Robotics: Science and Systems 2022
New York City, NY, USA, June 27-July 1, 2022

Soft Robots Learn to Crawl: Jointly Optimizing Design and Control with Sim-to-Real Transfer

Charles Schaff*, Audrey Sedal†, Matthew R. Walter*
*Toyota Technological Institute at Chicago, Illinois, USA 60637
Email: {cbschaff,mwalter}@ttic.edu
†Department of Mechanical Engineering, McGill University, Montreal, Canada
Email: audrey.sedal@mcgill.ca

***Abstract*—This work provides a complete framework for the simulation, co-optimization, and sim-to-real transfer of the design and control of soft legged robots. The compliance of soft robots provides a form of "mechanical intelligence"—the ability to passively exhibit behaviors that would otherwise be difficult to program. Exploiting this capacity requires careful consideration of the coupling between mechanical design and control. Co-optimization provides a promising means to generate sophisticated soft robots by reasoning over this coupling. However, the complex nature of soft robot dynamics makes it difficult to achieve a simulation environment that is both sufficiently accurate to allow for sim-to-real transfer and fast enough for contemporary co-optimization algorithms. In this work, we describe a modularized model order reduction algorithm that significantly improves the efficiency of finite element simulation, while preserving the accuracy required to successfully learn effective soft robot design-control pairs that transfer to reality. We propose a reinforcement learning-based framework for co-optimization and demonstrate successful optimization, construction, and zero-shot sim-to-real transfer of several soft crawling robots. Our learned robot outperforms an expert-designed crawling robot, showing that our approach can generate novel, high-performing designs even in well-understood domains.**

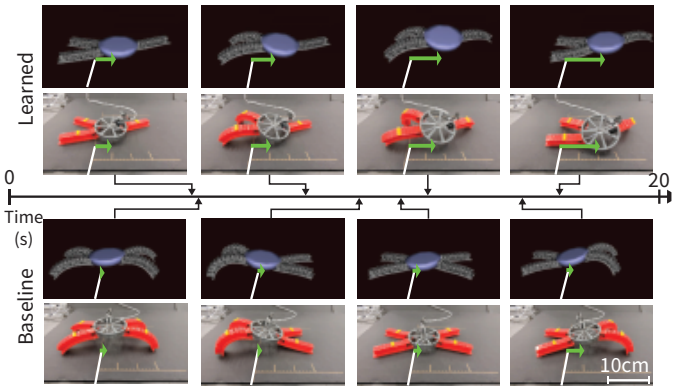


Fig. 1. Our framework jointly learns the design and control of crawling soft robots (top) that outperform an expert-designed baseline (bottom). While trained exclusively in simulation, our learned robots are capable of zero-shot sim-to-real transfer, with the optimal design moving more than 2× faster than the baseline in the real world.

I. Introduction

The deformable nature soft robots enables designs that respond to contact or control inputs in sophisticated ways, with behaviors that have proven effective across a variety of domains. The design of these robots is tightly coupled with the policy that controls their motion, giving rise to a form of "mechanical intelligence" [49] in which materials and mechanisms respond to their environment in useful ways that augment functionality, e.g., conforming to an object to create a better grasp or storing elastic energy to improve the efficiency and power of a walking gait. Therefore, methods that jointly optimize both the robot's physical design and its control policy provide a promising approach to realizing mechanically intelligent soft robots. However, while there is extensive work on joint design-control optimization in the context of rigid robots [53, 42, 44, 45, 55, 51, 14, 22, 64, 50, 21, 8, 43], relatively little exists for soft robotics.

This work provides a complete framework for the simulation, co-optimization, and sim-to-real transfer of the design and control of soft robots. Integral to this framework, we propose a co-optimization algorithm that utilizes multi-task deep reinforcement learning to generate a design-aware policy capable of generalizing across the space of designs. The algorithm exploits this policy to quickly focus its search on high-performing designs. To encourage "mechanical intelligence", we learn an open-loop controller, forcing complex behavior to be expressed through the resulting soft body.

An important prerequisite for co-optimization is a simulator that is both fast enough to explore a large set of designs and control strategies and accurate enough to ensure that the learned robots are physically realizable and capable of sim-to-real transfer. However, modelling soft bodies is both challenging and computationally intensive. The best way to simulate soft bodies for robotics is an open question, and the few existing co-optimization approaches for soft robotics suggest different simulation strategies [27, 54, 24]. However, these simulators have varying degrees of realism and their ability to produce soft robots that cross the reality gap is unclear.

With the focus of sim-to-real transfer in mind, we choose to employ finite element analysis (FEA), which is the defacto standard for simulating deformable materials with a high degree of accuracy. In this work, we show that FEA simulations, with enough computation, can allow for direct transfer of co-optimized soft robots. However, high-fidelity FEA simulations can be hundreds of times slower than real-time, rendering learning-based methods infeasible. In order to improve the computational complexity of FEA simulation while preserving

its accuracy, we extend the recent work of Goury et al. [20] that proposes a model order reduction technique for soft robotics in the open-source FEA simulation framework SOFA [17, 10]. While their method improves the computational efficiency of simulation, it incurs a large initial cost that prohibits learning over different designs. We propose a reconfigurable reduction framework that reduces a set of composable parts that can then be combined to create reduced order models of soft robots with varying morphologies. This allows for simulations with a computational efficiency that is sufficient for learning-based methods, while simultaneously preserving enough physical realism to support sim-to-real transfer.

