ADVANCED INTELLIGENT SYSTEMS
Open Access
www.advintellsyst.com

Design and Closed-Loop Motion Planning of an Untethered Swimming Soft Robot Using 2D Discrete Elastic Rods Simulations

Xiaonan Huang, Zach J. Patterson, Andrew P. Sabelhaus, Weicheng Huang, Kiyn Chin, Zhijian Ren, Mohammad Khalid Jawed, and Carmel Majidi**

Despite tremendous progress in the development of untethered soft robots in recent years, existing systems lack the mobility, model-based control, and motion planning capabilities of their piecewise rigid counterparts. As in conventional robotic systems, the development of versatile locomotion of soft robots is aided by the integration of hardware design and control with modeling tools that account for their unique mechanics and environmental interactions. Here, a framework for physics-based modeling, motion planning, and control of a fully untethered swimming soft robot is introduced. This framework enables offline co-design in the simulation of robot parameters and gaits to produce effective open-loop behaviors and enables closed-loop planning over motion primitives for feedback control of a frog-inspired soft robot testbed. This pipeline uses a discrete elastic rods (DERs) physics engine that discretizes the soft robot as many stretchable and bendable rods. On hardware, an untethered aquatic soft robot that performs frog-like rowing behaviors is engineered. Hardware validation verifies that the simulation has sufficient accuracy to find the best candidates for sets of parameters offline. The simulator is then used to generate a trajectory library of the robot's motion in simulation that is used in real-time closed-loop path following experiments on hardware.

1. Introduction

The range of materials and hardware architectures used in robotics has expanded dramatically over the past decade—extending from rigid plastics and metals for hard-cased motors and electronics to soft elastomers and fluids for bio-inspired artificial muscle and soft-bodied robots.[1] Much of this has been driven by interdisciplinary fields like bio-inspired engineering,[2] soft robotics,[3,4] and wearable computing[5] that seek to achieve robotic functionalities with soft and deformable materials that match the compliance and elasticity of natural biological tissue. Progress in "soft-matter engineering" with elastomers, fluids, and other forms of condensed soft matter[6] has led to recent interest in creating more complex and sophisticated soft robotic systems that are fully untethered[7] and capable of matching the mobility of natural biological organisms.[8] Of particular interest has been the ability to mimic invertebrate marine

X. Huang
Department of Mechanical Engineering and Materials Science
Yale University
9 Hillhouse Avenue, New Haven, CT 06511, USA

Z. J. Patterson, C. Majidi
Department of Mechanical Engineering
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA
E-mail: cmajidi@andrew.cmu.edu

A. P. Sabelhaus
Department of Mechanical Engineering
Boston University
110 Cummington Mall, Boston, MA 02215, USA

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202200163.

DOI: 10.1002/aisy.202200163

W. Huang
School of Mechanical Engineering
Southeast University
Nanjing 211189, China

W. Huang
Jiangsu Engineering Research Center of Aerospace Machinery
Southeast University
Nanjing 211189, China

K. Chin, C. Majidi
Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA

Z. Ren
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
50 Vassar Street, Cambridge, MA 02139, USA

M. K. Jawed
Department of Mechanical Engineering and Materials Science
University of California, Los Angeles
420 Westwood Plaza, Los Angeles, CA 90095, USA
E-mail: khalidjm@seas.ucla.edu

organisms like the octopus,[9] jellyfish,[10] or sea stars[11] that have little or no dependency on stiff or inextensible materials. However, as engineers attempt to create bio-inspired soft robots that can truly approach the robust mobility and versatility of natural organisms, they are impeded by key bottlenecks in hardware design,[12] modeling,[13] and motion planning and control.[14]

Progress in creating untethered soft robots capable of controlled motion along a prescribed path greatly benefits from a framework that tightly integrates hardware design and mechanics with motion planning and control. While early efforts in soft robotics largely looked at tethered systems that required connections to a compressed air source for pneumatic actuation[15] or a high voltage power supply for electrostatic actuation,[16,17] recent efforts have increasingly shifted to fully untethered systems powered using miniaturized on-board electronics.[18–22] Shape memory alloys (SMA) like nickel–titanium have been especially promising for creating untethered soft robots because of their high work density and compatibility with miniature lithium polymer (LiPo) batteries and microcontrollers. Such robots can be engineered for locomotion speeds approaching 1 body length per second (blps),[23] far exceeding that of untethered pneumatic soft robots.[24] Moreover, SMA-powered robots have also been shown to move and interact with their environment in ways that can be accurately captured using soft robot physics engines based on discrete differential geometry (DDG).[25] These DDG-based simulation tools, which typically utilize the method of discrete elastic rods (DER), fill a niche in the growing landscape of soft robot simulation tools[13] as a middle ground between more physically realistic but computationally intensive FEA-based models[26–28] and simpler but less realistic traditional robotics and mechanics models.[29–31] They are capable of rapid, "faster than realtime" modeling with a single-thread microprocessor in which the computational runtime is faster than the wall-clock time of the soft robot motion that is being simulated. However, despite such functionality, studies applying DER to soft robotics have yet to demonstrate its usefulness for robot design or motion planning.[32]

Here, we address this shortcoming by introducing a framework for design and closed-loop trajectory following a fully untethered soft robot. This framework allows offline robot and gait design with DER and online motion planning of an SMA-based soft robot with large-scale data collected from simulation. Others have created various solutions in model-based simulation for design[30,33–39] and planning/control,[27,40–44] and here we show that DER-based methods in this work are capable of achieving similar functionality. To highlight the ability to accurately model not only the soft robot's internal mechanics but also its environmental interactions, we have focused on a swimming soft robot that is governed through a combination of SMA-powered limb motion and fluid–structure interaction. As shown in **Figure 1**a, the robot is composed of four limbs and a central body that contains the onboard batteries, microcontroller, and circuitry. The limbs are configured in front and rear pairs to allow for fast forward locomotion (Figure 1b) or turning using a frog-like swimming gait (Figure 1c). Limb motion is achieved by supplying electrical current to an embedded nitinol wire that undergoes a reversible shape memory transition that causes the limb to rapidly kick out. To facilitate the shape memory response of the nitinol, a liquid metal (LM) alloy is embedded in the surrounding elastomer to improve the flow of heat from the nitinol wire to the aquatic environment.

To demonstrate the framework, we simulate across a range of parameters and gaits to select the right combination for the frog-inspired soft robot, generate a motion library using the calibrated simulation, and perform an online path following using a receding horizon planner. DER has previously been shown to simulate soft robotic systems accurately, including for legged locomotion[45] and drag-based rowing in the water.[32] Here, we build on those foundations by showing that it can simulate accurately across a range of parameters, including robot limb design (e.g., LM volume fraction) and gait (phase, frequency), and that it can be used as a design tool to identify the best combination of those parameters. Finally, we show that it can be used to quickly generate a large and accurate data library that can be used in online motion planning.

2. Results

2.1. System Overview

Here, we present information about our experimental system and simulation. At a high level, our experimental system consists

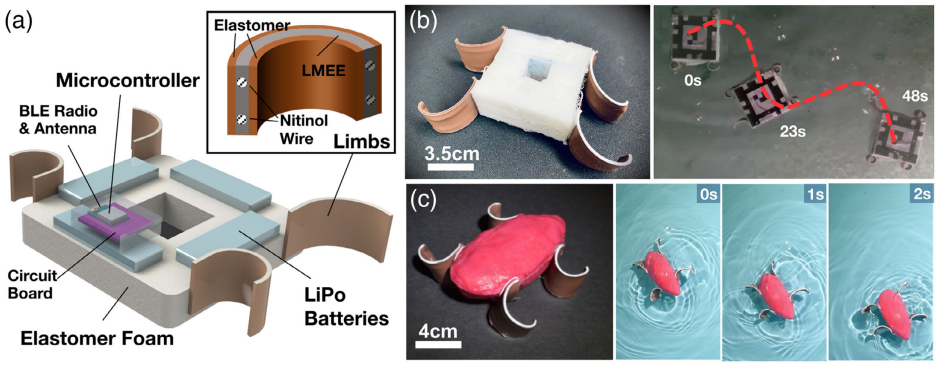

Figure 1. Design and geometry of the untethered frog-inspired soft robot. a) Components of robotic system. b) Square-shaped robot for examining motion along a curvilinear path. c) Stream-lined robot for >1 blps swimming along a straight line. The square-shaped robot is used in the simulation pipeline so that 2D simulations could be performed. The stream-lined robot is intended to show the potential performance of such swimming robots with a design that minimizes drag.

of an untethered robot that wirelessly communicates with the central computer which handles vision, planning, and logging. The simulation is a discretized dynamic model of the soft robot incorporating internal elastic stretching and bending forces along with externally applied fluid forces (e.g., drag).

