Underwater Soft Robot Modeling and Control With Differentiable Simulation

Tao Du, Josie Hughes, Sebastien Wah, Wojciech Matusik, and Daniela Rus

***Abstract*—Underwater soft robots are challenging to model and control because of their high degrees of freedom and their intricate coupling with water. In this letter, we present a method that leverages the recent development in differentiable simulation coupled with a differentiable, analytical hydrodynamic model to assist with the modeling and control of an underwater soft robot. We apply this method to Starfish, a customized soft robot design that is easy to fabricate and intuitive to manipulate. Our method starts with data obtained from the real robot and alternates between simulation and experiments. Specifically, the simulation step uses gradients from a differentiable simulator to run system identification and trajectory optimization, and the experiment step executes the optimized trajectory on the robot to collect new data to be fed into simulation. Our demonstration on Starfish shows that proper usage of gradients from a differentiable simulator not only narrows down its simulation-to-reality gap but also improves the performance of an open-loop controller in real experiments.**

***Index Terms*—Learning for soft robots, calibration and identification, control, learning from experience, model learning for control, modeling.**

I. Introduction

DEVELOPMENTS in marine robotics provide many advantages for tasks such as underwater exploration, sample collection, and observation of marine wildlife [1]. Aquatic animals demonstrate the advantages of having a soft-body structure for swimming and navigating aquatic environments, highlighting how compliance and flexibility is a key component for efficient underwater locomotion [2] and motivating the design of soft robotic swimmers. Although a wide variety of methods have been developed for such robots [3]–[5], modeling and controlling them is still an open problem due to the infinite degrees of freedom of soft systems and the problem's computational overhead.

Manuscript received October 23, 2020; accepted February 27, 2021. Date of publication March 31, 2021; date of current version April 20, 2021. This letter was recommended for publication by Associate Editor F. Renda and Editor C. Laschi upon evaluation of the reviewers' comments. This work was supported by NSF under Grant EFRI-1830901, in part by IARPA under Grant 2019-19020100001, and in part by DARPA under Grant FA8750-20-C-0075. *(Tao Du and Josie Hughes contributed equally to this work.) (Corresponding author: Tao Du.)*

Tao Du, Josie Hughes, Wojciech Matusik, and Daniela Rus are with Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: taodu@csail.mit.edu; jaeh2@cam.ac.uk; wojciech@csail.mit.edu; rus@csail.mit.edu).

Sebastien Wah is with the MIT Department of Mechanical Engineering, Cambridge, MA 02139 USA (e-mail: sebwah@mit.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3070305, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3070305

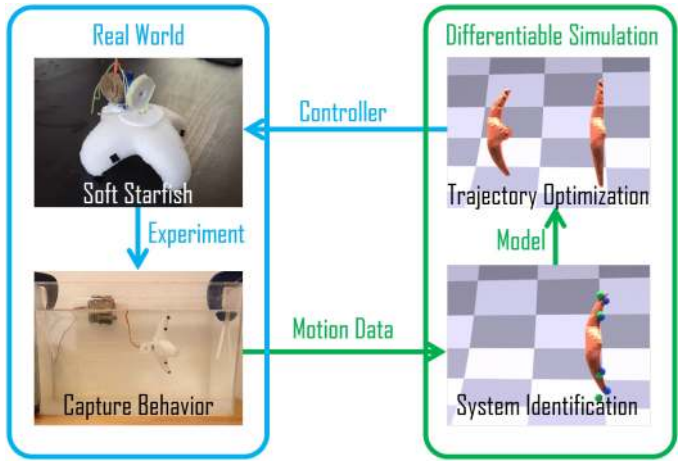

Fig. 1. An overview of the iterative, real-to-sim pipeline. Real-world motion data is captured from the robot for a given control input (left). This data is transferred to the differentiable simulator where system identification narrows the gap between reality (four green spheres bottom right) and simulation (four blue spheres bottom right) after which trajectory optimization is used to generate a new control sequence. Through iterative transfer we show trajectory optimization and reduction of the reality gap.

There has been a significant body of research focusing on the modeling of soft underwater systems. This includes modeling soft-body swimmers using discrete elastic rod simulation [6] and applying the Cosserat model to Cephalopod inspired soft robots [7]. Dynamic models have also been proposed for soft fish by combining bending beam theory with hydrodynamic and damping models [8], combining beam theory with fluidic models for modeling a compliant tail [9], and modeling fish bodies as multiple compliant rigid segments with hydrodynamic forces [10]. This body of work highlights the complexity of modeling not only the deformation of the compliant robot, but also accounting for the intricate solid-fluid coupling between a robot and water. The complex dynamics of both soft robots and their interaction with their aqueous environment typically leads to a significant *reality gap* between simulation and real experiments. To leverage the power of simulation, such a gap must be reduced. This will allow for increasingly reliable transfer of robot controllers and designs to the real world.

In this work, we present a method for modeling and controlling underwater soft robots with a focus on narrowing the simulation-to-reality gap (Fig. 1). Our core idea is to embed a *differentiable simulator* into a pipeline that alternates between simulated and real experiments. With gradient information readily available from a differentiable simulator, previous papers have

demonstrated promising results in various soft-robot applications, including system identification and controller design [11]–[15]. However, results from existing differentiable simulators are primarily focused on simulated robots, and demonstrations on real underwater soft robots have yet to be seen. Our work attempts to fill this gap by coupling differentiable simulation with a differentiable, analytical hydrodynamic model, to enable improved modeling and optimization of water-based soft systems.

Our pipeline starts by collecting motion data from a real underwater soft robot with synthetic control signals (Fig. 1, left). A dynamic model of the soft robot, including its actuators and its hydrodynamic forces, is initialized in simulation. The initial values of the model parameters are obtained from measurements on the soft robot and estimation from previous papers. Next, our pipeline compares the collected data and the motion predicted by the dynamic model in simulation, and gradients of the model parameters with respect to the error between the model and real world are automatically computed by the simulator to reduce the modeling error (Fig. 1, lower right). With an improved dynamic model in simulation, the pipeline then runs trajectory optimization to propose a new open-loop controller (Fig. 1, upper right), which is then executed on the hardware platform to collect more data for the next iteration (Fig. 1, left). The output of our pipeline is a calibrated dynamic model and an optimized open-loop controller that can be directly used on a real robot.

