.8 mm for 0.60 Hz and 16.1 mm for 1.19 Hz. We find the fluid coefficients $[0.40, 7.79, 2.81, 3.84, 0.27]$. The full simulation pipeline with a single swimmer runs at 15× real-time, and could be sped up more through parallelization. We showcase how a large number of robots can be simulated together in real-time in Figure 7.

### D. Comparison to EBT Baseline

We can similarly optimize the drag coefficient in the EBT model to match our 2 experimental trajectories at 0.60 Hz and 1.19 Hz. We compute $\frac{\partial h}{\partial t}$ and $\frac{\partial h}{\partial x}$ through finite difference from the experimental marker data, and tune the $\beta$ in added mass per unit length and $c_D$ from Equation (1) such that the cruising speed $U$ matches our experimental observation. We use the same Bayesian Optimization as for the fluid coefficients in Section IV-C, with bounds of $\beta \in [0.9, 1.1]$ and $c_D \in [0, 10]$, and found optimal values $\beta = 0.91$ and $c_D = 0.31$.

Note that the inherent nature of the optimization task for the fluid coefficients and EBT are different, since EBT optimizes directly to match cruising speed velocity, which is the target metric we care about, whereas our digital twin simulates the whole motion to match the real swimmer movement and extracts the forward velocity from this trajectory. Hence EBT can be seen more as a regression model on holistic descriptions of the swimmer, while the fluid coefficients describe the physics of the problem on a more detailed level of motion. This makes EBT less suitable for complex motions with controllable motor inputs, such as those encountered in reinforcement learning.

## V. Results

### A. Sim-to-Real for Forward Swimming

With the previously identified robot and fluid parameters, we generalize the behavior to different swimming environments. We collected 8 more forward-swimming trajectories at different constant motor angular frequencies, and test the optimized model on its generalizability across swimming frequencies. As shown in Figure 6, the simulated model can accurately capture the trend in forward swimming velocity of the robot, even though it was optimized on only two angular

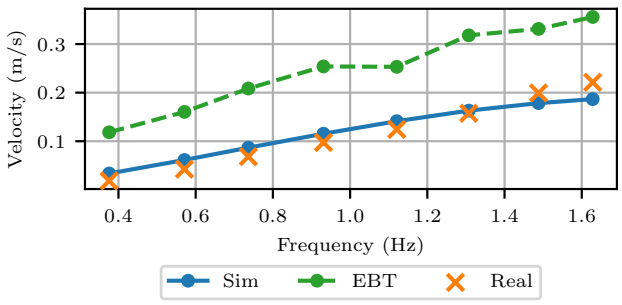


Fig. 6: Sim-to-real generalization across frequencies of our optimized fluid coefficient model, $0.019\,\mathrm{m\,s^{-1}}$ error, and the EBT model, $0.134\,\mathrm{m\,s^{-1}}$. Velocity direction is defined in the local fish frame.

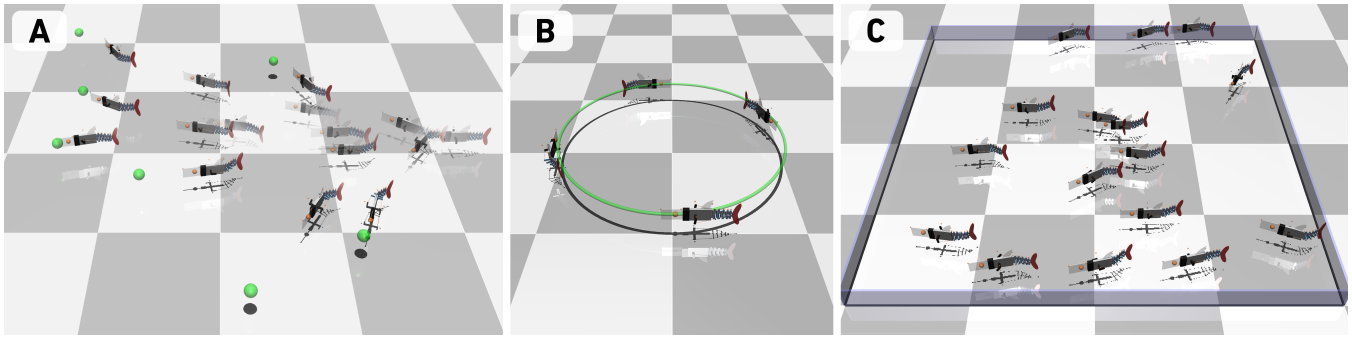


Fig. 7: *A)* SAC agent trained to track random target locations (green) with a 93% success rate. Robot states are overlaid 1 s apart. *B)* SAC agent tracking waypoints along the green circle trajectory with an average target error of 0.009 m. *C)* We demonstrate the ability to simulate a large number of swimmers in real-time.