2.1.1. Experimental Testbed

The untethered frog-inspired soft robot is composed of four SMA-driven soft actuators, four LiPo batteries, and an onboard PCB. (Figure 1a). The actuators are composed of stretched and unstretched layers of thermally conductive silicone that are bonded around an interface layer of LM-elastomer composite containing an SMA wire loop. The LM-elastomer is composed of microscale droplets of eutectic gallium–indium (EGaIn) LM alloy that serve to enhance the thermal conductivity[46] of the interface layer. The design is based on.[47] The amount of EGaIn used in the interface layer decreases the cooling time, increases actuator speed, and increases the mass.[48] The power and control electronics are housed inside of the foam bodies and sealed from water using silicone. Instructions are relayed to the robot's microcontroller via Bluetooth low energy (BLE) from a remote computer and microcontroller. The body of the robot has a hollow square shape, which is selected so that the body of the robot can be taken as four elastic rods with high stiffness connected together (Figure 1b). The robot limbs are attached to the body with velcro, resulting in a modular robot design so that each limb can be replaced within seconds. This enables the exploration of limbs with various actuation bandwidths without changing the robot design.

To demonstrate the versatility of this design, we also implement a more streamlined version of the robot that is capable of swimming at >1 blps speed (Figure 1c; Video S1, Supporting Information). Narrowing the central body of the untethered robot improves its hydrodynamics and allows for less drag during forward swimming. Although this allows for faster forward swimming, the narrow profile reduces the ability of the robot to make in-place turns. This is due to the reduced moment arm and in-plane torque that can be induced during differential limb actuation. For this reason, the remainder of this study will largely focus on the more maneuverable square-shaped robot. Nonetheless, the streamlined robot is presented here to demonstrate the potential capability for faster swimming speed that is possible within this design framework.

2.1.2. DER Simulation Tool

To simulate this robot, we utilize the DER algorithm for modeling structures with slender elastic elements.[49] Starting from the discrete representation of elastic energies, we formulate equations of motion at each node and solve in a backward Euler approach to update the configuration of the robot (i.e., position of the nodes) in time.

Similar to the DDG-based method presented in Refs. [25,32], this numerical framework starts with a discrete representation of the robot. The soft robot is treated as a collection of discrete elastic rods,[49] shown schematically in **Figure 2**a. The rods are comprised of N nodes, located at $x_i = [x_i, y_i]^T$ (with $i = 0, \ldots, N-1$), along the centerline. The limbs and the body of the robot are represented by rods with different densities. In the discrete setting in Figure 2a, the robot is represented by a lumped mass at each node and associated elastic stretching and bending energies—reminiscent of a mass-spring system. Since the motion of the robot remains in 2d, we do not include a twisting energy of the rod, although this can be readily integrated into our framework.

The rod segment between two consecutive nodes is an edge that can stretch as the robot deforms—analogous to a linear spring. The turning angle ϕ_i (Figure 2a) at node x_i between two consecutive edges can change—similar to a torsional spring. The elastic energy from the strains in the robot can be represented by the linear sum of two components: 1) stretching energy of each edge E_i^s and 2) bending energy E_i^b associated with variation in the turning angle ϕ_i at the nodes. The elastic stretching and bending forces acting on a node x_i can be obtained from the negative gradient of the elastic energies. The external forces acting on a node x_i are $f_i^{ext} = f_i^d + f_i^a$, where f_i^d is the damping force

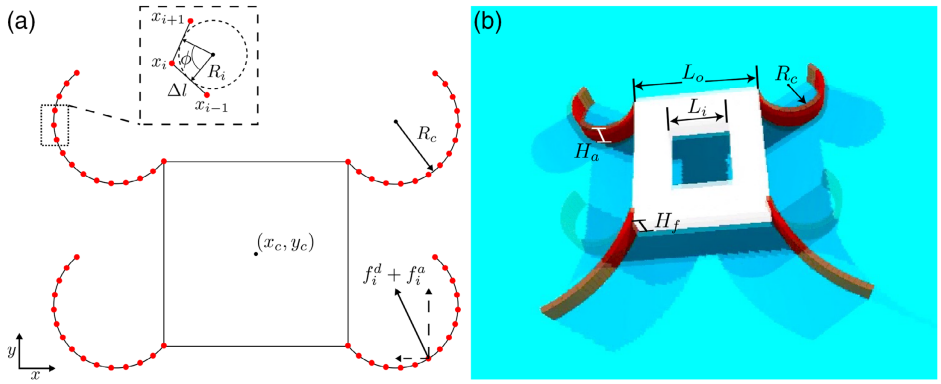

Figure 2. Robot simulation. a) Discretization of the robot with geometric properties and forces. R_c is the radius of curvature of the limb, (x_c, y_c) is the robot's center of mass position, and f_j^d and f_j^a are the drag and virtual mass forces on the jth node respectively. In the inset are properties of the discretized rod geometry, notably the turning angle, ϕ, and the rod length, Δl which is later used to calculate bending energy. x_{i-1}, x_i, and x_{i+1} are the labels of successive nodes. b) Rendering in simulation and geometry of the frog-inspired soft robot. $L_o = 90$ mm, $L_i = 35$ mm, $H_f = 35$ mm, $H_a = 24$ mm.

from fluid experienced by the soft limbs and body and f_i^a is the added-mass force from periodically accelerating the surrounding fluid (Figure 2a). These forces are then used to formulate equations of motion, which are implicitly solved for the next time step. See Materials and Methods for an in-depth coverage of the procedure.

In the following sections, we calibrate the parameters of the simulation and compare the performance of the simulation to experiments on hardware. We then perform a design study in which we vary the material properties and gait parameters of the robot in simulation to find fast and low cost of transport combinations. Finally, we use data collected from the simulation to produce a closed loop planner that enables the robot to follow curvilinear paths.

2.2. Soft Limb Actuators

We fabricated SMA actuators with 0–50% by volume of EGaIn droplets within the interface layer that contains the embedded nitinol wires. Referring to **Figure 3**a, the EGaIn droplets form an LM-embedded elastomer (LMEE) composite that is sandwiched between outer layers of elastomer. The purpose of the EGaIn droplets is to enhance the thermal conductivity of the interface layer and facilitate the flow of heat away from the wire as it cools at the end of each actuation cycle. To find the appropriate EGaIn volume fraction, we synthesized interface layers with volumes in 10% increments and characterized the actuated and unactuated curvature in water at various frequencies. We followed a similar approach to what we previously presented in Ref. [48]. The curvature of the actuator (κ_n) is determined by assuming constant curvature and fitting the actuator shape to a circular arc. Four samples at each EGaIn volume ratio are used for characterization. The actuators are cyclically actuated until steady state and mean curvature values are then estimated from recorded video.

Figure 3b presents the change in curvature as a function of activation frequency for various concentrations of EGaIn within the interface layer. In general, we don't see statistically significant influence of EGaIn volume fraction on the curvature change. However, we do see that actuators with greater amounts of EGaIn can be activated at higher frequencies due to the improved thermal conductivity and speed of nitinol cooling at the end of each actuation cycle. Faster cooling means that the nitinol wire can be electrically stimulated with higher frequency without degradation of the temperature-controlled shape memory response.

Figure 3c,d compares the forward locomotion speed of robots with 10 and 30 vol% EGaIn and operated with 0.5 phase offset at 0.4 and 1.3 Hz, respectively. As demonstrated, the higher concentration of EGaIn allows for higher frequency actuation and significantly faster swimming. In particular, note the difference in the time makers for the 10 and 30 vol% screenshots. Moreover, we observe good qualitative agreement between the experimental measurements and the computational predictions obtained using DER. Detailed results for the quantitative comparison between the experiment and simulation are presented in the following subsection.

2.3. Simulation Calibration and Validation

To calibrate the simulation, we test the robot between 0.4 and 1.1 Hz with limbs that are composed of 10 vol% EGaIn in the interface layer and recorded the locomotion when it reaches a steady speed. These results are used to calibrate the hydrodynamic simulation parameters C_{ln}, C_{bn}, C_{lt}, C_{bt}, and C_a, representing the drag coefficients along the normal direction for the limbs and body, the drag coefficients along the tangent direction for the limbs and body, and the added-mass coefficient, respectively. **Figure 4**a presents the locomotion speed as a function of actuation frequencies for experiments and simulations.