We demonstrate the efficacy of our pipeline on Starfish, a customized underwater soft robot design made of silicone foam and actuated with four tendons. Like many other existing soft robot designs, Starfish leverages non-symmetric placement of tendons or wires to achieve bending. Similar approaches have been shown to be effective in soft jellyfish robots [16], [17], and for undulating fish swimmers [18] including micro robot swimming fish [19]. Starfish is connected to a rail in a water tank to limit its motion to horizontal motion only. We find that our pipeline manages to not only narrow down the simulation to reality gap but also produce an effective open-loop controller after a few iterations, whose performance is increased significantly when compared to a handcrafted baseline controller.

II. Related Work

Modeling and controlling soft robots with assistance from simulation tools has been explored in a number of previous papers. For system identification, Real2Sim [14] is able to accurately reproduce motions of a real deformable specimen in simulation. However, gradients are derived with respect to material parameters only, limiting their capability of optimizing control signals. Conversely, large-scale simulation-to-reality transfer has been demonstrated for soft modular robots [20]. Trajectory optimization for soft robots is discussed in [21] with a special focus on terrestrial robots and locomotion tasks. Their work uses sensitivity analysis to obtain gradients from the dynamic equations but conducts system identification through trial and error. Finally, [15] proposes to control a soft tendon with a learned differentiable model. Compared to our pipeline, [15] uses a neural network model and assumes the motion is quasi-static, while our pipeline removes the quasi-static assumption and leverages an analytic dynamic model.

Our work is also closely related to prior efforts to designing and controlling soft fish [3]–[5]. With hydrodynamic forces playing an important role in the motion, obtaining an accurate dynamic model is even more difficult and existing papers rely heavily on design heuristics and trial and error for system identification. Our method connects differentiable simulation to this field, particularly for optimizing parameters in a hydrodynamic model. Our idea of differentiating an analytical hydrodynamic model is inspired by [22], which presents a differentiable Stokes flow simulator but does not consider soft robot applications.

Finally, our pipeline draws inspirations from recent advancements on narrowing the simulation-to-reality gap in the machine learning community [23]–[27]. In particular, the alternation between hardware and software in our pipeline shares similarities with model-based deep reinforcement learning methods in [27] or [23], but we leverage full gradient information from an analytic dynamic model in simulation. Another commonly used strategy for closing the reality gap is domain randomization, which trains the controller with randomized models in simulation [24], [25]. Essentially, domain randomization attempts to absorb modeling errors by training a robust but conservative controller. Our method is different from this family of methods in that we attempt to directly reduce modeling errors by improving the model parameter estimation.

III. Simulation

We now describe our simulation model for Starfish as well as its implementation in a differentiable simulator. We choose to base our simulator implementation on DiffPD [11], a recent differentiable soft-body simulator that supports fast and robust implicit time integration with contact handling. We augment DiffPD by implementing a differentiable actuator and hydrodynamic model with trainable parameters in optimization. It is worth mentioning that our pipeline is agnostic to the choice of differentiable simulators.

A. Governing Equations

We model the body of Starfish using the finite element method (FEM) with a tetrahedral discretization. An implicit Euler time-stepping scheme is used because of its numerical robustness and large time steps. Let n be the number of nodes after discretization and let $\mathbf{x}_i \in \mathbb{R}^{3n}$ and $\mathbf{v}_i \in \mathbb{R}^{3n}$ be the nodal positions and velocities at the i-th time step. The governing equations can be written as follows:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{v}_{i+1}, \tag{1}$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \frac{h}{m}[\mathbf{f}_e(\mathbf{x}_{i+1}) + \mathbf{f}_h(\mathbf{x}_i, \mathbf{v}_i) + \mathbf{f}_a(\mathbf{x}_{i+1}, \mathbf{a}_i)]. \tag{2}$$

Here, h is the time step ($1/60$ second in our simulation), m the mass of a node, $\mathbf{f}_e$ the elastic force computed from the material model, $\mathbf{f}_h$ the hydrodynamic force, and $\mathbf{f}_a$ the actuation force dependent on the action $\mathbf{a}_i$ at this time step. Equations (1) and (2)

contain parameters whose values need to be determined from the real system to build an accurate dynamic model. These include the material parameters in $\mathbf{f}_e$, the hydrodynamic parameters in $\mathbf{f}_h$, and the actuator parameters in $\mathbf{f}_a$, which are described in detail in the following three subsections.

B. Material Model

We use the same material model as described in [11] and [28], which is based on the corotated linear material model [29]. The material model has three parameters that require either a direct measurement or a reliable estimation: its density, its Young's modulus, and its Poisson's ratio. We compare these parameters to the product specification [30] of the silicon foam used (SomaFoam 25) to identify its density ($400\ \text{kg/m}^3$) and Poisson's ratio (0.48) reliably. The Young's modulus E is not included in the product specification, so we estimate its value based on the reported shore hardness and the Gent's relation [31]. Since this is an empirical estimation with a possibly large uncertainty, we make E a trainable parameter in our optimization.

C. Hydrodynamic Forces

We model hydrodynamic effects as external and explicit forces added to the system. For the sake of speed and simplicity in gradient computation, we choose to compute thrust and drag from water with a widely used approximation in aerodynamics [11], [26], [28]:

$$\mathbf{f}_d = \frac{1}{2}\rho A C_d(\alpha)\|\mathbf{v}_{rel}\|_2\mathbf{v}_{rel}, \tag{3}$$

$$\mathbf{f}_t = -\frac{1}{2}\rho A C_t(\alpha)\|\mathbf{v}_{rel}\|_2^2\mathbf{n}. \tag{4}$$

Here, $\mathbf{f}_d$ and $\mathbf{f}_t$ represent the drag and thrust forces evaluated on every triangle on the surface of Starfish after discretization. We use $\boldsymbol{\rho}$ to represent the density of water and A the area of the triangle. $\mathbf{v}_{rel} \in \mathbb{R}^3$ is the relative velocity between Starfish and the flow of water, and α is the angle of attack. C_d and C_t are scalar functions computing the drag and thrust coefficients. For aerodynamic applications, C_d and C_t are typically measured by conducting wind tunnel experiments. For our underwater experiments, however, a direct measurement of C_d and C_t is difficult. Therefore, we represent C_d and C_t as B-splines and make their control points trainable in our optimization. We initialize the B-splines with the curves suggested in [28] and optimize their shapes with the gradients calculated in the differentiable simulator.