While our approach is general, we focus our study on the easily manufacturable PneuNet actuator [40], which has previously been used to create robots capable of walking and crawling gaits [18, 52]. We experimentally validate our proposed approach by learning combinations of PneuNets and their controllers that together lead to faster gaits, and demonstrate the ability to successfully transfer optimized design-control pairs to reality.

Our work contributes a complete simulation and optimization framework for the joint design and control of soft robots capable of sim-to-real transfer. Specifically, this includes:

1) a model-free algorithm for optimizing the blended design and control spaces of soft robots;
2) a framework for creating reconfigurable reduced-order soft robot models that improve computational efficiency and enable the use of learning techniques;
3) the discovery of pneumatically actuated soft robots that outperform a standard expert-designed crawling robot in simulation and reality.

See our webpage[1] for code and videos of our results.

II. Related Work

The problem of jointly optimizing a rigid robot's physical structure along with its control has a long history in robotics research. Early work employs evolutionary methods to optimize the robot's design along with its (often neural) controller [34, 44, 41, 4]. Another common approach is to assume access to a parameterized model of the robot's dynamics and to then optimize these parameters together with those of control (or motion) [45, 61, 22, 55, 19, 58, 5]. Bolstered by the availability of efficient high-fidelity physical simulators, joint optimization methods based on reinforcement learning are able to learn capable rigid-body design-controller pairs without prior knowledge of the dynamics [50, 43, 21, 63]. Because these methods are trained in simulation, the sim-to-real transfer of the learned solutions must be considered. Our work focuses on the use of high-fidelity simulators, which have been shown to facilitate transfer for rigid robots [57]. Unlike rigid-body domains, however, achieving the level of fidelity necessary for soft-bodied robots typically requires computationally demanding simulators that prohibit learning-based co-optimization, a challenge that we address here.

[1] https://sites.google.com/ttic.edu/evolving-soft-robots

Compared to rigid robotics, jointly optimizing the design and control of soft robots is less explored. Of the work that exists, the large majority focus exclusively on simulation. Many approaches reason over design and control spaces that include a mix of discrete and continuous parameters (e.g., voxel-based soft robots (VSRs) [56] are composed of discrete voxels, but the input frequency to each voxel is considered to be continuous). Spielberg et al. [54] propose an autoencoder-based method that is able to optimize the placement of a large number of such voxels for simulated locomotion tasks with fewer iterations than other approaches. Cheney et al. [9] use an evolutionary neural strategy to develop designs for VSRs that locomote in simulation. Kriegman et al. [31] describe an approach to deforming the structure of VSRs subject to damage such that the original control policy remains valid. Ma et al. [37] use a material point method-based simulation and gradient-based optimization methods to co-optimize the shape and control of simulated swimming robots. Deimel et al. [13] use particle filter-based optimization to co-optimize finger angles and the grasp strategy of a soft gripper. The success of these methods in simulation is encouraging for soft roboticists, and recent simulation-based benchmarks allow for a rigorous comparison of co-optimization methods [11, 3]. However, existing work provides a limited evaluation of the physical design-control pairs, and so little is known about their ability to transfer to the real world. Indeed, experiments on voxel-based soft robots reveal that their behavior in simulation can differ significantly from reality [30].

One notable exception, Morzadec et al. [39] experimentally verify an optimized soft robotic joint, showing how shape optimized using a finite element analysis-based simulator [10] translates to improvements in a real-world soft robotic leg, however unlike our work they do not optimize the controller. Another exception is recent work that integrates a pneumatic-based passive controller into the robot's design to achieve a forward walking gait [15], providing an example of how soft robots can have unclear boundaries between design and control.

Meanwhile, individual design and control methods continue to be key areas of research in soft robotics [49]. There exist a wide variety of design concepts for soft robots [7] such as fluidically pressurizeable soft devices [40, 52], metamaterial-based designs [48, 35], and cable-driven devices [2]. Soft roboticists note that existing design optimization methods for compliant, nonlinear mechanisms, such as topology optimization, are challenging to use in soft robotics due to complicated soft material behavior [7]. The diversity of the design space for soft robots further exacerbates the challenge of automating the search for optimal designs [46]. Model-[1, 6] and learning-based [33, 12, 29] controllers have also proven successful, as well as hybrid policy designs [62, 2, 26]. Zhu et al. [65] consider an origami-like robot with various design configurations that all inform policy optimization, and Morimoto et al. [38] employ the soft actor-critic algorithm [23] for reaching tasks. Related, Vikas et al. [60] present a modular approach to designing 3D-printed motor-tendon soft robots that can be readily fabricated, and a model-free algorithm for learning the corresponding

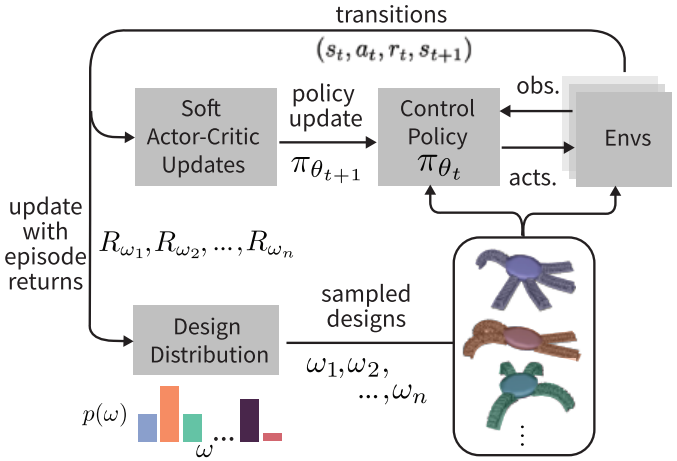


Fig. 2. Our approach maintains a distribution over designs $p(\omega)$. At each iteration, the method samples a set of designs $\omega_1, \ldots, \omega_n$ and controls each using a shared, design-conditioned, policy π_θ. We train the policy using soft actor-critic on a mixture of data from different designs, and update the design distribution based on the episode returns of the sampled designs.

control policy. Unlike our framework, however, they do not jointly reason over design and control.