To further compare the experiments and simulations and investigate the influence of the timing of limbs on swimming speed, we vary the phase offset between the actuation of front limbs and rear limbs, which represent the ratio of the time

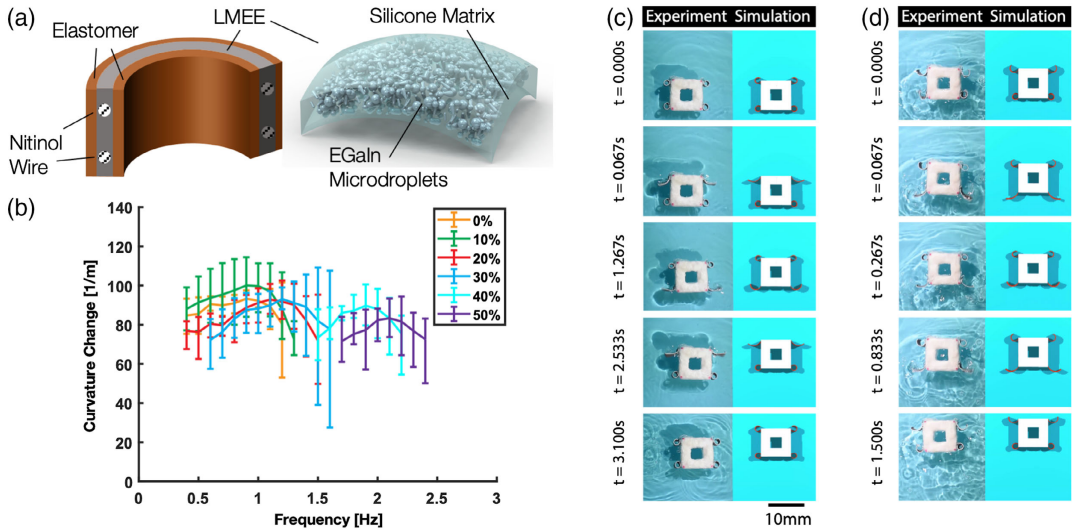

Figure 3. a) Schematic of soft robot limb and LM-embedded elastomer (LMEE) interface layer. b) Curvature change as a function of actuation frequency for limbs fabricated with various liquid metal volume ratios. Changes smaller than 70 1/m are not included. c) Experimental and simulation results for a robot with 10 vol% EGaIn limbs. d) Results for forward locomotion of robot with 30 vol% EGaIn.

difference between the actuation start time of front and rear limbs over total time in a cycle. As presented in Figure 4b,c, the experiment and simulation results have a good agreement on the speed at both 0.4 and 0.8 Hz for various phase offsets. At low actuation frequency (e.g., 0.4 Hz), the influence of phase offset on speed is trivial since there is little hydrodynamic interaction between the front and rear limbs during actuation (Figure 4b). As actuation frequency increases, the trend of speed as a function of phase offset becomes different between the experiment and simulation. The locomotion speed in the experiment is lower than that in simulation at phase offsets 0, 0.1, 0.2, and 0.8. This could be attributed to the instability caused by unmodeled hydrodynamic interactions leading to a sudden center of mass shift that triggers movement in the pitch direction. In the experiment, the speed is maximum at 0.5 phase offset as actuation frequency increases. From our observations, these trials had the least out-of-plane pitching. With the fitted hydrodynamic parameters acquired from the robot with limbs composed of 10% EGaIn, we test the robot with limbs composed of 40% EGaIn at frequencies between 1.5 and 1.9 Hz for validation. As shown in Figure 4d, there is a good agreement between experiments and simulations for both locomotion profile (Figure S1, Supporting Information) and speed, demonstrating that the simulation can successfully extrapolate to untested designs.

2.4. Parameter Sweep

Following the experimental validation, we use the DER simulation to find the design and control parameters that lead to the fastest locomotion and the parameters that lead to the most efficient locomotion (calculated by the cost of transport, or COT). The simulation is run at all combinations of EGaIn ratio, actuation frequencies, and phase offset. The top two candidates given by the numerical framework for the fastest locomotion are the robots built with limbs composed of 30% EGaIn and run at 1.3 Hz (4.7 cm s^{-1}) and 1.4 Hz (4.8 cm s^{-1}) with 0.5 phase offset. The design and control parameters that lead to the most efficient locomotion are the robot with limbs composed of 10% EGaIn and runs at 0.4 Hz (COT = 158) and 0.5 Hz (COT = 165) with 0.5 phase offset.

With the guidelines on design and control parameters provided by the simulation, we run the experiments accordingly and the results for the fastest and most efficient locomotion are presented in Figure 4e,f respectively. There is reasonable agreement between the experiment and simulation results for both fast and efficient locomotion cases. The simulation overpredicts the speed of the robot at low actuation frequency due to the planar assumption in the simulation being violated by out-of-plane pitching in practice. At high frequency, the simulation underestimates the speed, possibly due to beneficial vorticities ignored in the simplified hydrodynamic model used in the numerical framework. The snapshots of the frog robot with limbs composed of 10% and 30% EGaIn operated at 0.5 and 1.3 Hz, respectively, with 0.5 phase offset are shown in Figure 3c,d.

2.5. Motion Planning Framework

After validating the simulation and selecting the robot design and gait parameters, we develop a motion planning framework

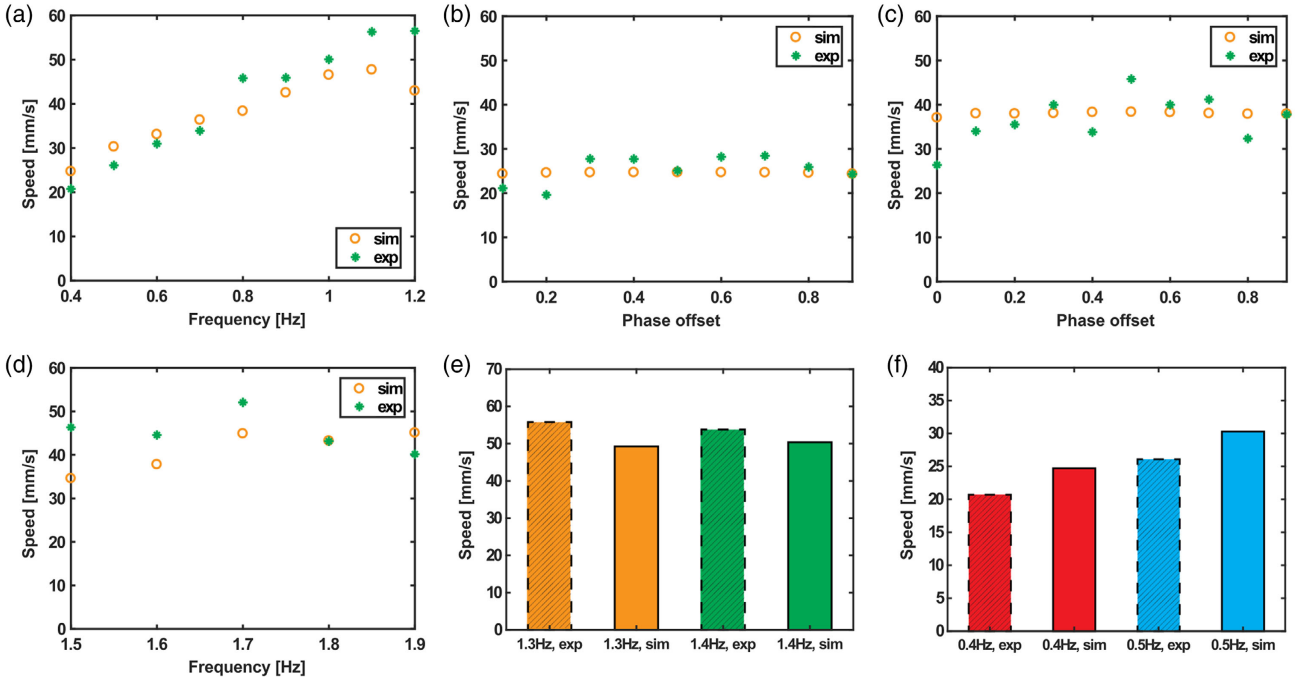

Figure 4. a) Robot speed as a function of actuation frequencies for experiments (green star) and simulation (orange circle) with 10 vol% EGaIn actuators. Comparison of experiment and simulation for a robot with 10 vol% EGaIn as a function of phase lag for b) 0.4 and c) 0.8 Hz actuation frequencies. d) Comparison of experiment and simulation for forward locomotion speed of robot with 40 vol% EGaIn. e) Experiment vs. simulation for robot optimized for the fastest speed. f) Experiment vs simulation for robot optimized for most "efficient" locomotion.

that enables a closed-loop path following along arbitrary shapes in 2D space. While DER is capable of faster than real-time simulations,[25] the dynamics implementation is not fast enough to run inside an online optimization loop. Instead, we leverage the capability to run many DER simulations in a relatively short period of time to collect a large set of data capturing the robot's dynamics resulting from a set of nine actions (or motion primitives). This data library can then be used as the basis for a closed loop motion planner—similar in form to explicit model predictive control—to choose the best available action according to some cost function associated with the projected future state.