frequencies of swimming. The behavior at higher frequencies does show higher errors, with an average error in velocity of $0.019\,\mathrm{m\,s^{-1}}$. The physical robot could not reach higher actuation velocities due to the motor used. Since we optimize the observed motor frequency in the video and do not use the commanded actuation frequency, we show in Figure 6 how this error grows at larger angular velocities.

Comparing with the EBT baseline on the experimental frequency sweep data, we observe a significantly larger error for EBT in Figure 6 with an average error of $0.134\,\mathrm{m\,s^{-1}}$. We attribute this larger error to the marker data being potentially different enough that the computation of $\frac{\partial h}{\partial t}$ and $\frac{\partial h}{\partial x}$ for EBT is significantly affected between the previous 2 trajectories and the new 8 trajectories in this task. However, our digital twin seems to be robust to these changes, and can generalize without much loss of accuracy. We verify this hypothesis by optimizing the EBT parameters on our 8 new trajectories, and find a reduced error of $0.027\,\mathrm{m\,s^{-1}}$, which would not be a fair comparison, and is still significantly higher than the error using fluid coefficients.

### *B. Reinforcement Learning for Target Tracking*

Since our matched simulation environment can serve as a digital twin for learning strategies, we showcase this using a Soft-Actor-Critic (SAC) [32] reinforcement learning algorithm trained with the objective to reach target positions. We set up the training environment in Stable Baselines 3 [33], and randomize the target location $\vec{x}_{target}$ in a $4\,\mathrm{m}\times 4\,\mathrm{m}$ square in front of the fish head, an example for a distribution of target points can be seen in Figure 8.

| | **Observations** |
|---|---|
| Motor Joint | $\cos(\alpha)$, $\sin(\alpha)$, $\dot{\alpha}$ |
| Head Frame | $\dot{\theta}_x, \dot{\theta}_y, \dot{\theta}_z$ |
| Tail Joint $i$ | $\phi_i, \dot{\phi}_i$ |
| Target Vector | $\|\vec{d}_t\|$, $\vec{d}_t$, $\dot{\vec{d}}_t$ |
| Prev Action | $u_{old}$ |

TABLE I: The observations used for training the SAC agent for target tracking. All observations are defined relative to the robot frame.

Due to the tendons in the fish spine, velocity control in MuJoCo tended to be more numerically stable than torque control for the motor. Consequently, we chose our control inputs $u(t)$ to be scalar accelerations for the motor such that $v_{new} = v_{old} + c_{action}u(t)dt$, where we can tune the multiplication factor $c_{action}$ and the default action space is $[-1, 1]$. The reward at each simulation step is defined as

$$r(t) = -\|\vec{x}(t) - \vec{x}_{target}\| - \lambda\,|u(t)| + r_{goal} \tag{3}$$

where we found weighing factor $\lambda$ worked best when set to 1, and $r_{goal}$ is defined as $+300$ when the center of mass $\vec{x}(t)$ is within 5 cm of the target location, otherwise 0. The simulation environment is reset when the goal reward is given. We limited the maximal tail flapping frequency to 5 Hz to keep results physically relevant. We defined all observations for the RL agent relative to the current position and orientation of the fish, where our observation space is 22-dimensional, also seen in Table I, consisting of:

- $\cos(\alpha)$, $\sin(\alpha)$, $\dot{\alpha}$: Motor joint position $\alpha$, choosing cos and sin to prevent discontinuities.
- $\dot{\theta}_x, \dot{\theta}_y, \dot{\theta}_z$: Angular velocities of head frame orientation.
- $\phi_i, \dot{\phi}_i$: Tail hinge joint $i$ position and velocity.
- $\|\vec{d}_t\|$, $\vec{d}_t$, $\dot{\vec{d}}_t$: Vector in local frame of the fish head pointing from the center of mass to the target.
- $u_{old}$: Previous action.