III. Co-Optimization of Design and Control

We first describe the general approach to jointly optimizing robot design and control, and then discuss a specific application to crawling soft robots. Algorithm 1 and Figure 2 give an overview of this approach.

A. General Approach via Multi-task Reinforcement Learning

We model the control problem as a Markov decision process (MDP) $\mathcal{M} = \text{MDP}(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition dynamics, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function. When co-optimizing design and control, we additionally define the design space Ω. Assuming we are optimizing for a single task specified by its reward $\mathcal{R}$, we define the design-specific MDP $\mathcal{M}_\omega = \text{MDP}(\mathcal{S}_\omega, \mathcal{A}_\omega, \mathcal{P}_\omega, \mathcal{R})$ for each design $\omega \in \Omega$. In most co-optimization settings with a single task, the state and action spaces will change between designs only when those designs have different morphologies.

Let $\pi^*_\omega : \mathcal{S}_\omega \times \mathcal{A}_\omega \to [0, 1]$ be the optimal policy for MDP $\mathcal{M}_\omega$. The goal of co-optimization is to find the optimal design and controller pair $(\omega^*, \pi^*_{\omega^*})$ such that:

$$\omega^*, \pi^*_{\omega^*} = \arg\max_{\omega, \pi_\omega} \mathbb{E}_{\pi_\omega}\left[\sum_t \gamma^t \mathcal{R}_t\right] \tag{1}$$

In this setting, we are faced with many MDPs that share common structure. Solving each MDP independently is intractable and ignores these similarities, which are critical to generalizing to new designs. We draw on insights from multi-task reinforcement learning [59] to more efficiently solve for the optimal design-control pairs (Eqn. 1) by exploiting this common structure. Similar to goal-conditioned policies, our approach learns a single *design-conditioned* policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \times \Omega \to [0, 1]$ to control all the designs in Ω for the specified task. We proposed this idea in the context of co-optimization [50], and it has also been used for the sub-problem of controlling a set of designs with different morphologies [28, 32]. This policy can be trained using any RL algorithm on a mixture of data collected with designs in Ω.

Algorithm 1: Joint Optimization of Design and Control

Initialize $\pi_\theta(a|s, \omega)$, $p(\omega)$, $T = 0$
while $T < BUDGET$ **do**
  Sample designs $\omega_1, \omega_2, \ldots, \omega_n \sim p_\phi$
  Control $\omega_1, \omega_2, \ldots, \omega_n$ with π_θ for t timesteps. Add transitions to replay buffer.
  Update θ using soft actor-critic.
  Update $R_{\omega_1}, R_{\omega_2}, \ldots, R_{\omega_n}$ with their obtained returns.
  Set timestep $T = T + nt$
  Set β_T to match entropy target $\mathcal{H}_T$.
  Set $p(\omega) = \frac{e^{\beta_T R_\omega}}{\sum_\Omega e^{\beta_T R_\omega}}$
end while

In order to search over designs, we maintain a distribution $p(\omega)$ over the design space Ω. This distribution generates designs for training the controller and models the belief about which designs are optimal given the current design-conditioned control policy. At the start of training, $p(\omega)$ should provide a large diversity of designs and then, once the controller has been sufficiently trained, slowly concentrate probability mass around high-performing designs. The controller can then specialize to an increasingly promising subset of designs until the algorithm converges on a single design and a controller that is then fine-tuned for that design.

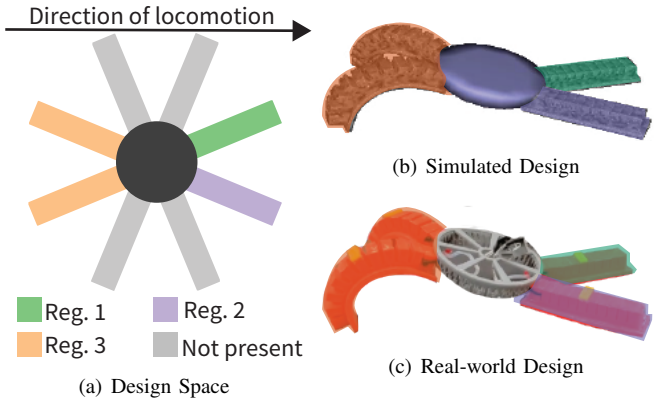
The design distribution can be modeled in a number of different ways depending on the nature of the design space. For example, Schaff et al. [50] use a mixture of Gaussians for a continuous design space, and shift the distribution towards high-performing designs in a manner analogous to a policy gradient update. In this work, we assume that the design space is discrete and that the number of designs is practically enumerable, and thus employ a categorical distribution. Following the principle of maximum entropy, we model $p(\omega)$ as a Gibbs distribution:

$$p(\omega) = \frac{e^{\beta R_\omega}}{\sum_{\omega \in \Omega} e^{\beta R_\omega}}, \tag{2}$$

where R_ω is the most recent reward obtained by design ω, and β is an inverse temperature parameter used to control entropy. At each point in training, we set β to maintain a decaying entropy target. Specifically, we set $\beta = 0$ to specify a uniform distribution for an initial training period, and then decay entropy according to a linear schedule. This schedule is akin to removing a constant fraction of designs from the search space at each step during training.