2.5.1. Online Planner for Path Following

The first step in preparing our motion planning framework is collecting our data library. The data library consists of a large number of transition models that give the future state of the robot for some initial state and some action. To produce such a library in a way that allows fast operation at run-time, we reduce the space of needed samples by constraining the number of actions that the robot can perform and the number of initial states of the simulations. We manually choose nine actions, i.e., SMA actuation sequences, corresponding to behaviors like "go forward," "turn and go forward", "turn in place," and "no action." Next, we take advantage of the invariance of the robot's dynamics to its position and orientation and only sample over distributions of initial linear and angular velocity states. The resulting set of simulations consists of 8503 initial velocity states (giving 76 527 total transitions). These simulations are performed offline and the data is collected and processed into a large matrix such that the transitions can be applied at run-time by efficient indexing operations.

To use the data library in a path following feedback control policy, we implemented a search-based planning algorithm over a tree generated by branching over primitives, allowing the robot to choose the best available action given its current state. The implemented algorithm is a receding horizon planner using the nearest neighbor in the data library to form predictions about future states. Paths are specified by generating a series of waypoints from a parametric curve in world-space. To track the path, the current state is first captured from a camera that tracks the robot using an Apriltag.[50,51] Then, this state is transformed from the world coordinate frame into the body frame to compare it to the states in the data library. We find the closest state in the library and apply the projected transitions for each of our nine actions to the current state and transform back to world coordinates to get our projected next state. We recursively repeat the above until we reach the specified depth of our motion planning tree. Finally, we compare the costs associated with our projected state and choose the action leading to the lowest cost branch on the tree.

The cost function used to evaluate the best action has three components (as in[52])—distance from the nearest waypoint, difference between the robot angle and the angle of the tangent of the path at the nearest waypoint, and the progression along the trajectory associated with the nearest waypoint: $\text{cost} = w_a \times \text{dist} + w_b \times \text{ang} + w_c \times \text{prog}$, where the w's represent weights for each element of the cost. This cost function induces the robot to stay near the path, maintain the correct heading, and continue forward along the path. See Experimental Section for in-depth coverage of the data collection procedure, the receding horizon planning algorithm, and the cost function.

We tested the motion planning approach on several different types of paths including a straight line, a sinusoidal curve, and an ellipse. The convergence of the cost function for experiments with each of these paths is shown in Figure S3, Supporting Information. **Figure 5**a–c shows snapshots of the robots along each trajectory and Figure 5d–f shows plots of the x–y position of the robot compared to the waypoints of each of the paths. Figure 5g shows the distance of the robot to the nearest point on the discretized trajectory for each case. Videos of some of the experiments are included in Video S2, Supporting Information. From these results, we can see that the robot can successfully follow paths of varying complexity at fairly high speed (2.2–3.2 cm s^{-1}). While tracking is not perfect due to the high dimensional representation and low control bandwidth, the robot recovers robustly from deviations and is able to maintain the trajectory qualitatively towards the goal. In practice, we found that a depth of one (i.e. only considering the next step) for the motion planner resulted in the best performance.

3. Discussion

After demonstrating that DER could predict system dynamics with reasonable accuracy across varying design and gait parameters, we performed a parameter sweep to identify the fastest and most efficient sets of design parameters (EGaIn volume fraction) and gait parameters (phase, frequency). We showed that there is good agreement between the predicted speeds based on simulation and the resulting robots' speeds. Thus, DER is shown to be an effective tool for soft robot design. While the parameter space examined here is narrow, since the simulation is fast and easily automated it would be easy to sweep across a larger parameter space with more candidate robots and gaits.

We also demonstrate, for the first time, an online planning framework leveraging DER. Our resulting implementation compares favorably with recent simulation-driven soft robot trajectory optimization and open-loop control schemes,[27] enabling high-level tasks to be performed at relatively fast robot speeds (3.2 cm s^{-1} vs 0.75 cm s^{-1}) and control frequencies. Our planning approach, where we precompute a large library of trajectories based on motion primitives, is popular across many areas of robotics[53] including flying robots,[54] grasping robots,[55] autonomous vehicles,[56] and other systems with real-time execution requirements and constraints on run-time execution. At run time, we can then plan ahead efficiently by finding the nearest neighbor within our library to the current state and projecting forward based on the pre-simulated transition model using a receding horizon plan.[57] We experimented with interpolation but the nearest neighbor proved more performant. This library-based, tabular method has pros and cons versus solving the full dynamics. By representing the dynamics as essentially a large array of transitions, our function evaluations are far more efficient than solving differential equations and we can plan deeper trees. We could also straightforwardly implement the planner on low-level hardware to enable cheap mobile autonomy

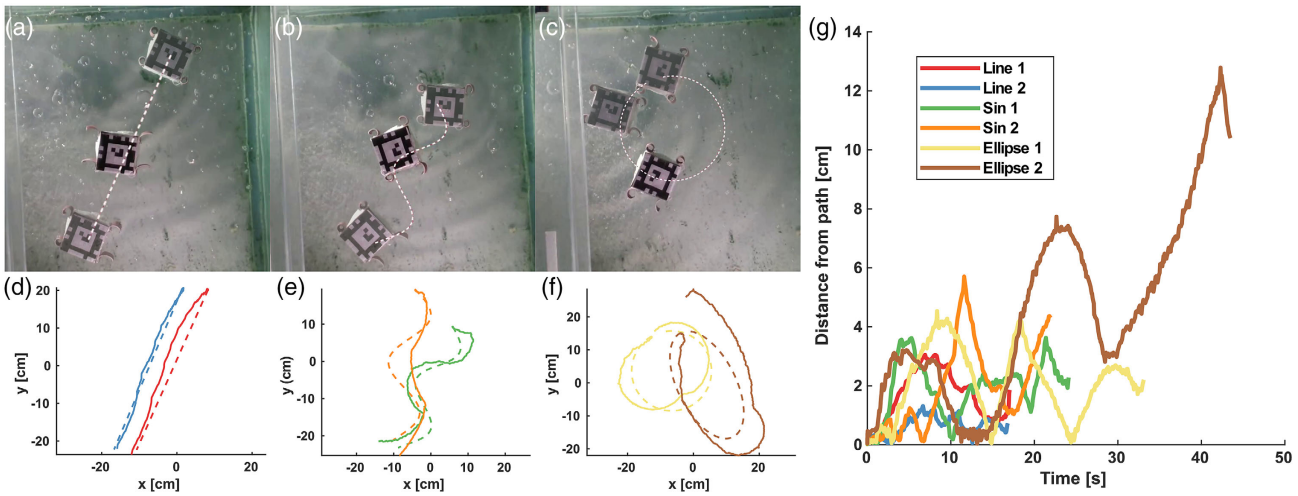

Figure 5. Stills of the robot at the beginning, middle, and end of: a a) straight line path, b) sinusoidal path, and c) ellipsoid path. 2D plots showing (x,y) position of the robot and the desired path for two trials of each type of parametric path including: d) straight line, e) sinusoid, f) ellipsoid. g) Distance from the nearest point on the path for each trial color-coded to correspond with the plots in (d)–(f). Therefore, Line 1 in g corresponds to the red lines in d, Sin 1 refers to the green lines in e, and so on.

and efficient collision avoidance.[58] In principle, as long as an appropriate sensor is incorporated into the robot for gathering position data, this approach to control can therefore run entirely on low-level hardware. Unfortunately, however, if actuator degradation significantly changes the performance of the hardware, new simulations would have to be run on a computer to obtain a new motion library.

One drawback of the approach is that we are constrained to a predefined set of motion primitives and cannot optimize at runtime over the full space of possible actuations. This produces trajectories that are predictably sub-optimal. Also, although we implement a receding horizon planner capable of fast performance at large depth, in practice we find that the best performance occurs for a tree depth of one. One likely explanation is a distribution mismatch between the library and the states produced by performing sequential action on the hardware. This would mean that transition prediction error is propagated at each level and degrades performance at larger search depth. This problem could be remedied by sampling more of the state space during library generation. To assess these issues, in future work we will compare the performance across various sizes of the data library and compare against a planner that queries the DER simulation itself for transitions in the loop. Regardless, the effectiveness of our approach on hardware demonstrates that DER can be readily adapted into standard robot motion planning frameworks.

Regarding the hardware itself, the robots introduced are quite fast compared to existing untethered soft robots.[7] This is enabled by the EGaIn-embedded design of the actuators that allows them to achieve higher frequency performance than is usually expected from SMA-based actuators. Unfortunately, due to the nature of thermal actuation using resistive Joule heating, the inefficiency of SMA-based actuators (usually less than 1%) is much more difficult to overcome.[59] The result of this is that the robots presented in this article are inefficient. The cost of transport varies depending on the specific robot design, configuration, and gait, but for the parameters discussed in this paper, observed values range from about 1000–10 000. While this is poor from an efficiency standpoint, the buoyancy force allows some degree of mitigation since relatively large batteries can be used as long as their mass is sufficiently balanced by buoyant material. Thus, qualitatively, while performing the experiments discussed in the article, we observed reasonable run-times, operating the robot continuously for about 20 min to 2 h depending on the gait chosen.