We ran a Bayesian hyperparameter optimization on the SAC training parameters, including learning rate, number of

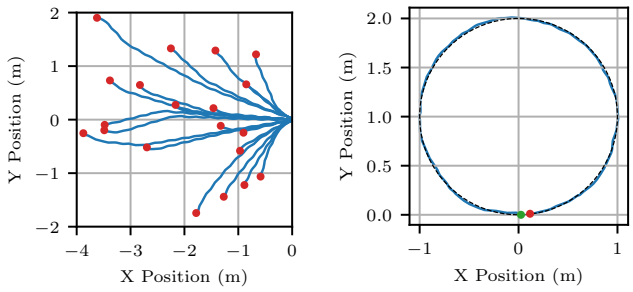


Fig. 8: (Left) Tracking random waypoints in 20 evaluation environments. The red points are the randomized target locations, while the blue curves are the trajectories of the swimmer. (Right) Tracking waypoints along circle with an average distance error of 0.009 m. The green and red points indicate start and end positions respectively.

layers and layer dimension of the policy network, training frequency, when learning starts, soft update coefficient of SAC, and the action multiplier $c_{action}$. From the search we conclude that $c_{action}$ is ideal in the range of $[400, 500]$, and a large number of transitions collected before start of training helps as well (around $500\ 000$). We found the best performing control policy to have $93\%$ success rate on reaching the target, sampled in $100$ evaluation environments, some of which are visualized in Figure 8. The failing targets either tend to be at too high of a curvature to reach, or the target was reached a bit further than the tolerance, hence no environment reset occurs, and the robot keeps swimming and does not know how to turn back. The latter is likely solved with further training, while the former would require more complex controls and trajectory planning, beyond the scope of this paper.

We evaluate our target tracking policy on following waypoints on a $1\,\mathrm{m}$ radius circle. The results in Figure 8 show the tracking behavior, where the waypoint is moving over time along the trajectory. We compute the closest distance to the circular curve at each timestep, and find a mean distance of $0.009\,\mathrm{m}$ with a maximum of $0.023\,\mathrm{m}$. Given that the RL agent is trained to reach within $5\,\mathrm{cm}$ of the target, this tracking performance of an average $0.9\,\mathrm{cm}$ distance has already reached, if not exceeded, the best result it can produce. The results show that the simple, single degree-of-freedom velocity control of tail deflection can result in a wide variety of motions. The balance of simple hardware with complex control demonstrate that asymmetric actuation signals can be learned to cause turning, acceleration, and deceleration.

## VI. Conclusion and Future Work

In this paper we show the applicability of a simplified 5-parameter fluid model for bridging the sim-to-real gap on an underwater swimming fish robot. We show that with as little as 2 trajectories we can identify plausible fluid coefficients that can generalize to a range of angular motor frequencies for real-world swimming experiments, which simpler analytical models such as EBT struggled with. We demonstrate that this efficient digital twin can then be used for downstream tasks such as training RL agents to, for example, track target waypoints with a $93\%$ success rate.

For further tasks such as shape optimization more expressive fluid models would be needed, since we believe stateless fluid descriptions cannot capture the intricacies of robot shape changes. One potential increase in complexity in the fluid model is tuning the fluid coefficients of each segment of the body separately, or use a more advanced extension to EBT such as [34]. Another limitation of the current work lies in the experimental approach of having a fully buoyant surface swimmer, instead of being able to explore a full underwater 3D space. This extension would require fully underwater state estimation and pitch or buoyancy control, which we will look into as a next step. We have considered constant cruising speed conditions for the sim-to-real in this paper, which opens up future challenges in turning and acceleration/deceleration behavior.

Overall, this paper presents an initial step towards computationally efficient methods that can approximate soft robot swimmers in a realistic manner, better than previous analytical solutions. We show how these underwater swimmers can be easily included in standard pipelines for the robot learning community, and hope to reach a wider audience with these simple-to-build and simple-to-deploy biomimetic swimming robots.

## References


[1] R. K. Katzschmann, J. DelPreto, R. MacCurdy, and D. Rus, "Exploration of underwater life with an acoustically controlled soft robotic fish," Science Robotics, vol. 3, no. 16, p. eaar3449, Mar. 2018.