B. Application to Soft, Legged Robots

The design space that we study here (Fig. 3) consists of a disk with N equally spaced positions where soft, pneumatic



(a) Design Space (b) Simulated Design (c) Real-world Design

Fig. 3. A visualization of (a) our design space that consists of a disk with $N = 8$ candidate locations for pneumatic actuators, each of which can be connected to one of $M = 3$ pressure regulators. On the right are examples of a (b) simulated and (c) real-world design, where colors denote the pressure regulator for each actuator. The forward direction is to the right.

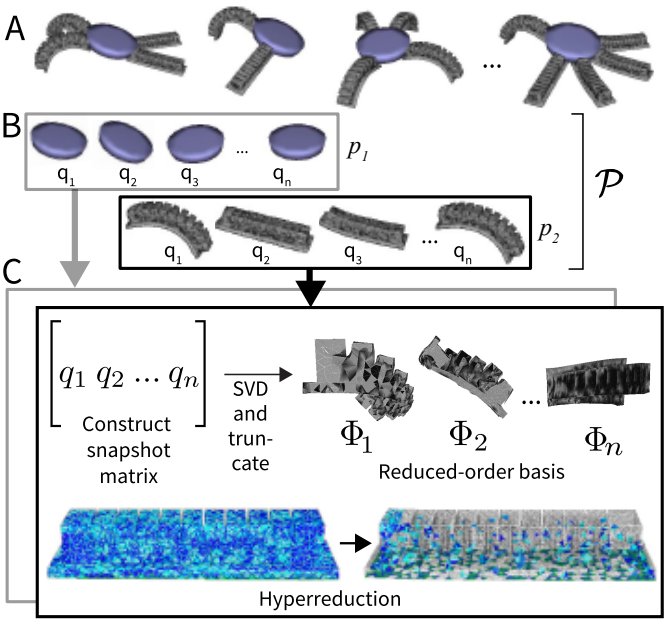


Fig. 4. Our proposed technique for model order reduction that is compatible with co-optimization. **A**: Sample candidate designs and record their animations. **B**: Collect parts across designs and transform them into a common reference frame. **C**: For each part, perform a snapshot POD reduction and hyperreduction to obtain a reduced-order basis for motion, and reduced integration domain. We create reductions for new designs by combining reductions of their parts.

actuators can be positioned radially outward. Each actuator is connected to one of M different pressure regulators. Designing the robot then amounts to choosing whether (or not) to place an actuator at each of the N locations and, for each placed actuator, connecting it to one of the M pressure regulators. Our specific implementation considers $N = 8$ candidate locations and $M = 3$ regulators, and restricts the design to having between three and six actuators. This results in a total of 41202 unique designs, which can be reduced to 6972 by exploiting symmetry in the regulator assignments. Each actuator is a PneuNet [40] which, like similar soft actuators, has been combined to achieve crawling gaits [20, 52, 18, 60], providing a well-studied baseline.

While our approach is compatible with any RL algorithm, we use the standard soft actor-critic (SAC) algorithm because it offers stable and data-efficient learning dynamics. We train an open-loop controller modeled as a feed-forward neural network for the task. This simplification of the controller forces the design to perform "morphological computation" [49] to enable intelligent behavior. The policy takes as input the design parameters along with the four most-recent actions and outputs pressure targets for each regulator.

IV. Design-Reconfigurable Model Order Reduction

Integral to our approach, we propose a modular form of model order reduction (MOR) suited to co-optimization that significantly improves the efficiency of FEA-based simulators while preserving their accuracy, supporting sim-to-real transfer.

A. Reduction through Snapshot POD and Hyperreduction

The finite element method provides an approximate numerical solution to partial differential equations (PDEs) by discretizing space into a mesh consisting of a set of finite elements. Often, dense meshes (and subsequently, large amounts of computation) are needed to reach acceptable accuracy.

The soft actuator mesh contains nodes with position q_{t_n} and velocity v_{t_n} at discrete time step t_n. At each t_n, simulation requires solving a discrete form of Newton's second law [20]:

$$A(q_{t_n}, v_{t_n})dv = b(q_{t_n}, v_{t_n}) + H^\top \lambda, \tag{3}$$

where $dv = v_{t_{n+1}} - v_{t_n}$, $A \in \mathbb{R}^{d\times d}$ collects inertial and internal forces, $b \in \mathbb{R}^d$ contains terms from internal and external forces, and $H^T\lambda \in \mathbb{R}^d$ collects constraints (e.g., associated with contact with the floor), with d being the number of degrees-of-freedom in the mesh. When using dense meshes for accurate simulation, constructing the matrix A and solving this system of equations are often the main bottleneck in FEA simulations.

We first reduce the system dimension through snapshot proper orthogonal decomposition (POD). Using the methods of Goury et al. [20], we find a low-dimensional subspace Φ that well-approximates the space of possible motions and deformations while reducing the order through a Galerkin projection onto Equation 3:

$$\Phi^\top A(q_{t_n}, v_{t_n})\Phi d\alpha = \Phi^\top b(q_{t_n}, v_{t_n}) + \Phi^T H^\top \lambda. \tag{4}$$

We achieve this by recording "snapshots" of the position q_t of the mesh throughout a series of predefined motions that try to cover the space of common deformations. The simulation then uses lower-dimensional coordinates α, with $q_{t_n} = q_0 + \Phi\alpha_{t_n}$.

