SoMo: Fast and Accurate Simulations of Continuum Robots in Complex Environments

Moritz A. Graule[1,*], Clark B. Teeple[1,*], Thomas P. McCarthy[1], Grace R. Kim[1], Randall C. St. Louis[1], and Robert J. Wood[1]

***Abstract*— Engineers and scientists often rely on their intuition and experience when designing soft robotic systems. The development of performant controllers and motion plans for these systems commonly requires time-consuming iterations on hardware. We present the SoMo (Soft Motion) toolkit, a software framework that makes it easy to instantiate and control typical continuum manipulators in an accurate physics simulator. SoMo introduces a standardized and human-readable description format for continuum manipulators. It leverages this description format and the Bullet physics engine to enable fast and accurate simulations of soft and soft-rigid hybrid robots in environments with complex contact interactions. This allows users to vary design and control parameters across simulations with minimal effort. We compare the capabilities of SoMo to other physics simulators and highlight the benefits and accuracy of SoMo by demonstrating the agreement between simulation and real-world experiments on several examples; these include an in-hand manipulation task with continuum fingers, an automated exploration of how to design soft fingers for precision grasping, and a brief snake locomotion study. Overall, SoMo provides an accessible way for designers of soft robotic hardware and control systems to gain access to a simulation-accelerated workflow.**

I. Introduction

Soft robots are useful for a variety of applications: from resilient locomotion [1], [2], to gentle grasping [3], [4], [5], [6], to whole-arm manipulation [2], [7], [8], soft robots can provide increased adaptability in the presence of uncertainty in the real world [9]. The dynamics of soft robots, however, are notoriously difficult to model on a physical level due to material and structural compliance. For these reasons, most soft-robotic hardware is controlled using heuristic actuator input trajectories [1], [6], [10] or utilizes simplified dynamic models that are tricky to implement in more-generalized environments [8], [7]. Consequently, the process of designing soft-bodied actuators and their controllers often relies heavily on the intuition and expertise of engineers.

[1]All authors are with the John A. Paulson School of Engineering and Applied Sciences, Harvard University, 60 Oxford St. Cambridge MA 02138, USA.

[*]Authors contributed equally to this work

This material is based upon work supported by the Wyss Institute for Biologically Inspired Engineering; the National Science Foundation (award number EFMA-1830901); a Space Technology Research Institutes grant (number 80NSSC19K1076) from NASA's Space Technology Research Grants Program; and the National Science Foundation Graduate Research Fellowship (under grant DGE1745303). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the funding organizations.

All correspondences should be addressed to Moritz A. Graule, or Robert J. Wood (graulem@g.harvard.edu, rjwood@seas.harvard.edu)

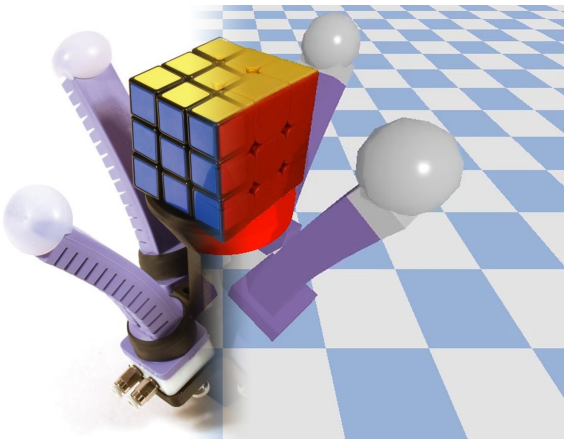

Fig. 1. Composite figure of a soft robotic hand manipulating a Rubik's cube in a hardware experiment (left) and in a simulation that leverages SoMo (right).

On the other hand, the development of traditional robots relies heavily on simulations during design and the evaluation of controllers, control strategies, planning algorithms [11], [12], [13], and learned control policies [14], [15]. A wide variety of physics-based simulation tools are used in traditional robotics such as Gazebo [16], PyBullet [17], Webots [18], and MuJoCo [19]. These general-purpose tools are capable of quickly simulating full environments with multiple robots and complex interactions between robots and their environment. These simulators typically assume that all robots and objects are rigid, making it difficult to model the continuous deformation of soft robots.

A comparable simulation-accelerated development workflow is mostly missing for soft robots. One of the main reasons for this is the lack of standardized and generalizable tools that allow engineers to accurately simulate soft robotic systems in the presence of robot-robot and robot-environment interactions. Consequently, soft roboticists often rely on customized, project-specific physics engines to leverage simulations in their development process [20], [21]. In this work, we present our **So**ft **Mo**tion framework (SoMo): a standardized framework for simulating continuum robots using an off-the-shelf rigid-body physics engine (Bullet/PyBullet). The key features of this framework are: 1) simulations are fast to set up even for complex environments with soft-rigid hybrid robots; 2) the code is open-source to facilitate community-driven development; and 3) it is straightforward to accurately calibrate simulations, such that they closely align with experiments in hardware.

II. Related Work

A number of rigid-body robot simulators exist; they are well-suited for the simulation of complex rigid-bodied environments, and many of them have a wide and active user base. However, building and controlling continuum manipulators in these simulators using a rigid-body approximation typically requires a user to generate and control many different joints, which is time-consuming to set up.