One shortcoming of the presented work is that the robots and simulators are two-dimensional. Simulating 3D robots with DER is currently possible (see e.g.,[60]), and thus the framework presented in this paper for design and control can be applied in principle to three-dimensional swimming robots. The two primary issues that may impede ready application are speed and accuracy. Regarding speed, the move to 3D will incur a computational cost due to the need to keep track of additional coordinates, but this cost should not make the simulations impractical to perform and Du et. al.[60] have shown that 3D DER simulations can be practical for robotics. Regarding accuracy, we cannot say without validation that 3D simulations will have an adequate agreement with reality, but previous work such as Ref.[61] gives confidence that DER can effectively simulate 3D nonlinear fluid–structure interactions.

It may even be the case that extending to three dimensions would increase accuracy. It was observed, for example, in Figure 4d, that the agreement between simulation and experiment tends to deviate more for certain frequencies and phases of the gait, which we suspect is due to nonplanar hydrodynamics, which could be accounted for with a 3D model. Since this would reduce the speed of the simulator, decisions about the fidelity of the simulation could be made on a case-by-case basis depending on the application and its needs for speed versus fidelity. Additionally, as computing power increases and scientific

progress is made in the realm of fluid dynamics, the incorporation of more complex fluid dynamics may become feasible in the future.

While our approaches to design and control were effective for the class of robots introduced, the effectiveness is uncertain for more complex soft robots that interact frequently with the environment.[9,62] While it has been previously shown that such robots can operate with even simpler high-level planning over a small set of motion primitives,[11] it would be useful to use DER to compare performance across design parameters, to optimize gaits offline, and to enable more complex sequences of actions. Recent work shows that DER can accurately simulate complex contact scenarios.[63] While the speeds achieved in that work are impressive, future work is necessary to determine if they are fast enough to be practical in our framework.

4. Conclusion

This work introduces a new class of frog-like soft robots that can achieve fast locomotion speeds. We also show multiple new functionalities of the DER framework that make it an effective modeling tool for simulating and controlling robots with deformable components. In utilizing DER, we first calibrate the simulation and show that it provides realistic results across a range of parameters, including robot design and gait. Then, we performed a parameter sweep to find the fastest and most efficient sets in the space. Next, we use the simulation to generate an online planner that can be used for trajectory tracking with one of our selected robot designs. Lastly, we implement this motion planning scheme on an experimental testbed and demonstrate the ability of a fully untethered, frog-like soft robot to swim along various pre-defined paths within a water tank.

5. Experimental Section

Robot Electronics: The robot contains a Laird BL652 chip which is based on the nRF52832 microcontroller. The Bluetooth-enabled chip communicates via BLE with an offboard nRF52832 dev-kit. This offboard controller relays instructions transmitted from a computer through a UART connection. The SMA wire actuators need high power pulses to actuate, and therefore require the use of MOSFETs (AO3416, Alpha & Omega Semiconductor Inc.). Finally, for power, the robot uses four 11.1 V, 300 mAh, 45/75C drone batteries (BETAFPV) to ensure sufficient swimming time (≈257 cycles) for characterization. Each limb is powered by an individual battery. The body of the robot has a hollow square shape with dimensions as $L_o = 90\,\text{mm}$, $L_i = 35\,\text{mm}$, $H_f = 35\,\text{mm}$, $H_a = 24\,\text{mm}$ (Figure 2b). A battery is placed on each edge individually to balance the weight. Foam (Soma Foama 15, Smooth-On) is used to cancel the weight of the batteries to ensure buoyancy of the robot and silicone (Dragonskin 30, Smooth-On) is used to seal the electronics from water.

Energy, Forces, and Equations of Motion: For the following sections, we denote vectors as lowercase bold and italicized letters (e.g., $\boldsymbol{v}$) and matrices as bold capital letters (e.g., $\mathbf{M}$). The robot is discretized as a series of elastic rods. The rods are represented by N nodes each with a position $x_i = [x, y]$, forming the state $x = [x_0, x_1, \ldots, x_{N-1}]$. Each node is associated with a lumped mass and elastic energies. The rod segment between two nodes is, therefore, a deformable edge. In all of our simulations, the length of each segment is $\Delta l \approx 4$ mm resulting in $N_{\text{total}} = 128$ nodes. The elastic energies are as follows. The discrete stretching energy at the edge connecting x_i and x_{i+1} is $E_i^s = EA\varepsilon^2\Delta l/2$, where EA is the stretching stiffness and $\varepsilon = \|x_{i+1} - x_i\|/\Delta l - 1$ is the axial stretch. The discrete bending energy is $E_i^b = EI(\kappa_i - \kappa_i^0)^2\Delta l/2$, where EI is the bending stiffness, $\kappa_i = 2\tan(\phi_i/2)/\Delta l$ is the curvature, κ_i^0 is the natural curvature (i.e., curvature evaluated in undeformed configuration), and ϕ_i is the turning angle between two consecutive edges. The total elastic energy of the robot can be obtained simply by summing over all the edges, i.e., $E^s = \sum_i E_i^s$, and, similarly, the total bending energy is $E^b = \sum_i E_i^b$. In both experiments and simulations, the structure is nearly inextensible and the prominent mode of deformation is bending. $E_0 = 3.0\,\text{MPa}$ and $E_{\max} = 8.04\,\text{MPa}$ reported in[25] are used in the simulation. Note that each joint node at the corners of the robot is connected to three different nodes and, therefore, is associated with three turning angles and two discrete bending energies.

To find the elastic stretching and bending forces acting on a node $\mathbf{x}_i$ we take the gradient of the energies, i.e., $f_i^s = -[\partial E^s/\partial x_i, \partial E^s/\partial y_i]^T$ and $f_i^b = -[\partial E^b/\partial x_i, \partial E^b/\partial y_i]^T$, respectively. So the internal force acting on a node x_i is $f_i^{int} = f_i^s + f_i^b$. An implicit treatment of the elastic forces requires the calculation of the $2N \times 2N$ Hessian matrix of the elastic energies. Other than the joint nodes, a node x_i is only coupled with the adjacent nodes x_{i-1} and x_{i+1} in the discrete energy formulation. This results in a banded Hessian matrix with 6×6 blocks of nonzero entries along the diagonal. The only off-diagonal nonzero entries correspond to the four joint nodes. A symbolic computing environment (Maple) to obtain the expressions for the Hessian matrix and covert them into our C^{++} implementation is used. The sparse nature of the Hessian leads to an outstanding computational efficiency. The external forces acting on a node x_i are $f_i^{\text{ext}} = f_i^d + f_i^a$, where f_i^d is the damping force from fluid experienced by the soft limbs and body, and f_i^a is the added-mass force from periodically accelerating the surrounding fluid. The drag force acting on a node x_i is given by

$$f_i^{\text{d}} = -\frac{1}{2}\rho_{\text{f}}\mathbf{C}_{\text{d}}D\|v_i\|v_i\Delta l_i \tag{1}$$

where ρ_{f} is the density of the fluid medium, D is the diameter of the rod, and v_i is the relative velocity of the ith node to the fluid. $\mathbf{C_d}$ is the drag coefficient matrix

$$\mathbf{C}_{\text{d}} = \begin{bmatrix} C_{\text{t}} & 0 \\ 0 & C_{\text{n}} \end{bmatrix} \tag{2}$$

where C_{t} and C_{n} represent the drag coefficient along normal and tangent directions respectively. The added mass force is associated with the volume of the surrounding fluid that moves with the robot and it is represented as

$$f_i^{\text{a}} = -C_{\text{a}}\rho_{\text{f}}\pi\left(\frac{D}{2}\right)^2\Delta l_i\frac{\partial v_i}{\partial t} \tag{3}$$

where C_{a} is the added-mass coefficient and ρ_{f} is the density of the fluid. Instead of taking the added mass force as an external force, we include it into the mass matrix and form a revised mass matrix

$$\mathbf{M}_i = (\rho_{\text{r}} + C_{\text{a}}\rho_{\text{f}})V_i\mathbf{I} \tag{4}$$

where $\mathbf{I}$ is the identity matrix and V_i is the discretized volume at node i calculated by averaging the volume of the segments of the actuator represented by adjacent edges.