Though snapshot POD reduces the time to solve Equation 3, it still requires computing the high-dimensional matrix A at every time step. We therefore perform a hyperreduction to further approximate A by predicting its entries from the contributions of a small number of elements. We use the hyperreduction method of energy conservation sampling and

weighting (ECSW) [16]. For further details regarding this two-part method and a demonstration in soft robotics, we refer the reader to Goury et al. [20].

B. Modularized Reduction for Design-Reconfigurability

The MOR method described in the previous section uses a single fixed mesh. For high-dimensional soft robot design spaces, separately reducing each design is computationally intractable. Instead, we define a modularized design space: each design $\omega \in \Omega$ is defined as a combination of a small set of fixed parts $\mathcal{P}$. We then reduce each part in $\mathcal{P}$ using the method of Section IV-A independently and combine the parts in arbitrary ways to form new designs. The number of times we perform MOR is then of the same size as $\mathcal{P}$ rather than the size of Ω.

MOR on the modularized design space only well-approximates the full-order model when the computed subspace Φ^p for each part $p \in \mathcal{P}$ is close to all frequently achieved deformations. Because designs will deform in different ways, it is necessary to include 'snapshots' of motions from a large set of designs to achieve high-quality reduced-order models. Therefore, careful snapshot selection on each module in $\mathcal{P}$ is crucial and inaccuracies may be exploited during optimization, resulting in invalid design-control pairs. We achieve high-quality reduced-order models for each part by collecting snapshots from a heuristically chosen subset of Ω and animating those designs by cycling through the pressure extremes of each actuator.

When constructing the reduced basis for new designs, we transform the basis Φ_p to match the initial pose (t_i, R_i) of each part p_i by rigidly rotating the node positions that make up each basis vector:

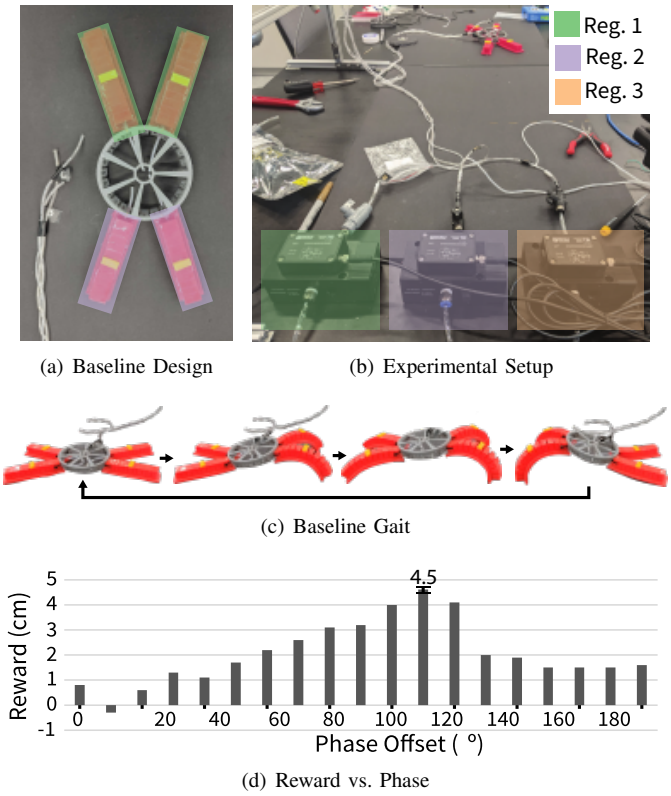
$$\Phi_j^{p_i} = \begin{bmatrix} R_i \Phi_j^p[0:3] & R_i \Phi_j^p[3:6] \cdots R_i \Phi_j^p[n-3:n] \end{bmatrix}, \quad (5)$$

where $\Phi_j^p \in \mathbb{R}^n$ is the j^{th} column of Φ^p and $\Phi_j^p[k:l]$ is a slice of that vector from index k to index l. We ignore the initial translation t_i because translation basis vectors are included in Φ_p. Figure 4 gives an overview of this approach.

C. Reduction of Crawling Soft Robots

We apply this reduction technique to our design space of crawling soft robots. Our designs are composed of two parts: the central disk, and some number of identical PneuNets. Therefore, the above approach allows us to perform two reductions (one for each part) as opposed to reducing each of the 6972 designs in our design space. We found that a sparse disk mesh was sufficiently fast and accurate for simulation and we therefore only reduce the PneuNet.

For the reduction of the PneuNet, we select a heuristic set of 256 designs for which we collect snapshots. Each design contains a unique subset of the eight potential PneuNet positions, and each PneuNet is controlled independently. Similar to Goury et al. [20], we iterate through the extremes of each actuator and record snapshots at fixed time intervals. This can be seen as a walk through the vertices of an n-dimensional hypercube, where n is the number of PneuNets present. In order to verify the accuracy of our reduction, we evaluate it on a set of four test designs and animations by computing the distance between the node positions of the reduced and unreduced models. We perform a grid search over the two tolerances in the reduction algorithm and select the reduction that has the best time-accuracy trade off. See Appendix B for more details.



(a) Baseline Design (b) Experimental Setup

(c) Baseline Gait

(d) Reward vs. Phase

Fig. 5. Our baseline consists of (a) an expert-designed soft robot with four legs, where the fore and hind legs are attached to pressure regulators one and two, respectively. The experimental setup consists of (b) three pressure regulators, a crawling surface, and 3.15 mm outer-diameter tubing connected to the robot (in the the distance). (c) Snapshots of the expert-designed gait. (d) Reward (distance traveled) obtained by an offset-sine gait with different phase shifts. A phase difference of 110° achieves the highest reward.