D. Actuators

As we use tendons inside foam to actuate Starfish, we model the actuation with an anisotropic elastic energy which exerts large forces along the tendon direction [11], [28]. For a tetrahedron through which the tendon passes, the associated anisotropic elastic energy is defined as follows:

$$E_a = \frac{w}{2}\|(1-a_i)\mathbf{F}\mathbf{m}\|_2^2 \tag{5}$$

TABLE I
A SUMMARY OF ALL TRAINABLE MODEL PARAMETERS

Name	Definition	Initial guess
E	Young's modulus	0.9MPa, estimated from [30].
w	The actuator's stiffness	2MPa, from [11].
P_{C_d}	Four control points of C_d	From [28], Fig. 2.
P_{C_t}	Four control points of C_t	From [28], Fig. 2.

where w is a prespecified stiffness, $\mathbf{F}$ is the deformation gradient, $\mathbf{m}$ is the direction of the tendon, and $a_i \in [0,1]$ is the control signal. Smaller a_i indicates greater contraction along the tendon direction. The actuation force $\mathbf{f}_a$ for each node is then computed by aggregating E_a from its adjacent tetrahedrons and calculating its spatial gradients. Our actuator model has one trainable parameter w in our optimization.

IV. OPTIMIZATION

Our ultimate goal is to find an open-loop controller for Starfish that maximizes its forward velocity. In this section, we describe how we combine the simulation model and real motion data to achieve this goal. Our core idea is to leverage the gradients from the differentiable simulator to improve both the dynamic model (i.e., system identification) and the open-loop control signals via trajectory optimization.

A. Problem Definition

We abstract the simulation model in Section III as follows:

$$\mathbf{s}_{i+1} = \mathbf{DiffSim}(\mathbf{s}_i, \mathbf{a}_i; \boldsymbol{\theta}) \tag{6}$$

where $\mathbf{s}_i = (\mathbf{x}_i, \mathbf{v}_i)$ represents the state of the robot at the i-th time step, $\mathbf{a}_i$ is the actuation signal as described before, $\boldsymbol{\theta}$ represents model parameters, and $\mathbf{DiffSim}$ can be any black-box differentiable simulator that computes $\mathbf{s}_{i+1}$, the new state of the system after one time step. The model parameters $\boldsymbol{\theta}$ consists of all trainable parameters in Section III, which we summarize in Table I.

The decision variables to be optimized in our problem are the model parameters $\boldsymbol{\theta}$ and the sequence of actions $\mathbf{a}_i$. For optimizing $\boldsymbol{\theta}$, we consider minimizing the following objective $L_{\boldsymbol{\theta}}$:

$$\min_{\boldsymbol{\theta}} \quad L_{\boldsymbol{\theta}} \tag{7}$$

$$s.t. \quad L_{\boldsymbol{\theta}} = \sum_i \|\mathbf{s}_i - \mathbf{s}_i^*\|^2 \tag{8}$$

$$\mathbf{s}_{i+1} = \mathbf{DiffSim}(\mathbf{s}_i, \mathbf{a}_i^*; \boldsymbol{\theta}); \quad \mathbf{s}_0 = \mathbf{s}_0^* \tag{9}$$

where $\mathbf{s}_i^*$ and $\mathbf{a}_i^*$ refer to the state and the action signal from the measurement at the i-th time step. In short, we adjust $\boldsymbol{\theta}$ to match the motion of Starfish in reality to its counterpart in simulation.

The objective for optimizing $\mathbf{a}_i$ is defined as $L_{\mathbf{a}}$ below:

$$\min_{\mathbf{a}_i} \quad L_{\mathbf{a}} \tag{10}$$

$$s.t. \quad L_{\mathbf{a}} = COM(\mathbf{s}_N) - COM(\mathbf{s}_0) \tag{11}$$

$$\mathbf{s}_{i+1} = \mathbf{DiffSim}(\mathbf{s}_i, \mathbf{a}_i; \boldsymbol{\theta}); \quad \mathbf{s}_0 = \mathbf{s}_0^* \tag{12}$$

Algorithm 1: Co- Optimize Model (θ) and Actions ($\mathbf{a}_i$).

Input: Initial θ and $\mathbf{a}_i$;
Output: Optimized θ and $\mathbf{a}_i$;
while *experiments do not converge* **do**
  // Hardware experiment;
  Execute $\mathbf{a}_i$ on the hardware to collect $\mathbf{s}_i^*$ and $\mathbf{a}_i^*$;
  // Check convergence;
  Use $\mathbf{s}_N^*$ and $\mathbf{s}_0^*$ to compute $L_{\mathbf{a}_i}$;
  if $L_{\mathbf{a}_i}$ *is similar to the last iteration* **then**
    // Convergence;
    break;
  // System identification;
  Minimize L_θ (Eqn. (7)) to update θ;
  // Trajectory optimization;
  Minimize $L_{\mathbf{a}}$ (Eqn. (10)) to update $\mathbf{a}_i$;

where N is the index of the last time step considered in this trajectory optimization problem and $COM(\mathbf{s})$ computes the x coordinate of the center of mass from state $\mathbf{s}$. We define the center of mass as the average of all vertices from $\mathbf{s}$. Since the forward direction of our Starfish is along the negative x axis, minimizing $L_{\mathbf{a}}$ will maximize the traveling distance as desired. Note that $\boldsymbol{\theta}$ is fixed in this optimization problem.

For a standard differentiable soft-body simulator, the procedure of computing the gradients with respect to $\boldsymbol{\theta}$ and $\mathbf{a}_i$ is well documented in previous work [12], which interested readers can refer to for more details. We use L-BFGS, a gradient-based quasi-Newton method to solve the two optimization problems above.