Motivated by the strong benefit of simulations in the development of hardware and controllers for traditional robots, several frameworks have been proposed for the design and simulation of continuum robots at different levels of system abstractions. These frameworks make use of full FEA simulations [22], [21], voxel- or particle-based approximations [23], [24], or approximations of slender continuum robots as bending beams [25], [26]. While some of these frameworks have been shown to achieve high accuracy and simulation speeds approaching real-time [25], [22], [26], [24], they are often built with a specific use case in mind. Using these frameworks to simulate custom soft robots in varied environments (e.g., environments that include additional objects or traditional rigid-body dynamical systems) is often not straightforward and requires significant effort. Furthermore, defining new soft robotic systems in these simulators typically requires in-depth knowledge of the respective software toolkit.

Table I provides an overview of the capabilities of various existing simulators for rigid and soft robots, highlighting how a simulation framework that streamlines the definition of continuum robots in different environments would enable faster iterations on designs and control policies for continuum manipulators in soft-rigid hybrid systems or complex environments. A rigorous comparison of simulation accuracy, speed, and computation requirements for various robot simulations tools is outside the scope of this paper. Furthermore, the outcome of such a comparison will heavily depend on the hardware employed and the specifics of the simulated systems.

III. Framework Definition

The SoMo framework couples a discretized rigid-body model of soft actuators (shown in Figure 2) with an easy-to-use Python module to build actuators and manipulators from user-specified descriptions defined in json or yaml files, or Python dictionaries. SoMo automatically builds rigid-link approximations of continuum manipulators using a standard format (universal robot description file, or URDF), and imports them into the PyBullet physics engine [17]. During a simulation run, the framework handles the individual joint springs and actuation torques for all joints in the simulated manipulators, such that individual actuators can be addressed with a single control input for each of their actuated axes. The restoring torques for passive axes of compliance are updated automatically at each simulation step. We also provide a streamlined method to calibrate model parameters to match real hardware systems characterized with a few simple measurements. Altogether, SoMo simplifies the setup and simulation process for soft manipulators in varied environments. A detailed documentation of SoMo and a collection of user-friendly examples can be found on the project's GitHub page [29].

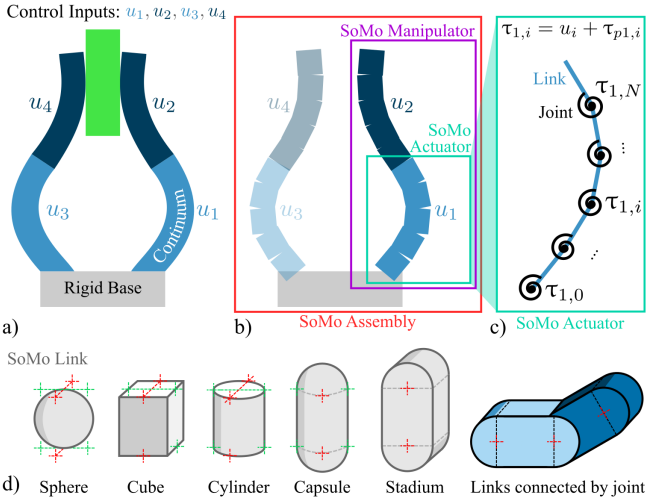

Fig. 2. The SoMo framework represents complex assemblies of continuum robots using a modular rigid-body approximation that matches the number of control inputs as the real system. a) For example, a soft gripper with two fingers, each with two independent segments has four control inputs. b) In SoMo, this gripper is represented as an assembly with a rigid base and two continuum manipulators. Each manipulator is made up of two actuators with one control input each. c) Each actuator is represented as a serial chain of rigid bodies connected with spring-loaded joints, all responding to the single actuation input, u_1. d) Several link geometries are available.

A. Actuator Definition

Modelling soft actuators as discretized rigid-link serial manipulators with spring-loaded joints allows us to take advantage of rigid-body physics engines. In SoMo, each actuator is split into several discrete segments with revolute or prismatic joints between them. Joints are defined to have stiffness and damping; can be defined along any axis; and can be offset along any axis relative to links. The user can choose from a wide selection of link shapes when specifying a robot in SoMo, including spherical, cylindrical, cuboid, capsule-shaped links, and right prisms with a stadium-shaped base. Actuators can then be chained together in series to create manipulators, such as dexterous soft fingers, continuum arms, or snake-inspired robots. Manipulators can be specified to have additional features at their ends, such as base links or fingertips. Manipulators can be constrained at the base, or allowed to move around freely. Manipulators can also be grouped into manipulator assemblies, which facilitate the construction and control of complex soft-rigid hybrid robot systems, such as a traditional robot arm with a soft gripper as the end effector.

In SoMo, the geometry and dynamics of continuum manipulators, actuators, joints, and links are prescribed in human-readable definitions (either json or yaml files, or Python dictionaries), which makes it easy to quickly set up and adjust simulated environments. Actuation torques can be applied to continuum manipulators and continuum manipulator assemblies with one line of code in the simulation loop.