[2] S. M. Youssef, M. Soliman, M. A. Saleh, A. H. Elsayed, and A. G. Radwan, "Design and control of soft biomimetic pangasius fish robot using fin ray effect and reinforcement learning," Scientific Reports, vol. 12, no. 1, p. 21861, Dec. 2022, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41598-022-26179-x

[3] T. Wang, H.-J. Joo, S. Song, W. Hu, C. Keplinger, and M. Sitti, "A versatile jellyfish-like robotic platform for effective underwater propulsion and manipulation," Science Advances, vol. 9, no. 15, p. eadg0292, Apr. 2023. [Online]. Available: https://www.science.org/doi/10.1126/sciadv.adg0292

[4] T. Du, J. Hughes, S. Wah, W. Matusik, and D. Rus, "Underwater Soft Robot Modeling and Control With Differentiable Simulation," IEEE Robotics and Automation Letters, vol. 6, no. 3, pp. 4994–5001, Jul. 2021, conference Name: IEEE Robotics and Automation Letters.

[5] M. Dubied, M. Y. Michelis, A. Spielberg, and R. Katzschmann, "Sim-to-Real for Soft Robots Using Differentiable FEM: Recipes for Meshing, Damping, and Actuation," IEEE Robotics and Automation Letters, vol. 7, pp. 1–1, Apr. 2022.

[6] G. Shi, X. Wang, A. Farinha, J. Pinskier, X. Shen, R. Scalzo, and D. Howard, "Soft robotic sim2real via conditional flow matching," Advanced Intelligent Systems, p. e202500690, 2025.

[7] C. Della Santina, A. Bicchi, and D. Rus, "On an Improved State Parametrization for Soft Robots With Piecewise Constant Curvature and Its Use in Model Based Control," IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 1001–1008, Apr. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8961972/

[8] X. Zhang, F. Chan, T. Parthasarathy, and M. Gazzola, "Modeling and simulation of complex dynamic musculoskeletal architectures," Nature Communications, vol. 10, no. 1, pp. 1–12, 2019.

[9] J. Z. Zhang, Y. Zhang, P. Ma, E. Nava, T. Du, P. Arm, W. Matusik, and R. K. Katzschmann, "Sim2Real for Soft Robotic Fish via Differentiable Simulation," in 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2022, pp. 12 598–12 605, iSSN: 2153-0866.

[10] X. Lin, S. Liu, C. Liu, and Y. Wang, "Dynamic Modeling of Robotic Fish considering Background Flow using Koopman Operators," in 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2024, pp. 11 843–11 848, iSSN: 2153-0866.

[11] R. Vaxenburg, I. Siwanowicz, J. Merel, A. A. Robie, C. Morrow, G. Novati, Z. Stefanidi, G.-J. Both, G. M. Card, M. B. Reiser et al., "Whole-body physics simulation of fruit fly locomotion," Nature, vol. 643, no. 8074, pp. 1312–1320, 2025.

[12] I. Borazjani and F. Sotiropoulos, "Numerical investigation of the hydrodynamics of carangiform swimming in the transitional and inertial flow regimes," The Journal of experimental biology, vol. 211, pp. 1541–58, May 2008.

[13] T. Zhang, R. Tian, H. Yang, C. Wang, J. Sun, S. Zhang, and G. Xie, "From Simulation to Reality: A Learning Framework for Fish-Like Robots to Perform Control Tasks," IEEE Transactions on Robotics, vol. 38, no. 6, pp. 3861–3878, Dec. 2022.

[14] W. Liu, K. Bai, X. He, S. Song, C. Zheng, and X. Liu, "FishGym: A High-Performance Physics-based Simulation Framework for Underwater Robot Learning," Jun. 2022, number: arXiv:2206.01683 arXiv:2206.01683 [cs].

[15] X. Lin, X. Liu, and Y. Wang, "Learning Agile Swimming: An End-to-End Approach Without CPGs," IEEE Robotics and Automation Letters, vol. 10, no. 2, pp. 1992–1999, Feb. 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10835134/

[16] T.-H. Wang, P. Ma, A. E. Spielberg, Z. Xian, H. Zhang, J. B. Tenenbaum, D. Rus, and C. Gan, "Softzoo: A soft robot co-design benchmark for locomotion in diverse environments," arXiv preprint arXiv:2303.09555, 2023.