B. Optimization Algorithm

We now present an alternating scheme to improve both the model parameter $\boldsymbol{\theta}$ and the sequence of actions $\mathbf{a}_i$. Our optimization process starts with an initial guess of $\boldsymbol{\theta}$ (Table I) and $\mathbf{a}_i$ (a synthetic control signal). We execute $\mathbf{a}_i$ on the hardware and collect the measurement $(\mathbf{s}_i^*, \mathbf{a}_i^*)$. Note that $\mathbf{a}_i^*$ and $\mathbf{a}_i$ differ slightly because the motor quantizes the real number $\mathbf{a}_i$ into integers. We then use $\mathbf{a}_i^*$ and $\mathbf{s}_i^*$ to minimize (7) and obtain an improved model parameter $\boldsymbol{\theta}$. Next, with the optimized dynamic model, we run trajectory optimization to update $\mathbf{a}_i$. Finally, we test $\mathbf{a}_i$ on the hardware, initiating the next round of experiment and closing the optimization loop in our pipeline. Algorithm 1 summarize the whole optimization algorithm in pseudo-code.

V. Results and Discussions

A. Hardware Setup

1) Fabrication: The fabrication method has been chosen to allow for transfer from a 3D CAD model to a real-world robot while minimizing the "fabrication gap" between the real and simulated system. Starfish is fabricated by creating an inverse mould into which silicone foam (SomaFoam 25, SmoothOn) is cast. Silicone foam, a material widely used for soft robotic fabrication [32], has been chosen as it allows for rapid fabrication, shows elastic properties, and has natural buoyancy. The "muscle fibers" or tendons can then be routed into the soft structure along the bottom of each of the legs of Starfish. The tendons are inserted using a thin metal tube through which the tendon fibers (non-extensible fishing line) can be inserted, and the tube removed. Each tendon is fixed at the end of each leg using adhesive and connected to a servo motor via an incompressible tube which runs through the center of the body of Starfish where they are connected to the servo by a pulley. The servo can contract the tendons, flexing the legs inwards, and then extend the tendons, to flex the legs to their initial position. The motion is highlighted in Fig. 2. It is important that the tendon length can be set accurately for effective sim-to-real transfer, as such the servo was chosen to have a torque which is higher than the load. This was validated by performing no-load and load tests and observing that the servo position is not significantly affected by the load. The servo position is controlled by a microcontroller which sets the position via a PWM signal.

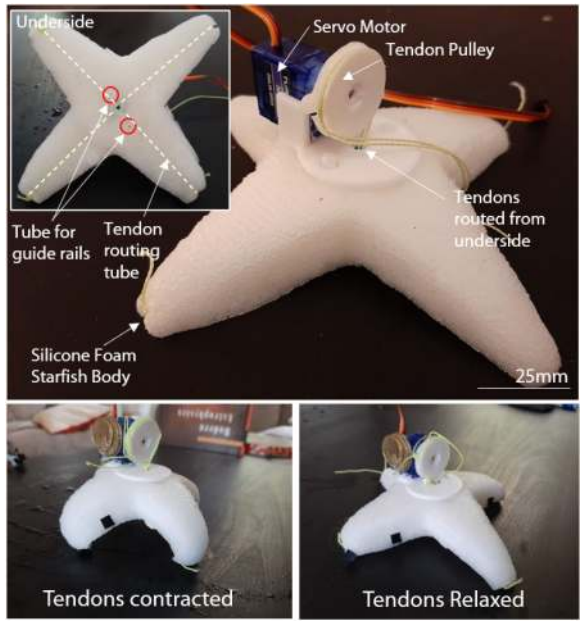

Fig. 2. The fabricated soft robot (top) showing the servo-based mechanism and the inset figure showing the underside and the tendon routing. The contracted and relaxed pose of the robot are shown in the bottom pictures.

2) Experimental Setup: A tank-based experimental setup has been created for testing the robot. To constrain the robot in an orientation that allows for motion capture, the tank system has been fitted with horizontal rails constructed from fishing twine. Starfish has low friction PTFE tubing through the body through which the guide rails run. The use of low friction materials and the presence of water results in the rails exhibiting low friction and enabling the robot to move forwards while the orientation is fixed. The rails do provide some negating frictional force potentially reducing the forward velocity. However, we expect this to be minimal. The weight of the robot has been adjusted such that it has approximately neutral buoyancy at the depth the rails are within the tank. Fig. 3 shows the experimental setup.

To capture motion data from the soft robot, a high-speed camera (Logitech BRIO) has been fixed outside the tank. Four

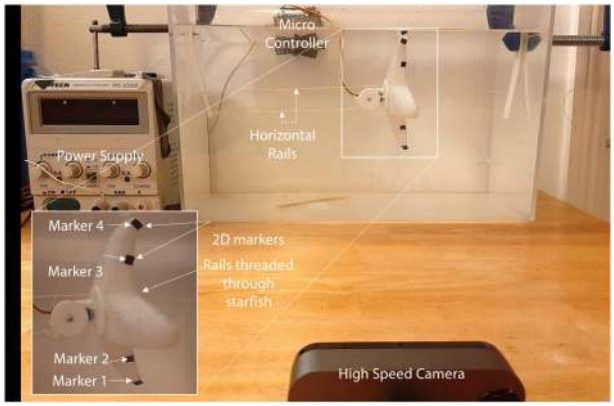

Fig. 3. Experimental setup showing the tank with the horizontal rails and the robot. The high-speed camera and markers allow the motion of the robot to be captured.