TABLE I

OVERVIEW OF POPULAR ROBOT SIMULATION FRAMEWORKS

Framework	Modeling approach	Physics model complexity	Installation	User interface	URDF*	Rigid robots*	Continuum robots*	Rigid-soft hybrid robots*	Open source	Ref.
Gazebo	rigid-body physics	low	Install ROS & compile	GUI, ROS (C++ or Python)	✓	✓	✗	✗	✓	[16]
MuJoCo	rigid-body physics	low	Executable	C++, Python	✓	✓	✗	✗	✗	[19]
PyBullet/Bullet	rigid-body physics	low	Python pip	C++, Python	✓	✓	✗	✗	✓	[17]
Webots	rigid-body physics	low	Executable	C, C++, Java, Python, MATLAB	✗	✓	✗	✗	✓	[18]
Elastica	Cosserat rods	medium-high	Python pip	C++, Python	✗	✗	✓	✗	✓	[27]
Huang et al. (2020)	Cosserat rods	medium-high	Compile source	C++	✗	✗	✓	✗	✗	[26]
ChainQueen	Particle-grid hybrid	high	Compile source	Python, Taichi	✗	✗	✓	✗	✓	[24]
SOFA	FEA	high	Executable	GUI, C++, Python	✗	✓	✓	✓	✓	[28]
PyBullet + SoMo	rigid-body physics	low	Python pip	Python	✓	✓	✓	✓	✓	

*indicates native support

B. Calibration

SoMo requires similar parameter tuning as the underlying PyBullet physics engine (specifically, tuning of friction, world scale, time step). In addition, the bending stiffness and damping need to be specified for each actuator. The following provides guidelines on this tuning process.

1) Bending stiffness: We present a standardized calibration method to relate real-world actuator deformation to joint angles in the simulated actuators that relies on simple physical properties that are easy to measure. Using a measurement of the linear bending stiffness of the actuators in two directions, we can approximate the appropriate simulated joint stiffnesses such that the tip deflection of each actuator segment matches the real-world deflection profile of the actuator. We then use blocked force measurements to estimate the equivalent simulated actuation torque required to achieve the same internal actuation moment during pressurization as the physical system. While a data-driven calibration procedure (similar to [30]) could be used in place of this calibration procedure, a full motion tracking system would be required. Our calibration procedure achieves reasonable accuracy using only a few easily-measurable physical parameters.

For an actuator split evenly into N segments, we begin with definitions and assumptions. Under the assumption of small deformations, the deflection δ of a cantilever beam as a function of position x along the beam can be computed as

$$\delta(x) = \delta_L \left(\frac{3x^2}{2L_2} - \frac{x^3}{2L^3} \right), \tag{1}$$

where δ_L is the deflection at the tip, and L is the length of the beam. In addition, the bending moment M_b of a cantilever beam with a tip load F_{tip} as a function of x along the beam is defined as

$$M_b = F_{tip}(L-x), \tag{2}$$

and the linear bending stiffness K_b of a cantilever beam at the tip as

$$K_b = \frac{F_{tip}}{\delta_L}. \tag{3}$$

Finally, we assume the calibration takes place over small deformations and angles, so $sin(\theta) = \theta$.

We first find the deflection at the tip of the first segment due to a load at the actuator's tip to be:

$$\delta(x = \frac{L}{N}) = \left(\frac{3}{2N^2} - \frac{1}{2N^3} \right) \delta_L = \delta_1 \tag{4}$$

For the first segment with length $\frac{L}{N}$ and deflection δ_1 we define a rotational spring constant κ_1 that achieves an angle θ_1 when a torque τ_1 is applied to the respective joint:

$$\kappa_1 = \frac{\tau_1}{\theta_1} \tag{5}$$

Solving geometrically for θ_1 and using the small angle approximation, we obtain:

$$\theta_1 \approx \sin(\theta_1) = \frac{\delta_1}{L_1} = \frac{\delta_1}{\frac{L}{N}} = \frac{N\delta_1}{L} \tag{6}$$

The bending moment withheld by this first link (i.e., the moment at $x = 0$), τ_1, is assumed to be:

$$\tau_1 = F_{tip}(L-0) = F_{tip}L \tag{7}$$

Finally, we can substitute (6) and (7) into (5) to obtain:

$$\kappa_1 = \frac{\tau_1}{\theta_1} = \frac{1}{\frac{3}{2N} - \frac{1}{2N^2}} L^2 K_b \tag{8}$$

In addition to joint stiffnesses, we must determine the joint torques that are to be applied to drive the actuators. For fluid-driven soft robotic systems, we can use blocked-force measurements at various pressures to obtain estimates of the actuation torques at pressures of interest. During a blocked force measurement, we assume that the forces that are produced at the actuator tip balance out the internal actuation moment. Given this assumption, we can use

$$M_{act} = F_b L \tag{9}$$

to approximate the actuation moment applied at each pressure. Additional details on calibration and examples that facilitate it can be found on the SoMo Github page [29].

2) Damping: SoMo simulates flexible joints by controlling joint torques so that they are a function of the joint angle and the approximated joint stiffness. The rate at which these joint torques are controlled can be limited using the 'joint_control_limit_force' property of the joint definitions. We empirically determined that setting this limit between 0.15 and 1.5 results in numerically stable and realistic simulations and that varying it within that range does not significantly affect the simulation results.

3) Friction: PyBullet defines a (non-physical) friction value for each object. The traditional friction coefficient between two objects is then computed as the product of the objects' friction values. To obtain realistic simulations in SoMo, one can either manually tune the objects' friction values, or set them such that the deviation between measured (physical) and calculated friction coefficients for all objects of interest is minimized.