The equations of motion at the ith node are

$$\begin{bmatrix} \ddot{x}_i \\ \ddot{y}_i \end{bmatrix} = \begin{bmatrix} \frac{1}{m_i} & 0 \\ 0 & \frac{1}{m_i} \end{bmatrix}(f_i^{\text{s}} + f_i^{\text{b}} + f_i^{\text{d}}) \tag{5}$$

where $\dot{()}$ represents derivative with respect to time and m_i is the revised lumped mass at node i that incorporates the added-mass from the fluid. We solve the $2N$ equation of motions and update the DOF vector x and its velocity $v = \dot{x}$ from time step t_k to $t_{k+1} = t_k + h$ (h is the time step size) based on the following force balance

$$f \equiv \mathbf{M}\frac{x(t_{k+1}) - x(t_k) - h\nu(t_k)}{h^2} - f_s - f_b - f_d \tag{6}$$

where $\mathbf{M}$ is the revised mass matrix, f_s is the stretching force, f_b is the bending force, f_d is the hydrodynamic damping force, and h is the time step size. The Newton–Raphson method is used to solve this set of nonlinear equations of motion. At each time step t_{k+1}, we start a new solution guess on the basis of the previous state, and optimize by gradient descent until a desired tolerance is achieved

$$x^{n+1}(t_{k+1}) = x^n(t_{k+1}) - \mathbf{J}^n \setminus f^n \tag{7}$$

where $\mathbf{J}$ is the Jacobian matrix

$$\mathbf{J} \equiv \frac{\partial f}{\partial x} = \frac{1}{h^2}\mathbf{M} + \frac{\partial^2 \mathbf{E}}{\partial x \partial x} - \frac{\partial f_d}{\partial x} \tag{8}$$

Actuator Characterization: Four samples of SMA actuators at each volume ratio from 0% to 50% EGaIn in 10% increments are characterized by measuring the actuated and unactuated curvature (κ_n) in water by fitting a circular arc to a captured image of the actuator shape (as in Ref. [32]). The experimental setup is as follows. Actuators are fixed horizontally at the bottom of a tank filled with water and filmed by a GoPro 5 that is submerged in water from the top. The temperature of the water is controlled at 19 °C. The actuation time is $t_a = 70$ ms for all actuators and the cooling time varies depending on the actuation frequency. The actuation profiles in a complete cycle are recorded after 100 actuation cycles are reached to guarantee that the actuators reach a steady state. The mean curvature value from four samples at each EGaIn volume ratio as a function of time within a single cycle are fit into piece-wise functions that have the logistic form of

$$\overline{\kappa}(t) = \begin{cases} \frac{C_{1a}}{(1+e^{-C_{2a}(t-C_{3a})})} + C_{4a} & \text{when } t < t_0 \\ \frac{C_{1d}}{(1+e^{-C_{2d}(t-C_{3d})})} + C_{4d} & \text{when } t > t_0 \end{cases} \tag{9}$$

where $C_{1a} - C_{4a}$ and $C_{1d} - C_{4d}$ represent the fitting parameters for the actuation and deactuation profiles, respectively. The piece-wise functions are used in the simulation to generate the actuation and deactuation of the limbs. The actuation profiles and fit functions are plotted in Figure S4, Supporting Information.

As the actuation frequencies increase, the curvature changes become inconsistent among the four samples. We select the frequencies with error bars that are smaller than 40 1 m^{-1} as inputs into the numerical framework for validation and optimization to ensure consistency between samples.

Hydrodynamic Characterization: Hydrodynamic parameters used in the numerical framework are the drag coefficients along the normal and tangent directions for the limbs (C_{ln}, C_{lt}) and body (C_{bn}, C_{bt}), and the added-mass coefficient (C_a), respectively. $C_{ln} = 1$ and $C_{lt} = 0.01$ are used as the drag coefficients for the limb[32] and C_a is simply set as its theoretical value $C_a = 1$.[64] $C_{bn} = 0.8$ is solved by minimizing $\sqrt{\sum_{i=f_{min}}^{f_{max}} (\nu_{sim_i} - \nu_{exp_i})^2 / n_f}$, for C_{bn} ranging between 0.8 and 2.0,[65] where i, f_{min}, f_{max}, and n_f represent the actuation frequency, minimum and maximum of the actuation frequency, and number of different frequencies for limbs with a fixed liquid metal volume ratio. ν_{sim} and ν_{exp} are the swimming speed of the robot in the simulation and experiment respectively. Drag coefficients along the tangent direction are set as 1/100 of the ones along the normal direction for slender structures.[66] The robot with limbs consisting of 10% liquid metal in volume is used for the hydrodynamic characterization. Ideally, hydrodynamic parameters should be determined experimentally.

Data Collection Using the DER Simulation: The first step in preparing our motion planning framework is collecting our data library. To simplify our robot's state space for planning, we use a rigid body approximation of the DER rod (see Figure S2A, Supporting Information), taken as the 6D vector $q(t_k) = \left[x_c(t_k), y_c(t_k), \theta(t_k), \dot{x}_c(t_k), \dot{y}_c(t_k), \dot{\theta}(t_k)\right]^T$. Here, $x_c(t_k) = [x_c(t_k), y_c(t_k)]^T$ is the DER's center of mass, weighted by the mass m_i at node i

$$x_c(t_k) = \frac{1}{M_T}\sum_{i=0}^{N-1} m_i x_i(t_k), \quad M_T = \sum_{i=0}^{N-1} m_i \tag{10}$$

The center of mass velocity $\dot{x}_c(t_k)$ is defined similarly.

However, since our rod is modeled as a system of particles, both a sense of "average rotation" and "average angular velocity" requires more thought. Here, we define an approximation using the nodal coordinates of the DER in its unactuated reference pose, x_i^{REF}, and the rod translated back to the origin as $x_i^O(t_k) = x_i(t_k) - x_c(t_k)$. We then define

$$\theta(t_k) = \frac{1}{N}\sum_{i=0}^{N-1} \cos^{-1}\left(\frac{x_i^O(t_k) \times x_i^{REF}}{||x_i^O(t_k)||_2 \, ||x_i^{REF}||_2}\right) \tag{11}$$

$$\dot{\theta}(t_k) = \frac{1}{M_T}\sum_{i=0}^{N-1} m_i \frac{x_i^O(t_k) \times \dot{x}_i(t_k)}{\|x_i^O(t_k)\|_2^2} = \frac{1}{M_T}\sum_{i=0}^{N-1} \frac{det\left[x_i^O(t_k), \dot{x}_i(t_k)\right]}{\|x_i^O(t_k)\|_2^2} \tag{12}$$

The aforementioned approximations are motivated by definitions of angular momentum. We note that there are more formal and theoretical approaches to defining the average angular velocity of a system of particles; however, this article focuses on deploying our simulation for practical uses. Future work will examine different state defintions and their impact on our planning procedure.

Ideally, our library would capture the robot's performance in response to any behavior from any initial conditions. However, to make the library manageable computationally, the set of behaviors and the set of initial conditions need to be reduced. For the set of behaviors, we choose nine actions based on our qualitative observations during simulation validation. While the effect of each on the robot's state varies with initial conditions, the actions generally correspond to behaviors like "go forward," "turn and go forward", "turn in place," and "no action" (Figure S2b, Supporting Information, depicts approximately the qualitative behavior of the action set). The actions are designed by simply specifying activation sequences of the SMA actuators. The actuation model is covered in Experimental Section. For each action, the front limbs, if any, are first actuated for a set activation time (70 ms throughout this work) and then the rear limbs, if any, are actuated for the same activation time. For a depiction of activation sequences and an example of one of the actions, see Figure S2c, Supporting Information.

Our planner framework uses the simulation environment to map transitions between time points in terms of this rigid body state q. To obtain these mappings, our simulations initialize the robot under different initial conditions, then simulate each of the nine actions from that initial condition. Since the effect of the action on the robot's state is invariant to the robot's position or orientation in the world frame, we only sample initial conditions in terms of velocities. Specifically, $q(0) = \left[0, 0, 0, \dot{x}_c(0), \dot{y}_c(0), \dot{\theta}(0)\right]$, where velocities are sampled from uniform distributions according to $\dot{x}_c(0) \in [-3, 3]$, $\dot{y}_c(0) \in [0, 4]$ cm s^{-1}, and $\dot{\theta}(0) \in [-30, 30]^\circ\ s^{-1}$. We do not sample negative y-velocities since we do not anticipate the robot swimming backward.

Initializing the DER rod requires mapping the "average" velocities back to individual nodes. Nodal velocities are taken as the sum of linear and angular contributions, $\dot{x}_i(0) = \dot{x}_c(0) + \dot{x}_i^\theta(0)$, where $\dot{x}_c(0) = [\dot{x}_c(0), \dot{y}_c(0)]^T$. The contribution of angular velocity is calculated as

$$\dot{x}_i^\theta(0) = \dot{\theta}(0)\, x_i^\perp(0) = \dot{\theta}(0)\, [-y_i(0)\ x_i(0)]^T. \tag{13}$$

The total number of simulations performed is 76 527 (8503 per action).