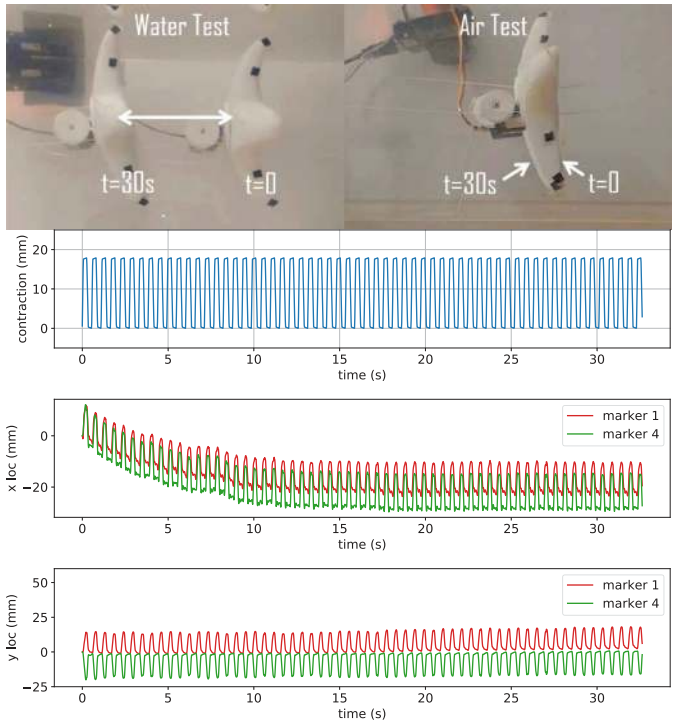

Fig. 4. Top row: The trajectories of Starfish running the same cyclic control sequence with water (left) and without water (right). The robot made little progress when water was not present. Bottom three: A cyclical control input (upper middle) is applied to the robot, with the outer most markers (marker 1 and marker 4) positions recorded. The tracked horizontal marker position (lower middle) and vertical marker position (bottom) show that after some initial transients, repeatable and cyclical movement is achieved.

black markers were attached to one side of Starfish at locations which capture the most dynamic information about the robot. To capture the 2D motion data from these markers, the video feed was calibrated using a standard checkerboard and the marker locations extracted by tracking features corresponding to the makers throughout the video. In each experiment, the motion of the robot and the control sequence (i.e. the length of contraction of the tendon) is recorded at 60 Hz.

B. Experimental Verification

To verify the experimental setup and ensure that the interaction between the fluid and Starfish is the direct cause of any resultant forward motion, we show in Fig. 4 the motion of Starfish when the tank is both full of water and empty with a given cyclic sequence of actions. The fact that the robot barely moved without water shows the influence of solid-fluid interaction and the necessity of calibrating the hydrodynamic model.

In addition to verifying the experimental setup, the repeatability and reliability of the experiments must be demonstrated to show that experiments are representative. To show this, we ran the robot with a cyclic sequence of actions and observed cyclic motions were established after the initial transient state of water (Fig. 4), indicating that repeated control signals lead to reproducible motions in our experiments.

C. Baseline Algorithms

To better evaluate the performance of our algorithm, we propose two baselines for comparison: **bl-ctrl** and **bl-one-iter**. Both baselines provide an open-loop controller that attempts to maximize the traveling distance of Starfish, which is our ultimate goal in physical experiments. The **bl-ctrl** baseline proposes to use a sinusoidal sequence of actions with an educated guess on its frequency and amplitude without further optimization. The role of this baseline is to understand if the problem can be solved trivially by a carefully chosen handcrafted solution. The **bl-one-iter** baseline simply runs our pipeline for 1 iteration and terminates, i.e., it conducts system identification and optimizes $\mathbf{a}_i$ exactly once. Comparing our pipeline to **bl-one-iter** will evaluate the necessity of alternating between system identification and trajectory optimization for multiple iterations. To ensure a fair comparison, we use **bl-ctrl** as the initial guess of $\mathbf{a}_i$ in both **bl-one-iter** and our pipeline (Algorithm 1).

D. Optimization Results

We now report the progress of our optimization pipeline and the performance of the optimized controller both in simulation and in reality. Note that the progress of our pipeline also covers the performance of the two baselines above. This is because **bl-ctrl** and **bl-one-iter** can be interpreted as terminating our pipeline at the beginning and after 1 iteration, respectively. We optimize a 3-second-long sequence of action in simulation and test it on Starfish for 30 seconds by repeating the sequence 10 times. Table II summarizes the system identification loss $L_{\boldsymbol{\theta}}$, the trajectory optimization loss $L_{\mathbf{a}}$, and the average velocity of the robot in the simulation environments (v_s) and the physical experiments (v_r). At each iteration, lower $L_{\boldsymbol{\theta}}$ and $L_{\mathbf{a}}$ and higher v_r are better. We have also reported the optimized Young's modulus E, the actuator stiffness w, and the control points of C_d and C_t at each iteration in Table II. To visualize the optimized results, we render the motion of the simulated robot before and after each iteration's optimization in Fig. 5 and plot the optimized control signals in Fig. 7. Finally, we show the