[17] J. H. Lee, J. Hu, S. Kwok, C. Majidi, and Z. Manchester, "A Unified Framework for Simulating Strongly-Coupled Fluid-Robot Multiphysics," Jun. 2025, arXiv:2506.05012 [cs].

[18] M. J. Lighthill, "Note on the swimming of slender fish," Journal of Fluid Mechanics, vol. 9, no. 2, pp. 305–317, Oct. 1960.

[19] J. Yu, J. Yuan, Z. Wu, and M. Tan, "Data-Driven Dynamic Modeling for a Swimming Robotic Fish," IEEE Transactions on Industrial Electronics, vol. 63, no. 9, pp. 5632–5640, Sep. 2016.

[20] Z. Chen, X. Tian, X. Chen, B. Wen, and X. Li, "An experimental study of the wire-driven compliant robotic fish," Ocean Engineering, vol. 279, p. 114433, Jul. 2023.

[21] B. Lu, C. Zhou, J. Wang, Z. Zhang, and M. Tan, "Toward Swimming Speed Optimization of a Multi-Flexible Robotic Fish With Low Cost of Transport," IEEE Transactions on Automation Science and Engineering, vol. 21, no. 3, pp. 2804–2815, Jul. 2024.

[22] A. Andersen, U. Pesavento, and Z. J. Wang, "Analysis of transitions between fluttering, tumbling and steady descent of falling cards," Journal of Fluid Mechanics, vol. 541, no. -1, p. 91, Oct. 2005.

[23] E. Nava, J. Z. Zhang, M. Y. Michelis, T. Du, P. Ma, B. F. Grewe, W. Matusik, and R. K. Katzschmann, "Fast Aquatic Swimmer Optimization with Differentiable Projective Dynamics and Neural Network Hydrodynamic Models," in Proceedings of the 39th International Conference on Machine Learning. PMLR, Jun. 2022, pp. 16 413–16 427, iSSN: 2640-3498.

[24] A. Saeed, H. Saeed, and J. Li, "Data-driven reduced-order modeling for undulating swimmers," Physics of Fluids, vol. 37, no. 7, Jul. 2025, publisher: AIP Publishing. [Online]. Available: https://pubs.aip.org/aip/pof/article/37/7/071903/3351664/Data-driven-reduced-order-modeling-for-undulating

[25] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in IEEE/RSJ International Conference on Intelligent Robots and Systems. Vilamoura-Algarve, Portugal: IEEE, Oct. 2012, pp. 5026–5033.

[26] N. Obayashi, A. Anastasiadis, J. Gumowski, K. Junge, K. L. Walker, K. Mülleners, and J. Hughes, "Scafi: Length-scalable, compliant, parametric robotic fish design for operation in multiple environmental niches," 2025.

[27] A. Lukezic, T. Vojir, L. Cehovin Zajc, J. Matas, and M. Kristan, "Discriminative Correlation Filter With Channel and Spatial Reliability," in CVPR, 2017, pp. 6309–6318.

[28] J. H. Lee, M. Y. Michelis, R. Katzschmann, and Z. Manchester, "Aquarium: A Fully Differentiable Fluid-Structure Interaction Solver for Robotics Applications," in International Conference on Robotics and Automation, Mar. 2023.

[29] Y. Zhong, Z. Li, and R. Du, "Robot fish with two-DOF pectoral fins and a wire-driven caudal fin," Advanced Robotics, vol. 32, no. 1, pp. 25–36, Jan. 2018.

[30] E. C. Garrido-Merchán and D. Hernández-Lobato, "Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes," Neurocomputing, vol. 380, pp. 20–35, 2020.

[31] F. Gao and L. Han, "Implementing the Nelder-Mead simplex algorithm with adaptive parameters," Comput. Optim. Appl., vol. 51, no. 1, pp. 259–277, Jan. 2012.

[32] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," Aug. 2018, arXiv:1801.01290 [cs].

[33] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," Journal of Machine Learning Research, vol. 22, no. 268, pp. 1–8, 2021.

[34] F. Boyer, M. Porez, and A. Leroyer, "Poincaré–cosserat equations for the lighthill three-dimensional large amplitude elongated body theory: Application to robotics," Journal of Nonlinear Science, vol. 20, no. 1, pp. 47–79, 2010.