From here, we can specify an arbitrary planner frequency for up to five seconds and process the data from each simulation by taking the last timestamp t_f corresponding to the chosen frequency and packaging the data from that timestamp in the following vector of action, initial state, and final state $\left[a_j, 0, 0, 0, \dot{x}(t_0), \dot{y}(t_0), \dot{\theta}(t_0), x(t_f), y(t_f), \theta(t_f), \dot{x}(t_f), \dot{y}(t_f), \dot{\omega}(t_f)\right]^T$. At run-time, we use this data to generate the transition model for each initial condition

$$q_k(t_f) = F(q_k(t_0), a_j) = \left[\Delta x_k, \Delta y_k, \Delta\theta_k, \dot{x}_k, \dot{y}_k, \dot{\theta}_k\right]^T \tag{14}$$

where we overload notation by using k to now refer to the index of a particular simulation rather than a discretized timestamp, $\Delta x_k = x(t_f)$, and $\dot{x}_k = \dot{x}(t_0)$. These transitions representing the robot's physics are stored as a table that can be rapidly processed in a real-time algorithm. Figure S2d, Supporting Information, shows a set of 1000 random transitions from our library showing the initial position (always at the origin) and the final position (x_{t_f}, y_{t_f}) along with a rectangle oriented according to θ_{t_f}.

Control Algorithm and Software: The desired path waypoints are determined by parametrizing 2D curves by a single parameter, s, which we discretize as 100 evenly spaced points in $[0, 1]$. The straight line path is

$$p_{\text{line}} = [\ell s, 0]^T \tag{15}$$

where ℓ is a constant that sets the path length. The sinusoidal path is

$$p_{\sin} = \left[\ell s, \frac{\ell}{A}\sin(fs)\right]^T \tag{16}$$

where f is the frequency and A is an arbitrary constant representing the height of the sin curve relative to the length, ℓ. The ellipse path is

$$p_{\text{ellipse}} = \left[\frac{\ell}{2}\left(1 - \cos\left(2\pi\frac{s}{s_f}\right)\right), \frac{\ell}{2A}\sin\left(2\pi\frac{s}{s_f}\right)\right] \tag{17}$$

where A is again an arbitrary constant representing the ratio of height to length of the ellipse and s_f is the final value of the parameter s.

The algorithm for controlling the robot to follow a specified path is shown in Algorithm 1:

Algorithm 1. Robot Planner

1: $C \Leftarrow \infty$

2: $i \Leftarrow 1$

3: Get state $q(t) = \left[x(t), y(t), \theta(t), \dot{x}(t), \dot{y}(t), \dot{\theta}(t)\right]^T$ from camera.

4: Repeat

5: Transform from world to body coordinates by rotating $[\dot{x}(t), \dot{y}(t)]^T$ by $-\theta$ using the standard 2D rotation matrix.

6: For states in library $\{q_k\}, k = 0, \ldots, K-1$ calculate distance d_k and stack into vector d:

$d_k = \| \left[(\dot{x}(t) - \dot{x}_k)/\dot{x}(t), (\dot{y}(t) - \dot{y}_k)/\dot{y}(t), (\dot{\theta}(t) - \dot{\theta}_k)/\dot{\theta}(t)\right] \|_2$.

$d = [d_0, d_1, \ldots, d_{K-1}]$

7: $k_{min} = \arg\min_k d$

8: **for** $a_j \in \mathcal{A}$ (where a_j is an action and $\mathcal{A}$ is the set of all actions) **do**

9: Obtain transition prediction of system state after primitive execution period T, $q(t+T) = q_{next} = F(q_{k_{\min}}, a_j)$ from Equation (17).

10: Rotate the transitions to the world frame by θ using the standard rotation matrix $R(\theta)$ and apply to the current state by applying Equation (18).

11: $q(t+T) \Leftarrow q(t)$.

12: $i = i + 1$

13: **end for**

14: **until** $i > H$

15: Calculate cost function from Equation (19).

16: $a^* = \arg\min_j C$

17: Execute a^*

When predicting the next state of the robot based on a given transition, the following update is applied

$$\begin{bmatrix} x(t+T) \\ y(t+T) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} + R(\theta(t)) \begin{bmatrix} x_{\text{next}} \\ y_{\text{next}} \end{bmatrix}$$
$$\theta(t+T) = \theta(t) + \theta_{\text{next}}$$
$$\begin{bmatrix} \dot{x}(t+T) \\ \dot{y}(t+T) \end{bmatrix} = R(\theta(t)) \begin{bmatrix} \dot{x}_{\text{next}} \\ \dot{y}_{\text{next}} \end{bmatrix} \tag{18}$$
$$\dot{\theta}(t+T) = \dot{\theta}_{\text{next}}$$
$$q(t+T) = \left[x(t+T), y(t+T), \theta(t+T), \dot{x}(t+T), \dot{y}(t+T), \dot{\theta}(t+T)\right]^T$$

The cost function used to determine the best action is

$$C = w_a \times \min(\|x_c(t) - p(s_c)\|) + w_b \times \frac{x_c(t) \times p(s_c)}{\|x_c(t)\|\|p(s_c)\|} + w_c \times (1 - s_c) \tag{19}$$

where $s_c = \arg\min_s(\|x_c(t) - p(s)\|)$.

The cost function is intended to incentivize proper positioning on the path, proper angular heading and forward progression. The weights used in the experiments presented are $w_a = 500$, $w_b = 50$, and $w_c = 300$ for positioning, angular heading, and forward progression components respectively.

Data from the path following experiments showing the cost function and its components are shown in Figure S3, Supporting Information.

The control algorithm, AprilTag tracking, serial communications, and logging are all implemented in Python, using the multiprocessing package to concurrently perform all necessary tasks and shared queues for process communication and synchronization.

Code Availability

Code is available in the git repository https://github.com/softmachineslab/frog.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

X.H. and Z.J.P. contributed equally to this work. This work was supported by the Army Research Office under grant W911NF-16-1-0148 (Program Manager: Dr. Sam Stanton), Office of Naval Research under grant N00014-17-2063 (Program Manager: Dr. Tom McKenna), and National Oceanic Partnership Program under grant N00014-18-12843 (Program Manager: Reginald Beach). Researchers working on this project also received support through the following fellowship programs: National Science Foundation Graduate Research Fellowship (Kiyn Chin) and Intelligence Community Postdoctoral Research Fellowship, Oak Ridge Institute for Science and Education (Andrew Sabelhaus). Lastly, we acknowledge support from the Henry Samueli School of Engineering and Applied Science, University of California, Los Angeles.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Keywords

model-based control, motion planning, simulation, soft robotics

Received: June 13, 2022
Revised: July 26, 2022
Published online: September 23, 2022