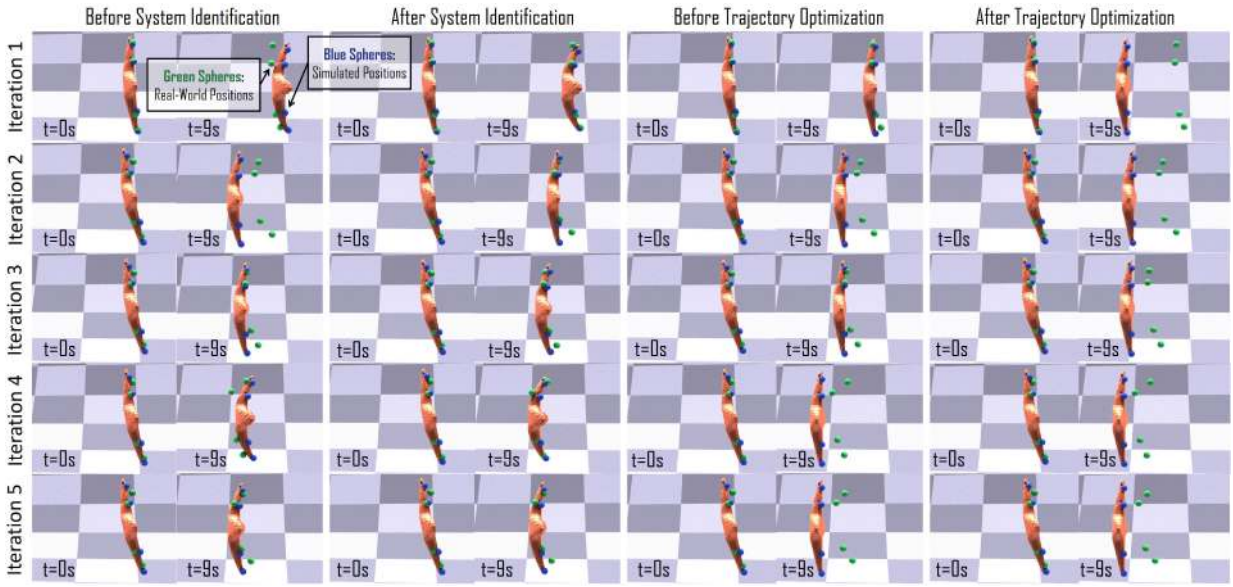

Fig. 5. Left two columns: The motion of our simulated Starfish before (column 1) and after system identification (column 2) for the 1st to 5th iterations. The blue and green spheres indicate the 4 marker's locations in simulation and from real experiments, respectively. We visualize the motion of the robot at $t = 0$ and $t = 9$ seconds. The goal of this optimization is to narrow the distance between each pair of blue and green spheres. Right two columns: the motion before (column 3) and after trajectory optimization (column 4). The goal of this optimization is to push the robot towards its left as much as possible. Note that we assume the motion to be solved in trajectory optimization is cyclic with a period of 3 seconds.

TABLE II
THE OPTIMIZATION PROGRESS OF ALGORITHM 1

Iter.	L_θ	$L_\mathbf{a}$	v_s $(cm\,s^{-1})$	v_r $(cm\,s^{-1})$	E	w
0 (**bl-ctrl**)	9.2e-2	-1.2e-2	0.01	0.21	9.0e5	2.0e6
1 (**bl-one-iter**)	2.9e-2	-3.8e-2	0.83	0.48	5.0e5	4.1e6
2	7.7e-2	-3.4e-2	0.68	0.56	1.0e6	1.4e6
3	5.1e-2	-3.5e-2	0.66	0.67	4.3e5	4.8e6
4	5.2e-2	-3.9e-2	0.77	0.75	4.0e5	5.7e6
5	7.5e-2	-3.9e-2	0.75	0.75	3.8e5	5.8e6
Iter.	$P^1_{C_d}$	$P^2_{C_d}$	$P^3_{C_d}$	$P^4_{C_d}$		
0	0.1	0.1	1.9	2.1		
1	0.0	0.0	0.5	2.0		
2	0.2	0.2	0.7	2.4		
3	0.0	0.0	0.9	2.5		
4	0.0	0.0	0.8	2.2		
5	0.0	0.0	0.8	2.2		
Iter.	$P^1_{C_t}$	$P^2_{C_t}$	$P^3_{C_t}$	$P^4_{C_t}$		
0	-0.8	-0.5	0.1	2.5		
1	-0.5	0.0	1.0	3.0		
2	-0.5	-0.2	0.7	3.0		
3	-0.6	-0.5	0.0	3.0		
4	-0.6	-0.2	0.5	3.0		
5	-0.6	-0.2	0.5	3.0		

performance of our optimized open-loop controller in real-world experiments in Fig. 8.

E. Discussion

1) Comparisons to Baselines: By comparing the quantitative results between different methods in Table II, we reach the following conclusions: First, the control signal proposed by **bl-ctrl** performs poorly without further system identification or trajectory optimization. The real robot travelled at 0.21 cm s^{-1}, corresponding to only 6 centimeters during the 30-second-long test time. This shows that finding an open-loop controller for an underwater soft robot is not a trivial task. Second, and more importantly, we noticed that the traveling velocity measured in real experiments increases monotonically with each iteration until the optimization process converges. After 1 iteration, the

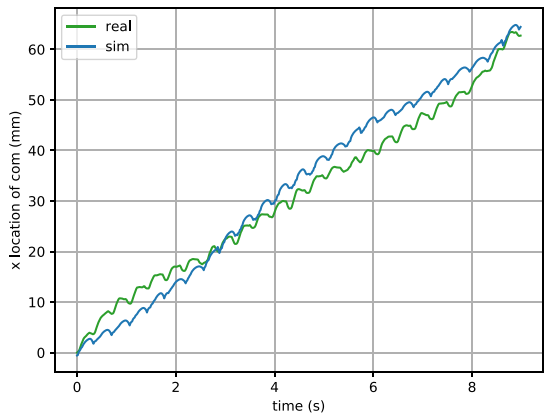

Fig. 6. The motion of the robot's center of mass in simulation and in the real experiments with the identical control signal. The simulation data is computed with the dynamic model after our algorithm converges.

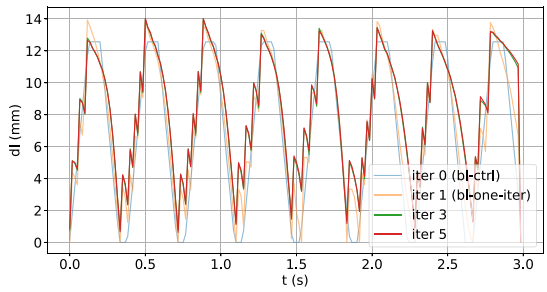

Fig. 7. The optimized control sequences $\mathbf{a}_i$, reported as the tendon contraction (dl in millimeters) from our method and two baselines.