V. Experiments

We test our approach by attempting to find a design and open-loop controller that crawl as far as possible on a flat plane in a 20 second episode. We define reward as the distance traveled in the (forward) x-direction (in cm) as measured at the center of the disk.

After performing model order reduction, we carry out FEA simulation using the SOFA Finite Element framework [17] with the soft robotics [10] and model order reduction [20] plugins. We model the PneuNet legs (including the inflatable and constraint material) and central disk as linear elastic materials. We estimated the Young's modulus of the PneuNet material (Smooth-On DragonSkin 30) based on the published Shore hardness together with the method of Qi et al. [47]. We modeled the constraint layer of the PneuNet as being linear elastic

with a Young's modulus twice the magnitude of the inflatable material. We tuned the Poisson's ratio in order to maintain numeric convergence and qualitative realism. We used Coulom friction as the friction model. To account for any inaccurate or unmodelled effects, we measure deformation of a single, real PneuNet under fixed pressures, find the corresponding pressures that results in the same deformation of the simulated PneuNet, and fit an affine function to this data. Pressures commanded by our learned policies are then mapped through this function to ensure a simulated response similar to that of the real PneuNets. We find that this step facilitates sim-to-real transfer. For details about the fabrication of our robots, see Appendix C.

We designed a baseline design-controller pair similar to the robot used by Shepherd et al. [52]. The baseline (Fig. 5) has two fore legs and two hind legs placed $45°$ apart with each pair controlled by a single regulator. Based on recent analysis of inching gaits [18], we constrain each pressure regulator to produce a sine wave of equal amplitude and period. We achieve forward motion by imposing a phase shift between the sine waves for the fore and hind legs. We select a pressure range of 0 to 90 kPa to avoid both physical instabilities (i.e., aneurysms of the PneuNets) and numerical instabilities in the FEA simulator. We use the maximum amplitude allowed in this range of 45 kPa and choose a period of 4 sec, which is the fastest period that led to stable motion. The optimal phase shift depends heavily on friction [18] so we conducted experiments with different phase shifts between $0°$ and $180°$ in increments of $10°$, and chose the value that resulted in the highest reward. Figure 5 shows the effect of phase shift on the reward.

We use 96 parallel environments for data collection. Each environment contains a design sampled from $p(\omega)$ (Algorithm 1, line 3) that is controlled with the current policy for one episode (Algorithm 1, line 4). The control policy is then updated using the soft actor-critic algorithm on data from a replay buffer (Algorithm 1, line 5). We repeat this process for 1M environment timesteps during which we fix the design distribution to be uniform for the first 200K timesteps, after which we linearly decay the entropy to zero at 1M timesteps. See Appendix A for more details on the experimental setup, including a full list of simulation and learning parameters. After training, we manufacture several of the highest-performing designs and conduct a series of experiments to evaluate the sim-to-real transfer of the learned design-control pairs.

VI. Results

We examined the performance and capacity for sim-to-real transfer of the top-five learned design-control pairs that our framework discovers along with our baseline design-control pair. We refer to the baseline as "B" and the learned pairs as "L1–L5" in order of decreasing reward. Figure 6 visualizes these designs and compares their reward to that of the baseline in simulation and in the real world (tested over five trials). The top four learned robots (L1–L4) outperform the baseline in both simulation and the real world, while the fifth robot (L5) performs comparably to the baseline in the physical

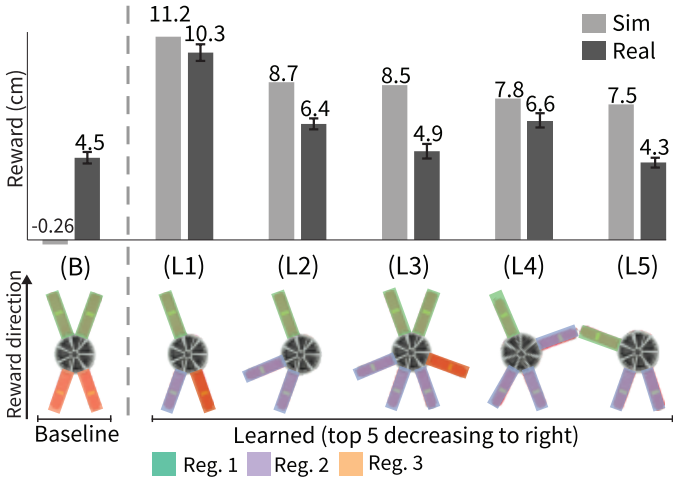


Fig. 6. A comparison of the reward (distance traveled in cm) achieved in simulation and reality for the duration of the 20 sec episode for the baseline robot (B) and the top-five learned design-control pairs (L1–L5).