4) Scaling: In some cases, it may be beneficial to scale the world in PyBullet (and therefore, SoMo) for numerical stability (Bullet was originally designed for dimensions and velocities between 0.05 and 10). To scale the world by a factor X, the following properties should be scaled by a factor of X: collision shapes about origin, positions, linear (but not angular) velocities, linear [Sleep Threshold], gravity, user-defined impulses, and inertias (if not computed by Bullet). Torques need to be scaled by a factor of X^2. Damping and angular velocity do not need to be changed.

IV. Accurate Simulation of Continuum Manipulators in Complex Environments

We showcase the capability and accuracy of SoMo and demonstrate the calibration workflow outlined above on a challenging in-hand manipulation task with a soft robotic hand. In this task, a cube is rotated around its axis by four dexterous fingers with two independently controlled degrees of freedom. To achieve large object rotations in this setup, the fingers need to repeatedly make and break contact with the manipulated object.

The hardware setup for this experiment is described and characterized in detail in [10]; in short, the fingers are 0.1m long, achieve a blocked force (as measured at the fingertip) of 0.75N when actuated with a pressure of 100kPa, and the actuation pressures are between 0 and 240kPa. We scaled

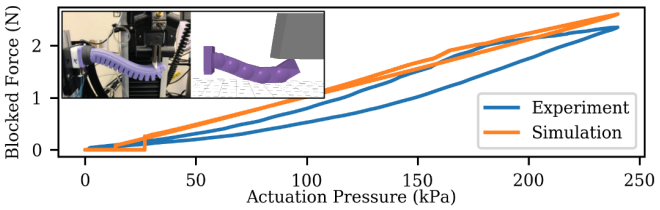

Fig. 3. Actuator-level validation of calibrated SoMo simulations. The fingers were characterized using a blocked force measurement (inset, top). We replicated this setup in simulation, measuring the normal force between the finger tip and a rigid cube (inset, bottom). The experimental and simulated actuation-force curves show a good agreement between hardware and simulations (RMSE = 0.28N).

the world by a factor of 20 and calibrated the SoMo simulation as outlined above. We then validated the simulation accuracy at the actuator-level by replicating the blocked-force characterization experiments in simulation. The results of this study, shown in Figure 3, confirm that we can obtain accurate actuator-level calibration.

Having confirmed that simulations closely match hardware experiments on actuator-level tasks, we show that SoMo simulations with calibrated finger definitions produce similar behaviour to real hardware for a complex in-hand manipulation task. Using the four-fingered hand developed in [10], we implemented an experimentally-determined finger gait that achieves continuous rotation about the object's z-axis. This finger gait is a combination of rotation primitives performed by alternating pairs of fingers on a target object which enables the hand to rotate the object well beyond the dexterous workspace of the fingers.

The finger gait used in [10] is defined in terms of four sets of two pneumatic pressure signals that differentially actuate the soft fingers. These pressure signals must be converted to equivalent orthogonal actuation signals for the simulations. We use a simple linear map achieve this. We calculate the real actuation torques achieved by these soft fingers at each pressure in the trajectory by utilizing the calibration procedure for actuation torques described above. This results in a linear map from real differential actuation pressures p_{diff} to simulated orthogonal actuation torques τ:

$$\begin{bmatrix} \tau_{grasp} \\ \tau_{side} \end{bmatrix} = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} p_{diff1} + p_{diff2} \\ p_{diff1} - p_{diff2} \end{bmatrix}, \tag{10}$$

where $w_1 = 1.0$ and $w_2 = 3.25$.

With these calibrated actuator inputs, we performed the finger gait on both the real hardware and simulated hand, as shown in Figure 4. Using a cube of dimension 60 mm and mass of 20 g, we applied the finger gait trajectory to both systems. In the simulated system, the object's 6-axis pose was logged directly. For the real hardware, we used a custom real-time pressure controller to actuate the hand (the same system as in [10]), and measured the 6-axis pose of the object with April Tags [31] via a webcam viewing from the top.

Our experiments show that the SoMo simulations produce similar task-level behavior to the real system for complex

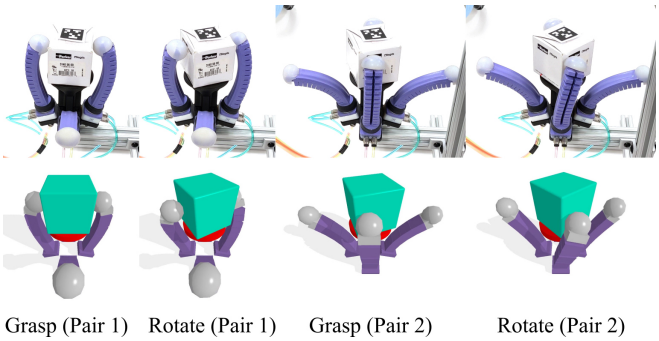

Fig. 4. The simulation framework is validated against real hardware on a complex finger gait for continuous rotation of the object within the hand. A comparison of several key events during a single gait cycle are shown.