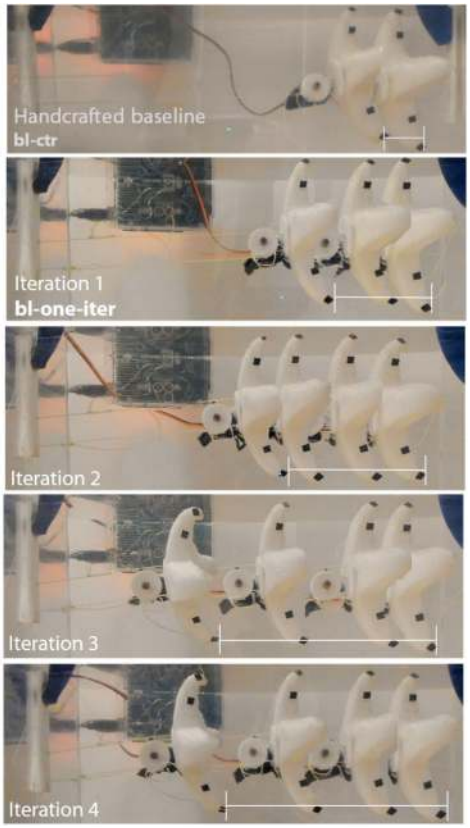

Fig. 8. Overlaid motion sequence showing the progress made by the robot in the fixed time period (30 seconds) when using the baseline handcrafted control sequence (top), the **bl-one-iter** baseline control sequence (upper middle), and control sequences for the second to fourth iteration.

traveling velocity from **bl-one-iter** (0.48 cm s^{-1}) is more than twice that of **bl-ctrl** (0.21 cm s^{-1}). This trend continues until the algorithm convergences after three more iterations with a velocity of 0.75 cm s^{-1}. Such an improvement after each iteration highlights the value of running Algorithm 1 for multiple iterations.

2) Simulation-to-Reality Gap: Another metric of success in our experiments is whether the sim-to-real gap has been narrowed after optimization. The sim-to-real gap measures the discrepancy between the dynamic model in simulation and the robot in real experiments, which can be understood by answering two questions: First, does the dynamic model fit the given measurement data well? Second, can the model predict new behaviors accurately? Such questions can also be motivated from the classic bias-variance tradeoff in machine learning, which aims to explain the expressiveness and generalizability of a model.

To answer the first question, we refer readers to the second column of Fig. 5. By definition, the distance between the measured and simulated marker positions (green and blue spheres respectively) is a direct, quantitative metric of the fitting error of our model (see also the $L_{\boldsymbol{\theta}}$ column in Table II). By comparing column 1 and 2, we can see that our system identification step manages to explain the measurement data well, as indicated by the closer distance between blue and green spheres after optimization.

To answer the second question, i.e., the generalizability of our dynamic model after system identification, we compare the simulated and actual motions of the robot with a sequence of action *not seen* in the training process of the dynamic model. Such a comparison is reflected in the first column of Fig. 5, i.e., the simulated and actual motions *before* system identification at each iteration. Note that the simulated and actual motions in this column execute the same sequence of action, but the corresponding motion capture data have not been used to train the dynamic model yet (which is what the algorithm is about to do afterwards). This also serves as a direct measurement of the reality gap, i.e., if we run the simulated and actual robot with exactly the same control signal, how different could the motions be? To quantify the motion difference, we report in Table II the average velocity from this motion in simulation (the v_s column) and the real experiments (the v_r column). It can be seen that the difference between v_s and v_r becomes significantly smaller as the algorithm proceeds with more iterations. Specifically, the two velocities become almost identical after only 3 iterations, indicating our algorithm's good generalizability as well as its effort into narrowing the sim-to-real gap. To visualize the motions more thoroughly, we plot the location of the robot's center of mass in these two motions obtained at the last iteration of our algorithm in Fig. 6. It can be seen that although the absolute location of the center of mass still differs from time to time between simulation and reality, the simulated motion exhibits local, oscillating patterns that are very similar to its real-world counterpart. The full motion sequences can be found in our video and in Fig. 8.

3) Optimized Controller: To better understand the optimized control sequence, we plot the intermediate controllers after each iteration in Fig. 7. Comparing to the baseline controller proposed in **bl-ctrl**, we notice the optimizer made two significant changes to the control sequence: First, it increased its amplitude by about 16% (from 12 mm to 14 mm). Second, it injects very high-frequency signals from time to time. We believe that both changes allow Starfish to leverage hydrodynamic forces more effectively and lead to the longer traveling distance.

VI. Conclusion

Computational tools for dynamic modeling and controller development of soft robotics have the potential to change how we design and control soft robots. However, the modeling of soft structures and environmental interactions make this challenging. Differentiable simulators offer potential advantages as they expose gradient information and also show computation efficiency. We proposed a pipeline for the development of an open-loop controller for a swimming soft robot using a differentiable simulator in which we iteratively loop between the real and simulated worlds. We demonstrate this approach on a simple four-legged starfish-shaped soft swimming robot. Within four iterations, we show the forward swimming velocity can be increased by a factor of 3.6 in comparison to a handcrafted baseline. In addition, we show that the gap between the simulated and real-world robot can be reduced, with the simulation showing realistic dynamic behaviors.

While this approach demonstrates how the simulation-to-reality gap can be *qualitatively* reduced, there still remains a *quantitative* gap between the simulated and real world performance, as can be seen from the discrepancy between the simulated and actual center-of-mass motions in Fig. 6. We believe this is due to inaccuracies in hydrodynamic force and actuator modeling. The pipeline concept we propose can be used interchangeably with alternative simulators and other robots. This could allow for simulators with alternative hydrodynamic models to be explored as a means as further reducing the sim-to-real gap. However, despite this, we still demonstrate how the performance of the robot can be improved, showing that despite the existence of a quantitative reality gap, by reducing the qualitative reality gap, optimization is still successful and contributes to the performance improvement observed in real experiments.

In this first demonstration of the iterative use of a differentiable simulation for system identification and trajectory optimization, we have considered a relatively simplistic robot system. Our Starfish has only a single control signal for all four limbs. For robots with an increased number of actuators, the system identification and control problem becomes more challenging. The iterative approach we propose may become increasingly beneficial as it allows for a wider range of conditions to be experienced which otherwise may not be observed. This is a trade-off with the additional complexity. Future work should investigate how the iterative nature of the pipeline can be optimized by considering control sequences not only optimized for forwards locomotion but also system identification.

REFERENCES

[1] R. Bogue, "Underwater robots: A review of technologies and applications," *Ind. Robot.*, vol. 42, no. 3, pp. 186–191, 2015. [Online]. Available: https://doi.org/10.1108/IR-01-2015-0010

[2] A. Pabst, "Springs in swimming animals," *Amer. Zoologist*, vol. 36, no. 6, pp. 723–735, 1996.

[3] R. Katzschmann, J. DelPreto, R. MacCurdy, and D. Rus, "Exploration of underwater life with an acoustically controlled soft robotic fish," *Sci. Robot.*, Science Robotics, vol. 3, no. 16, 2018, doi: 10.1126/scirobotics.aar3449.