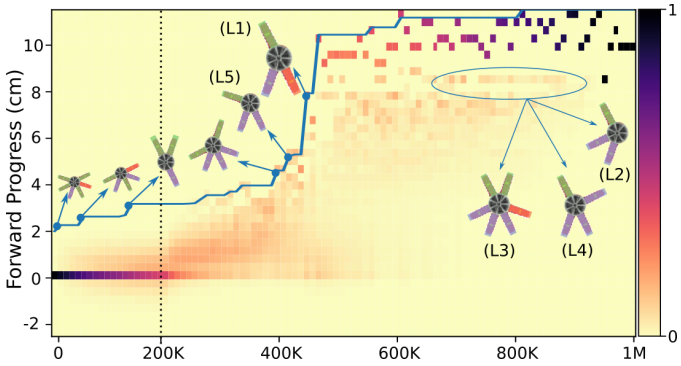


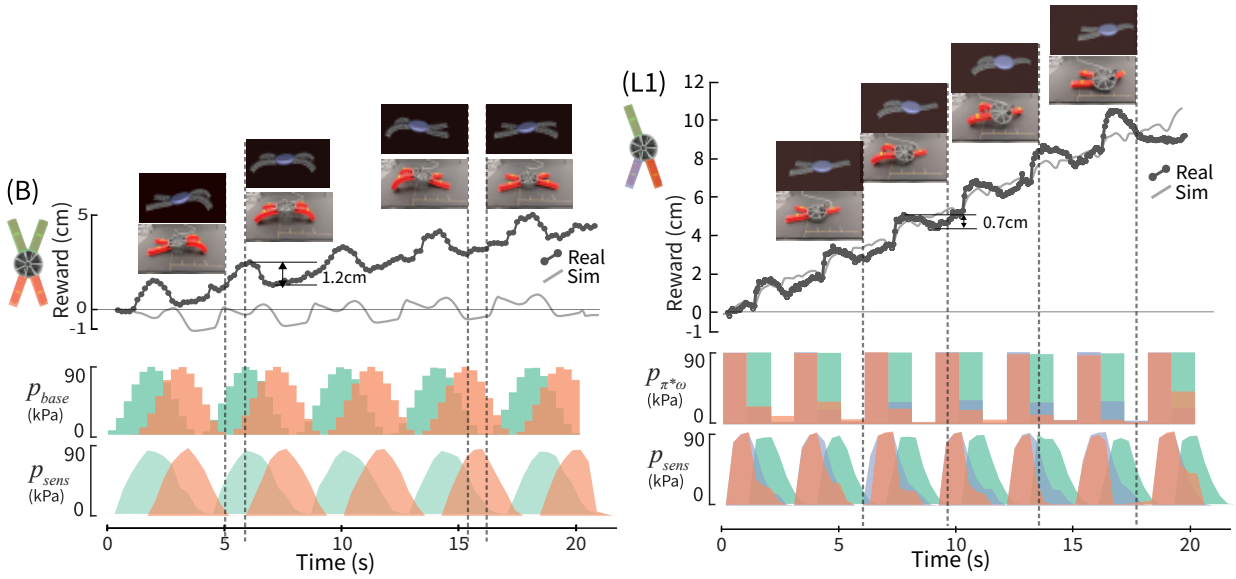
Fig. 7. A histogram of rewards throughout training, weighted by the design distribution. The blue line represents the highest reward achieved at each point throughout training. The designs pictured on the left show important mode switches, and the right shows other high-performing designs. The design distribution is fixed to be uniform until 200K timesteps and then a temperature parameter is tuned to match a linearly decaying entropy target.

experiments. The top learned robot, L1 outperforms the baseline (in the real world) by a factor of 2.3.

Figure 7 inspects the training dynamics of our approach; it shows a histogram of rewards achieved by the design distribution throughout training. In the beginning of training, the design distribution is constrained to be uniform and nearly every design achieves zero reward. Starting at 200K timesteps, the algorithm constrains the distribution with a linearly decaying entropy, after which the algorithm specializes to high performing designs (i.e., L5). Approximately halfway through training, the algorithm converges on design L1, which achieves a reward that is several centimeters better than the next-best design-control pair.

A. Qualitative Analysis of Learned Designs and Gaits

Observing the learned design-control pairs in simulation and through real-world experiments reveals that they exploit changes in frictional forces in clever ways to create forward motion. Figure 8 compares the open-loop (pressure) gaits and



(a) Baseline Policy

(b) Learned Policy with Highest Reward

Fig. 8. A comparison between the (a) baseline (B) and (b) highest-performing learned robot (L1) in terms of the reward (distance traveled in cm) achieved in simulation and reality for the duration of the 20 sec episode, along with the corresponding control policy in terms of commanded and sensed pressures.

reward trajectories (in simulation and the real-world) over the duration of the 20 sec episode for the baseline and top learned design-control pairs. The baseline robot (B) uses a symmetric design that moves forward through out-of-phase actuation of the fore and hind legs (Fig. 8(a)). In contrast, robot L1 uses an asymmetric design alongside out-of-phase actuation of the three attached pressure regulators. The result of this design and motion is a pivoting behavior visible in the photos within Figure 8(b) and the supplementary video. Due to its asymmetry, the soft robot rolls to its side and the contact area of the front leg with the floor is reduced (leading to reduced frictional forces on that leg). This reduction in contact area enables the front PneuNet 'leg' to slip forward instead of pushing backward. The result is that robot L1 uses forward-slipping motion without as much backward displacement ("backsliding") per step (0.7 cm backsliding) compared to the baseline (1.2 cm backsliding).

Figures 9 and 10 show the reward per timestep and pressurization (commanded and sensed) for robots L2–L5. Four of the five top robots (L1, L2, L4, and L5) also leverage asymmetric morphologies (Fig. 6) to tilt the front leg onto its corner or edge and reduce friction by reducing contact. Gaits for the L2–L5 designs (Fig. 10) operate in similar stages to L1: the hind legs inflate first to anchor the robot and subsequently follow a cycle that is out of-phase with the cycle followed by the front legs. While the morphology of robot L3 is symmetric, the use of regulator three on only one leg adds asymmetry to its gait. Interestingly, each of the learned gaits inflate the hind legs before the fore legs, while the baseline does the opposite. We find that the learned gaits, especially L1, L2, and L5, make the most forward progress at the transition between inflation of the fore legs and deflation of the hind legs (e.g., overlap of green and orange sensed pressures in Figure 8(b)). We hypothesize that the asymmetry of the learned designs makes this motion possible. See our webpage[2] for side-by-side recordings of the learned designs and gaits in simulation and reality.