[1] E. W. Hawkes, C. Majidi, M. T. Tolley, *Sci. Robot.* **2021**, *6*, eabg6049.
[2] R. Salazar, V. Fuentes, A. Abdelkefi, *Ocean Eng.* **2018**, *148* 75.
[3] S. Kim, C. Laschi, B. Trimmer, *Trends Biotechnol.* **2013**, *31*, 287.
[4] B. Mazzolai, C. Laschi, *Sci. Robot.* **2020**, *5*, eaba6893.
[5] M. Stoppa, A. Chiolerio, *Sensors* **2014**, *14*, 11957.
[6] C. Majidi, *Adv. Mater. Technol.* **2019**, *4*, 1800477.
[7] S. I. Rich, R. J. Wood, C. Majidi, *Nat. Electron.* **2018**, *1*, 102.
[8] R. K. Katzschmann, J. DelPreto, R. MacCurdy, D. Rus, *Sci. Robot.* **2018**, *3*, eaar3449.
[9] M. Cianchetti, M. Calisti, L. Margheri, M. Kuba, C. Laschi, *Bioinspir. Biomimetics* **2015**, *10*, 035003.
[10] S.-W. Yeom, I.-K. Oh, *Smart Mater. Struct.* **2009**, *18*, 085002.
[11] Z. J. Patterson, A. P. Sabelhaus, K. Chin, T. Hellebrekers, C. Majidi, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, IEEE, Piscataway, NJ **2020** pp. 8758–8764.
[12] C. Laschi, M. Cianchetti, *Front. Bioeng. Biotechnol.* **2014**, *2*.
[13] C. Armanini, C. Messer, A. T. Mathew, F. Boyer, C. Duriez, F. Renda, arXiv:2112.03645 [cs] **2021**.
[14] C. Della Santina, C. Duriez, D. Rus, arXiv:2110.01358 [cs, eess] **2021**.
[15] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, G. M. Whitesides, *Proc. Natl. Acad. Sci. USA* **2011**, *108*, 20400.
[16] J. Shintake, V. Cacucciolo, H. Shea, D. Floreano, *Soft Robot.* **2018**, *5*, 466.
[17] U. Gupta, L. Qin, Y. Wang, H. Godaba, J. Zhu, *Smart Mater. Struct.* **2019**, *28*, 103002.
[18] M. T. Tolley, R. F. Shepherd, B. Mosadegh, K. C. Galloway, M. Wehner, M. Karpelson, R. J. Wood, G. M. Whitesides, *Soft Robot.* **2014**, *1*, 213.
[19] G. Li, X. Chen, F. Zhou, Y. Liang, Y. Xiao, X. Cao, Z. Zhang, M. Zhang, B. Wu, S. Yin, Y. Xu, H. Fan, Z. Chen, W. Song, W. Yang, B. Pan, J. Hou, W. Zou, S. He, X. Yang, G. Mao, Z. Jia, H. Zhou, T. Li, S. Qu, Z. Xu, Z. Huang, Y. Luo, T. Xie, J. Gu, et al., *Nature* **2021**, *591*, 66.
[20] J. Cao, L. Qin, J. Liu, Q. Ren, C. C. Foo, H. Wang, H. P. Lee, J. Zhu, *Extreme Mech. Lett.* **2018**, *21* 9.
[21] N. W. Bartlett, M. T. Tolley, J. T. B. Overvelde, J. C. Weaver, B. Mosadegh, K. Bertoldi, G. M. Whitesides, R. J. Wood, *Science* **2015**, *349*, 161.
[22] T. Li, G. Li, Y. Liang, T. Cheng, J. Dai, X. Yang, B. Liu, Z. Zeng, Z. Huang, Y. Luo, T. Xie, W. Yang, *Sci. Adv.* **2017**, *3*, 1602045.
[23] X. Huang, K. Kumar, M. K. Jawed, A. M. Nasab, Z. Ye, W. Shan, C. Majidi, *Sci. Robot.* **2018**, *3*, eaau7557.
[24] D. Drotman, S. Jadhav, D. Sharp, C. Chan, M. T. Tolley, *Sci. Robot.* **2021**, *6*, eaay2627.
[25] W. Huang, X. Huang, C. Majidi, M. K. Jawed, *Nat. Commun.* **2020**, *11*, 2233.
[26] E. Coevoet, T. Morales-Bieze, F. Largilliere, Z. Zhang, M. Thieffry, M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, J. Dequidt, C. Duriez, *Adv. Robot.* **2017**, *31*, 1208.
[27] T. Du, J. Hughes, S. Wah, W. Matusik, D. Rus, *IEEE Robot. Automat. Lett.* **2021**, *6*, 4994.
[28] M. Li, Z. Ferguson, T. Schneider, T. Langlois, D. Zorin, D. Panozzo, C. Jiang, D. M. Kaufman, *ACM Trans. Graph.* **2020**, *39*, 4.
[29] C. Della Santina, R. K. Katzschmann, A. Biechi, D. Rus, in *IEEE Int. Conf. on Soft Robotics (RoboSoft)*, IEEE, Piscataway, NJ **2018**, pp. 46–53.
[30] M. A. Graule, C. B. Teeple, T. P. McCarthy, G. R. Kim, R. C. St. Louis, R. J. Wood, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, IEEE, Piscataway, NJ **2021** pp. 3934–3941.
[31] Z. J. Patterson, A. P. Sabelhaus, C. Majidi, *IEEE Robot. Automat. Lett.* **2022**, *7*, 2210.
[32] X. Huang, W. Huang, Z. Patterson, Z. Ren, M. K. Jawed, C. Majidi, *IEEE Int. Conf. on Robotics and Automation*, IEEE, Piscataway, NJ **2021**, pp. 11884–11890.
[33] A. Spielberg, T. Du, Y. Hu, D. Rus, W. Matusik, *Robotica* **2021**, 1–31.
[34] O. Goury, C. Duriez, *IEEE Trans. Robot.* **2018**, *34*, 1565.
[35] T. Du, K. Wu, P. Ma, S. Wah, A. Spielberg, D. Rus, W. Matusik, arXiv:2101.05917 [cs] **2021**.
[36] C. Armanini, I. Hussain, M. Z. Iqbal, D. Gan, D. Prattichizzo, F. Renda, *IEEE Trans. Robot.* **2021**, *37*, 2083.
[37] S. Kriegman, A. M. Nasab, D. Shah, H. Steele, G. Branin, M. Levin, J. Bongard, R. Kramer-Bottiglio, in *3rd IEEE Int. Conf. on Soft Robotics,* IEEE, Piscataway, NJ **2020** pp. 359–366.
[38] W. L. Scott, D. A. Paley, *Adv. Intell. Syst.* **2020**, *2*, 1900186.
[39] Z. Wolf, G. V. Lauder, *Integr. Comparat. Biol.* **2022**.
[40] J. Bern, P. Banzet, R. Poranne, S. Coros, in *Robotics: Science and Systems XV*, Robotics: Science and Systems Foundation **2019**, ISBN 978-0-9923747-5-4, http://www.roboticsproceedings.org/rss15/p52.pdf.
[41] N. Naughton, J. Sun, A. Tekinalp, T. Parthasarathy, G. Chowdhary, M. Gazzola, *IEEE Robot. Automat. Lett.* **2021**, *6*, 3389.
[42] T. Truong, R. Mysa, T. Stalin, P. A. Raj, P. V. Y. Alvarado, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, IEEE, Piscataway, NJ pp. 7854–7859.
[43] W. L. Scott, P. J. Prakash, D. A. Paley, in *Bioinspired Sensing, Actuation, and Control in Underwater Soft Robotic Systems*, Springer, New York **2021**, pp. 261–279.
[44] S. Tonkens, J. Lorenzetti, M. Pavone, in *IEEE Int. Conf. on Robotics and Automation*, IEEE, Piscataway, NJ **2021** pp. 12010–12016.
[45] N. N. Goldberg, X. Huang, C. Majidi, A. Novelia, O. M. O'Reilly, D. A. Paley, W. L. Scott, *Soft Robot.* **2019**, *6*, 595.
[46] M. D. Bartlett, N. Kazem, M. J. Powell-Palm, X. Huang, W. Sun, J. A. Malen, C. Majidi, *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 2143.
[47] X. Huang, K. Kumar, M. K. Jawed, A. Mohammadi Nasab, Z. Ye, W. Shan, C. Majidi, *Adv. Mater. Technol.* **2019**, *4* 1800540.
[48] X. Huang, Z. Ren, C. Majidi, in *3rd IEEE Int. Conf. on Soft Robotics*, IEEE, Piscataway, NJ **2020** pp. 367–372.
[49] M. Bergou, B. Audoly, E. Vouga, M. Wardetzky, E. Grinspun, *ACM Trans. Graph.* **2010**, *29*, 1.
[50] E. Olson, in *Proc. of the IEEE Int. Conf. on Robotics and Automation,* IEEE, Piscataway, NJ **2011** pp. 3400–3407.
[51] J. Wang, E. Olson, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, IEEE, Piscataway, NJ **2016**, pp. 4193–4198.
[52] A. Nagabandi, G. Yang, T. Asmar, R. Pandya, G. Kahn, S. Levine, R. S. Fearing, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, IEEE, Piscataway, NJ **2018** pp. 4606–4613.
[53] M. Stolle, C. Atkeson, in *Proc. 2006 IEEE Int. Conf. on Robotics and Automation*, IEEE, Piscataway, NJ **2006**. pp. 3344–3349.

[54] S. Arora, S. Choudhury, D. Althoff, S. Scherer, in *IEEE Int. Conf. on Robotics and Automation*, IEEE, Piscataway, NJ **2015** pp. 6431–6438.
[55] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, J. Kuffner, in *7th IEEE-RAS Int. Conf. on Humanoid Robots*, IEEE, Piscataway, NJ **2007** pp. 42–48.
[56] E. Frazzoli, M. Dahleh, E. Feron, in *Proc. of the 39th IEEE Conf. on Decision and Control (Cat. No.00CH37187)*, Vol. *1*, IEEE, Piscataway, NJ **2000** pp. 821–826.
[57] M. Watterson, V. Kumar, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, IEEE, Piscataway, NJ **2015** pp. 3235–3240.
[58] V. K. Viswanathan, E. Dexheimer, G. Li, G. Loianno, M. Kaess, S. Scherer, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, IEEE, Piscataway, NJ **2020** pp. 2510–2517.
[59] X. Huang, M. Ford, Z. J. Patterson, M. Zarepoor, C. Pan, C. Majidi, *J. Mater. Chem. B* **2020**, *8*, 4539.
[60] Y. Du, J. Lam, K. Sachanandani, M. K. Jawed, *IEEE Robot. Automat. Lett.* **2022**, *7*, 6495.
[61] M. K. Jawed, N. K. Khouri, F. Da, E. Grinspun, P. M. Reis, *Phys. Rev. Lett.* **2015**, *115*, 168101.
[62] M. A. Bell, J. C. Weaver, R. J. Wood, *Soft Robot.* **2021**.
[63] D. Tong, A. Choi, J. Joo, M. K. Jawed, arXiv preprint arXiv:2205.10309, **2022**.
[64] A. Wiens, M. Nahon, *Bioinspir. Biomimetics* **2012**, *7*, 046016.
[65] I. Cheeseman, *Aeronaut. J.* **1976**, *80*, 371.
[66] E. Kelasidi, K. Y. Pettersen, J. T. Gravdahl, P. Liljebäck, in *IEEE Int. Conf. on Robotics and Automation,* IEEE, Piscataway, NJ **2014**, pp. 4540–4547.