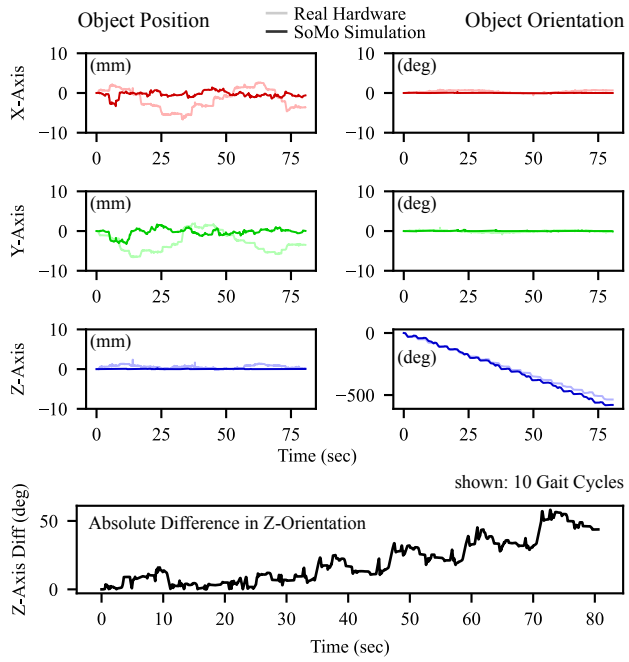

Fig. 5. SoMo produces similar behaviour to real hardware for a complex in-hand manipulation gait, validating our calibration process. Using a finger gait for continuous rotation developed in [10], object's rotation about the z-axis after ten gait cycles in simulation is within 9% of rotation when the same gait is performed on real hardware. All other off-axis motion is of similar orders of magnitude. *Line colors indicate x, y, and z-axes.*

tasks with multiple contact points. The object's z-orientation in simulation tracks the orientation of the real system within 9% of the real hardware after ten gait cycles (50° drift over the full-scale 582°), as shown in Figure 5. All off-axis motion is of similar orders of magnitude for both systems and is very small (x and y positions within 6 mm, x and y orientations within 0.6°, and z positions within 2 mm). The deviations in x and y positions are likely due to small differences in friction and contact between the simulated and real systems. The simulated z-position, x-orientation, and y-orientation have especially small magnitudes compared to the real system. This is likely due to measurement variation in the real system. Overall, these results validate that a simple finger-level calibration can produce realistic simulation behavior for complex tasks.

V. Additional Demonstrations

To illustrate the power and versatility of the SoMo framework, we apply it to several additional applications ranging from optimizing soft fingers for pinch grasping, to manipulation, to locomotion. Each case study showcases different aspects of the framework, and together these demonstrations solidify the utility of the SoMo framework.

A. Design Exploration: Pinch Grasping

One extremely useful application of a fast simulation framework is to efficiently perform large-scale design explorations. With sufficiently-accurate calibration, the SoMo platform provides an easy way to set up simulations for multi-dimensional sweeps of design parameters that return realistic results.

In recent work on precision grasping with soft robotic grippers, a purely experimental approach was taken to elucidate the effect of fingers with independent serial bending actuators in pinch grasping [6]. One key finding of this study was that the length ratio of serial bending segments in a soft finger affects the range of object sizes that can be grasped precisely. There is a trade-off between strong power grasping capabilities achieved with short passive distal segments, and precise pinch grasping achieved with longer distal segments. Extending this study to find the smallest distal segment length ratio (DSR) capable of pinch grasping a wide range of objects would enable the design of fingers that maximize power grasping strength while ensuring reliable pinch grasping.

To further understand this phenomenon with high granularity, we simulated the grasping experiments from [6] using SoMo. We are primarily interested in how the DSR affects the transition between pinch grasping and power grasping as a function of the object's width and the hand's centering position with respect to the object. Thus, for a simple grasping task, we set up a three-dimensional parameter sweep over centering positions (70 mm to 180 mm, in 5 mm increments), object widths (10, 20, 40, 60, 100, and 120 mm), and distal segment ratios (0 to 0.5, in 0.05 increments).

For each set of parameter values, a simple grasping task is performed with a two-finger soft gripper. First, simulated soft fingers were calibrated to the real fingers in the study as per the procedure outlined in Section III-B. We used 20 segments per finger split into two independently-actuated segments. The fingers are arranged into a two-fingered antipodal gripper as in [6], and a target object is spawned at the origin. The gripper is moved to the desired centering position (with respect to the object), attempts to grasp the object, and then lifts the object 75 mm. Grasps are performed by applying an actuation torque corresponding to 100 kPa of pneumatic pressure in the physical system, while leaving the distal segments unactuated. We classified the resulting grasp types based on the contact locations along the fingers.

The results of these experiments indicate that a DSR of 0.20 is the smallest DSR that achieves pinch grasping for all object sizes. As shown in Figure 6, the pinch grasping region for DSRs lower than 0.2 does not span all object