[4] T. Li *et al.*, "Fast-moving soft electronic fish," *Sci. Adv.*, vol. 3, no. 4, 2017, Art. no. 1602045.

[5] J. Shintake, V. Cacucciolo, H. Shea, and D. Floreano, "Soft biomimetic fish robot made of dielectric elastomer actuators," *Soft Robot.*, vol. 5, no. 4, pp. 466–474, 2018.

[6] W. Huang, Z. Patterson, C. Majidi, and M. K. Jawed, "Modeling soft swimming robots using discrete elastic rod method," in *Proc. Bioinspired Sensing, Actuation, Control Underwater Soft Robotic Syst.*, 2021, pp. 247–259.

[7] F. Renda, F. Giorgio-Serchi, F. Boyer, and C. Laschi, "Modelling cephalopod-inspired pulsed-jet locomotion for underwater soft robots," *Bioinspiration Biomimetics*, vol. 10, no. 5, 2015, Art. no. 0 55005.

[8] H. El-Daou, T. Salumäe, L. D. Chambers, W. M. Megill, and M. Kruusmaa, "Modelling of a biologically inspired robotic fish driven by compliant parts," *Bioinspiration Biomimetics*, vol. 9, no. 1, 2014, Art. no. 0 16010.

[9] V. Kopman, J. Laut, F. Acquaviva, A. Rizzo, and M. Porfiri, "Dynamic modeling of a robotic fish propelled by a compliant tail," *IEEE J. Ocean. Eng.*, vol. 40, no. 1, pp. 209–221, Jan. 2014.

[10] J. Wang, P. K. McKinley, and X. Tan, "Dynamic modeling of robotic fish with a base-actuated flexible tail," *J. Dyn. Syst., J. Dynamic Syst., Meas. Control*, vol. 137, no. 1, Aug. 2014, doi: 10.1115/1.4028056.

[11] T. Du *et al.*, "DiffPd: Differentiable projective dynamics with contact," 2021, *arXiv:2101.05917*.

[12] Y. Hu *et al.*, "ChainQueen: A real-time differentiable physical simulator for soft robotics," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 6265–6271.

[13] Y. Hu *et al.*, "DiffTaichi: Differentiable programming for physical simulation," in *Proc. Int. Conf. Learn. Representations*, 2020.

[14] D. Hahn, P. Banzet, J. Bern, and S. Coros, "Real2Sim: Visco-elastic parameter estimation from dynamic motion," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 1–13, 2019.

[15] J. Bern, Y. Schnider, P. Banzet, N. Kumar, and S. Coros, "Soft robot control with a learned differentiable model," in *Proc. 3rd IEEE Int. Conf. Soft Robot.*, 2020, pp. 417–423.

[16] A. Villanueva, C. Smith, and S. Priya, "A biomimetic robotic jellyfish (robojelly) actuated by shape memory alloy composite actuators," *Bioinspiration Biomimetics*, vol. 6, no. 3, 2011, Art. no. 0 36004.

[17] Y. Almubarak, M. Punnoose, N. X. Maly, A. Hamidi, and Y. Tadesse, "Kryptojelly: A jellyfish robot with confined, adjustable pre-stress, and easily replaceable shape memory alloy niti actuators," *Smart Mater. Structures*, vol. 29, no. 7, 2020, Art. no. 0 75011.

[18] Y. Zhong, Z. Li, and R. Du, "A novel robot fish with wire-driven active body and compliant tail," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 4, pp. 1633–1643, Aug. 2017.

[19] Z. Wang, G. Hang, J. Li, Y. Wang, and K. Xiao, "A micro-robot fish with embedded SMA wire actuated flexible biomimetic fin," *Sensors Actuators A Phys.*, vol. 144, no. 2, pp. 354–360, 2008.

[20] S. Kriegman *et al.*, "Scalable sim-to-real transfer of soft robot designs," in *Proc. 3rd IEEE Int. Conf. Soft Robot.*, 2020, pp. 359–366.

[21] J. Bern, P. Banzet, R. Poranne, and S. Coros, "Trajectory optimization for cable-driven soft robot locomotion," in *Proc. Robot. Sci. Syst.*, Freiburgim Breisgau, Germany, Jun. 2019, doi: 10.15607/RSS.2019.XV.052.

[22] T. Du, K. Wu, A. Spielberg, W. Matusik, B. Zhu, and E. Sifakis, "Functional optimization of fluidic devices with differentiable stokes flow," *ACM Trans. Graph.*, vol. 39, no. 6, pp. 1–15, 2020.

[23] Y. Chebotar *et al.*, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 8973–8979.

[24] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 23–30.

[25] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 3803–3810.

[26] J. Xu *et al.*, "Learning to fly: Computational controller design for hybrid UAVs with reinforcement learning," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–12, 2019.

[27] M. P. Deisenroth, C. E. Rasmussen, and D. Fox, "Learning to control a low-cost manipulator using data-efficient reinforcement learning," *Robot. Sci. Syst. VII*, vol. 7, pp. 57–64, 2011.

[28] S. Min, J. Won, S. Lee, J. Park, and J. Lee, "Softcon: Simulation and control of soft-bodied animals with biomimetic actuators," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 1–12, 2019.

[29] E. Sifakis and J. Barbic, "Fem simulation of 3D deformable solids: A practitioner's guide to theory, discretization, and model reduction," in *Proc. ACM Siggraph Courses*, 2012, pp. 1–50.

[30] "Soma foama 25 product specification," Accessed: Feb. 7, 2021. [Online]. Available: https://www.smooth-on.com/tb/files/SOMA_FOAMA_TB.pdf

[31] A. N. Gent, "On the relation between indentation hardness and Young's modulus," *Rubber Chem. Technol.*, vol. 31, no. 4, pp. 896–906, 1958.

[32] L. Somm, D. Hahn, N. Kumar, and S. Coros, "Expanding foam as the material for fabrication, prototyping, and experimental assessment of low-cost soft robots with embedded sensing," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 761–768, Apr. 2019.