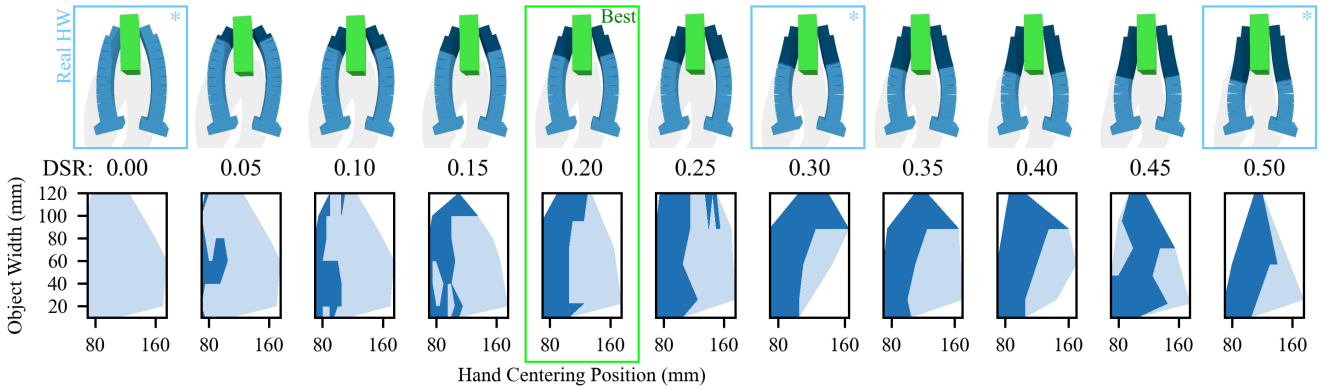

Fig. 6. The smallest DSR that achieves pinch grasping (dark regions) for all object sizes is 0.20, which maximizes pinch-grasping capabilities while also maximizing power-grasping strength according to [6]. For each DSR, a simulated grasp on a 20 mm-thick box demonstrates the change in finger shape from fingertips to pinch-grasping. These finger shapes and trends in the resulting grasp data align well with the real hardware, which was tested at three values of the DSR (indicated with *). The region of acquisition for precision grasping (dark regions) increases in area as a function of the distal segment ratio (DSR), and the power grasping region (light regions) decreases in area.

widths. This is likely due to the fact that shorter distal segments have higher stiffnesses, causing grasping instabilities when pinch-grasping. For DSRs higher than 0.2, the pinch grasping region increases in area, indicating that pinch grasps become less sensitive to the gripper's distance behind the object. However, as shown in [6], the grasping strength decreases with increasingly long distal segments. Overall, soft fingers with a DSR of 0.2 strike a balance between precision grasping capabilities and power grasping strength. These observations are consistent with the trends observed in hardware experiments [6].

The first benefit of the SoMo platform for large-scale design explorations is the reduction in time required to prepare multiple actuator designs. Setting up this study in SoMo requires only a few hours. In real soft-robotic hardware, each design iteration would take approximately 1-2 weeks and require the generation and fabrication of new mold designs and actuators [6].

The second benefit of the SoMo platform for design studies is the ability to quickly perform a large set of experiments with no active user time. In this pinch grasping study, each single grasping task ran at ∼50% real time on a modern desktop computer (Ubuntu 20.04, 3.5GHz CPU with four cores). Utilizing all four cores of our machine, the set of 2310 experiments (each a 7 sec. grasping task) ran in a total of 2.3 hours. No intervention or monitoring was required during these experiments. In contrast, running the same number of experiments on real hardware would take 4.5 hours of active user participation, since runs must be performed in sequence - excluding setup time between experiments. Overall, running the simulations in this study took approximately 1/4 the time that would be required to do them with real hardware.

The final benefit of SoMo simulations is that they produce consistent soft robot behavior, with no added effort required to monitor all aspects of the experimental environment, including phenomena which are difficult to measure in real-life (e.g., exact contact locations between fingers and objects). For example, our simulated experiments achieve less variation in finger and object motion because simulated fingers are not subject to manufacturing errors or fatigue, and object locations in the simulated world are controlled precisely. Should a user want to study the effect of manufacturing error, fatigue, or other uncertainties on the performance of soft robots, it is straightforward to introduce respective probabilistic variables into the SoMo simulations. In addition, PyBullet provides access to the poses of all links in the fingers and all objects, as well as the contact vectors between all entities in the environment. Furthermore, measuring the poses of objects and soft actuators in real environments usually requires some motion tracking or perception system, and measuring contact conditions between fingers and object is usually challenging and prone to measurement noise, or outright intractable.

B. Design Exploration: In-hand Manipulation with an Actuated Palm

We further used SoMo to gain insights into suitable designs for a hand that leverages soft, pneumatic fingers and a rigid palm to achieve dexterous in-hand manipulation. In such a system, contacting the manipulated object with the fingertips (as opposed to along the manipulators) maximizes the object motion that can be achieved within one 'contact-cycle' (i.e., a period in time during which no regrasping occurs). Placing the contact points as close to the object's center of mass as possible minimizes the effect of destabilizing torques caused by gravity. An actuated palm with adjustable height can ensure that objects of varying sizes can be supported by the palm while the fingertips contact it near its center of mass.

Ensuring contact between object and palm further facilitates in-hand manipulation tasks, as contact with the palm stabilizes the object and reduces its unconstrained degrees of freedom, thus lowering the uncertainty in object pose. For larger palm diameter, this remains true for large deviations in object pose. This means that a large palm diameter is generally desirable. However, if the palm is brought closer

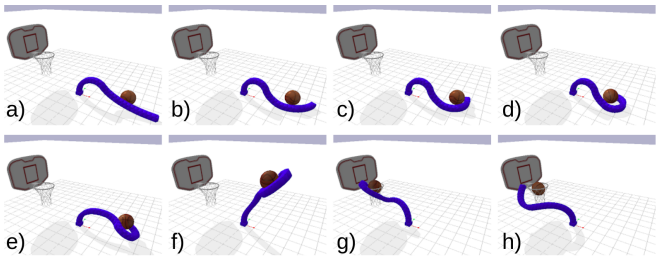

Fig. 7. Simulation of a continuum manipulator arm picking up a basket ball and placing it into a hoop. a-h) shows sequential screenshots from the simulation. The actuation torques required to complete this task were determined heuristically.

towards the fingertips, as is necessary to ensure that small objects contact both the palm and the fingertips, the palm must not be too wide, as it would otherwise obstruct the grasp.

This motivated our development of a soft-robotic manipulation platform that leverages a palm with controllable height and diameter. In this development process, we explored the design space of varying palm heights and diameters through SoMo simulations. A detailed discussion of this exploration, a description of the final system, and a demonstration of its capabilities is presented in [32].

C. Customizable Environments: Whole-arm Manipulation

We showcase the flexibility and ease in defining custom environments by simulating a whole-arm manipulation task in which a continuum manipulator (consisting of five actuators with two independent degrees of freedom each) is used to pick up a basket-ball and place it into a hoop, shown in Figure 7. In this example, the actuator torques used to envelop and move the basketball were determined by trial and error. This process is more convenient in (parallelizable) simulations than with physical hardware for the reasons outlined in Section V-A.

D. Floating-Base Robots: Snake Locomotion Through Anisotropic Friction

SoMo maintains all the invaluable features available in Bullet, while facilitating the manual or programmatic specification of continuum robots. In this example, we leverage PyBullet's anisotropic friction feature to simulate realistic snake locomotion - which leverages anisotropic friction at the snake's bottom scales [33]. The simulated snake manipulator consists of 50 segments; an active torque τ is applied to each joint for locomotion. This torque is a function of the joint position along the actuator x and time t, such that $\tau(x,t) = A(t)\sin(kx - \omega t)$. To avoid numerical instabilities, the amplitude $A(t)$ of this wave is linearly increased from $A(t=0) = 0$ at the beginning of the simulation to $A(t=1) = A_1 = 1$ and held constant afterwards. The resulting snake poses are shown for isotropic and anisotropic friction in Figure 8; a snake with isotropic friction is unable to move in the desired direction with sinusoidal actuation, while a snake with anisotropic friction consistently moves forward.

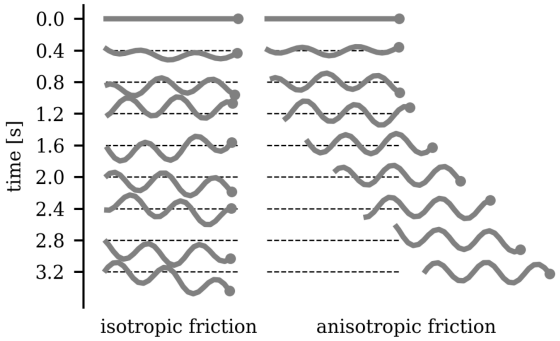

Fig. 8. Simulated snake locomotion with isotropic (left) and anisotropic (right) friction. A snake-like continuum manipulator is actuated with a sinusoidal torque function of the form $A(t)\sin(kx - \omega t)$. The manipulator pose (gray line) and its start pose (black dotted line) are shown at different time points. The simulated snake requires anisotropic friction to move in the desired direction, agreeing with observations of live snakes [33].

VI. Conclusions and Outlook

We introduced SoMo, a toolkit that facilitates the preparation and execution of simulations with soft robots in complex and varied environments. This is achieved through a standardized definition of continuum robots and a thin wrapper that facilitates their instantiation and control in PyBullet. We show that SoMo is versatile, accurate, and runs in near real-time for diverse demonstrations including pick-and-place, dexterous manipulation, and locomotion.

This makes SoMo a valuable tool for the development of soft robotic systems. Specifically, SoMo provides four distinct benefits for the large-scale exploration of design parameters: 1) preparing new actuator designs is many times faster in simulation; 2) experiments for various design parameters can be performed in parallel without requiring active user interaction; 3) it produces consistent results; and 4) it allows for the monitoring of all experimental parameters, including those that are hard or impossible to observe in hardware experiments. In addition, SoMo can be used to evaluate sensing algorithms, planned or heuristically determined trajectories, and controller parameters.

We showcased these capabilities through a series of examples. We demonstrated a complex in-hand manipulation gait which tracks the behavior of the real hardware within 9%. In addition, we highlight the value of SoMo in conducting design exploration studies by exploring the relevant parameters in two soft-robotic platforms: one developed for precision grasping with antipodal fingers, the other employing an active palm with varying height and diameter to achieve in-hand manipulation with increased dexterity.

In future work, we aim to further refine the agreement between simulation and experiments through non-linear models for joint stiffnesses, and a simulation-in-the-loop calibration routine. This will enable us to leverage SoMo to learn improvements of and alternatives to our experimentally-determined control policies for in-hand manipulation in simulation that can be executed successfully on hardware. Additionally, we will explore using SoMo for the real-time validation and correction of motion plans for continuum manipulator.

Acknowledgements

The authors thank Daniel Bruder and Justin Werfel for helpful discussions and comments on the manuscript.

References

[1] M. T. Tolley, R. F. Shepherd, B. Mosadegh, K. C. Galloway, M. Wehner, M. Karpelson, R. J. Wood, and G. M. Whitesides, "A resilient, untethered soft robot," *Soft robotics*, vol. 1, no. 3, pp. 213–223, 2014.

[2] M. Cianchetti, M. Calisti, L. Margheri, M. Kuba, and C. Laschi, "Bioinspired locomotion and grasping in water: the soft eight-arm octopus robot," *Bioinspiration & biomimetics*, vol. 10, no. 3, p. 035003, 2015.

[3] K. C. Galloway, K. P. Becker, B. Phillips, J. Kirby, S. Licht, D. Tchernov, R. J. Wood, and D. F. Gruber, "Soft Robotic Grippers for Biological Sampling on Deep Reefs," *Soft Robotics*, vol. 3, no. 1, p. soro.2015.0019, 2016.

[4] R. Deimel and O. Brock, "A novel type of compliant and underactuated robotic hand for dexterous grasping," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 161–185, 2016.

[5] J. Zhou, X. Chen, U. Chang, J.-T. Lu, C. C. Y. Leung, Y. Chen, Y. Hu, and Z. Wang, "A soft-robotic approach to anthropomorphic robotic hand dexterity," *IEEE Access*, vol. 7, p. 101483–101495, 2019.

[6] C. B. Teeple, T. N. Koutros, M. A. Graule, and R. J. Wood, "Multi-segment soft robotic fingers enable robust precision grasping," *International Journal of Robotics Research*, 2020.

[7] A. D. Marchese, R. Tedrake, and D. Rus, "Dynamics and trajectory optimization for a soft spatial fluidic elastomer manipulator," *International Journal of Robotics Research*, vol. 35, no. 8, pp. 1000–1019, 2016.

[8] R. K. Katzschmann, A. D. Marchese, and D. Rus, "Autonomous object manipulation using a soft planar grasping manipulator," *Soft robotics*, vol. 2, no. 4, pp. 155–164, 2015.

[9] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, p. 467, 2015.

[10] S. Abondance, C. B. Teeple, and R. J. Wood, "A dexterous soft robotic hand for delicate in-hand manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5502–5509, 2020.

[11] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1366–1373.

[12] Z. Manchester and S. Kuindersma, "Dirtrel: Robust trajectory optimization with ellipsoidal disturbances and lqr feedback." in *Robotics: Science and Systems*, 2017.

[13] D. Driess, O. Oguz, and M. Toussaint, "Hierarchical task and motion planning using logic-geometric programming (hlgp)," in *RSS Workshop on Robust Task and Motion Planning*, 2019.

[14] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, and et al., "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, p. 3–20, Jan 2020.

[15] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 627–12 637.

[16] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, Sep 2004, pp. 2149–2154.

[17] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[18] O. Michel, "Webots: Professional mobile robot simulation," *Journal of Advanced Robotics Systems*, vol. 1, no. 1, pp. 39–42, 2004. [Online]. Available: http://www.ars-journal.com/International-Journal-of-Advanced-Robotic-Systems/Volume-1/39-42.pdf

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

[20] R. Gasoto, M. Macklin, X. Liu, Y. Sun, K. Erleben, C. Onal, and J. Fu, "A validated physical model for real-time simulation of soft robotic snakes," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6272–6279.

[21] O. Goury and C. Duriez, "Fast, generic, and reliable control and simulation of soft robots using model order reduction," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1565–1576, 2018.

[22] E. Coevoet, T. Morales-Bieze, F. Largilliere, Z. Zhang, M. Thieffry, M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, J. Dequidt, and C. Duriez, "Software toolkit for modeling, simulation, and control of soft robots," *Advanced Robotics*, vol. 31, no. 22, pp. 1208–1224, 2017. [Online]. Available: https://doi.org/10.1080/01691864.2017.1395362

[23] J. Hiller and H. Lipson, "Dynamic simulation of soft multimaterial 3d-printed objects," *Soft robotics*, vol. 1, no. 1, pp. 88–101, 2014.

[24] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, "Chainqueen: A real-time differentiable physical simulator for soft robotics," in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6265–6271.

[25] S. Grazioso, G. Di Gironimo, and B. Siciliano, "A geometrically exact model for soft continuum robots: The finite element deformation space formulation," *Soft robotics*, vol. 6, no. 6, pp. 790–811, 2019.

[26] W. Huang, X. Huang, C. Majidi, and M. K. Jawed, "Dynamic simulation of articulated soft robots," *Nature communications*, vol. 11, no. 1, pp. 1–9, 2020.

[27] M. Gazzola, L. Dudte, A. McCormick, and L. Mahadevan, "Forward and inverse problems in the mechanics of soft filaments," *Royal Society open science*, vol. 5, no. 6, p. 171628, 2018. [Online]. Available: https://doi.org/10.1098/rsos.171628

[28] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin, "SOFA: A Multi-Model Framework for Interactive Physical Simulation," in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, ser. Studies in Mechanobiology, Tissue Engineering and Biomaterials, Y. Payan, Ed. Springer, Jun. 2012, vol. 11, pp. 283–321. [Online]. Available: https://hal.inria.fr/hal-00681539

[29] M. A. Graule and C. B. Teeple, "Somo framework," https://github.com/GrauleM/somo, 2021.

[30] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Koopman-based control of a soft continuum manipulator under variable loading conditions," *arXiv preprint arXiv:2002.01407*, 2020.

[31] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, oct 2016, pp. 4193–4198.

[32] C. B. Teeple, G. R. Kim, M. A. Graule, and R. J. Wood, "An active palm enhances dexterity for soft robotic in-hand manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[33] D. L. Hu, J. Nirody, T. Scott, and M. J. Shelley, "The mechanics of slithering locomotion," *Proceedings of the National Academy of Sciences*, vol. 106, no. 25, pp. 10 081–10 085, 2009.