B. Sim-to-Real Transfer

Given the goal of being able to co-optimize the design and control of physically realizable soft robots, we compare the learned and baseline robots through physical experiments. We measure sim-to-real transfer by manufacturing the learned designs and applying the learned policy without modification.

We find that all of optimized designs and gaits make consistent forward progress, with four out of five (L1–4) outperforming the baseline in physical measurements, and one performing comparably (L5) to the baseline. The highest-performing robot (L1) and robot L4 have strong agreement with simulation—with the standard deviation across trials taken into account, real reward is within 1 cm of the simulated reward.

Despite strong qualitative shape agreement seen in the supplementary video, L2, L3, and L5 see a pronounced drop in performance. A comparison of the simulated and real reward over time shows the main reason for this performance decrease. Figure 9 reveals that while these reward functions have qualitatively similar waveforms, the backsliding phases of the waveforms appear to have higher magnitude in reality than in simulation. The initial phase of each of these gaits inflates the hind legs, relying on the friction from the fore leg(s) to hold the robot in place rather than slide backwards. The front PneuNets exhibit little slip in simulation, but they maintain less grip in the real world. The resulting error in the backsliding phase of each of these gaits is then accumulated with every step, resulting in less forward motion.

[2]https://sites.google.com/ttic.edu/evolving-soft-robots

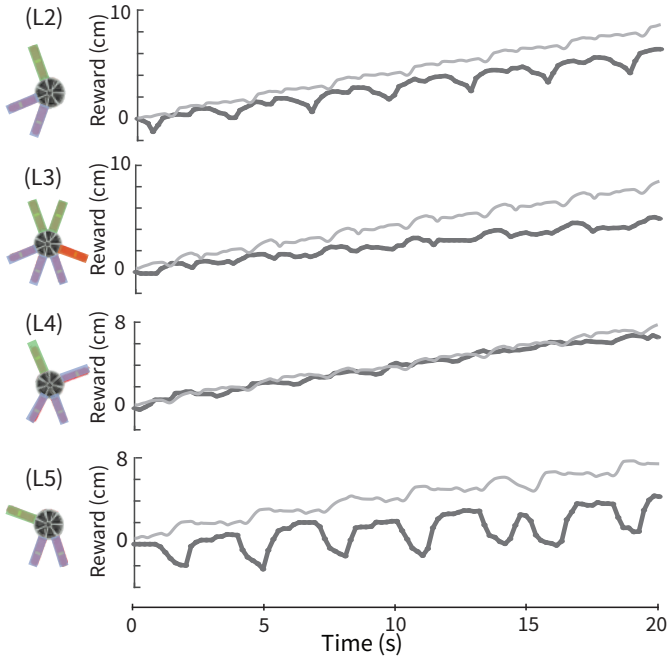


Fig. 9. Reward per timestep achieved in simulation and reality by the second- to fifth-ranking learned design-controller pairs L2–L5.

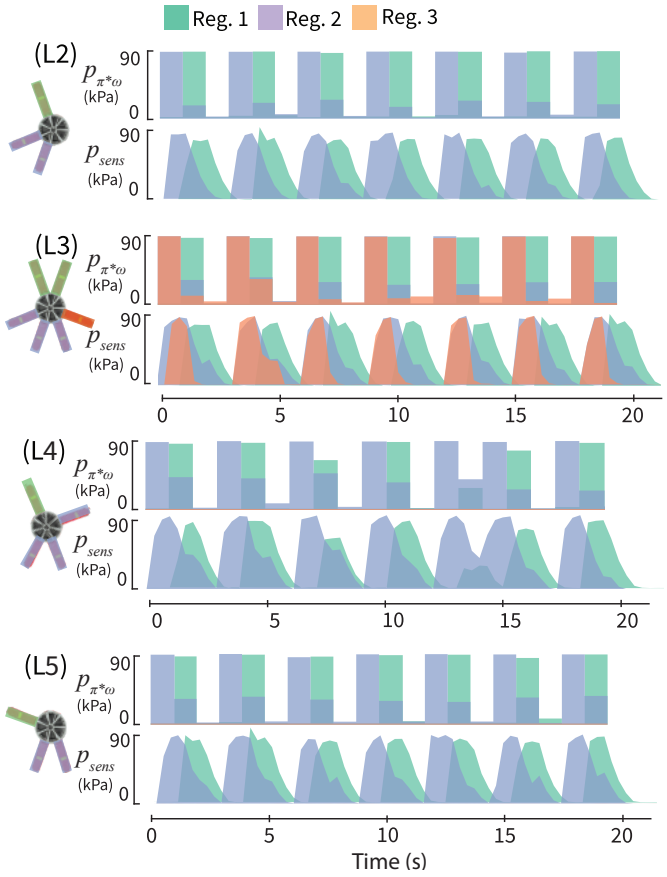


Fig. 10. Commanded and sensed pressures $p_{\pi*\omega}$ and p_{sens} for the second- to fifth-ranking learned design-controller pairs L2–L5.

Figure 8(a) compares the forward progress of the simulated baseline against its real-world counterpart. Despite detailed tuning of simulation parameters, our baseline gait achieves poor reward in simulation. The simulated baseline gait does indeed make slow forward progress. Yet, in key segments of the gait (shown by images in Fig. 8(a)), the simulation records less progress, or backward progress, compared to the real-world. As shown by prior analysis of crawling robots [18, 60], this type of sinusoidal gait is very sensitive to frictional forces. Because the design is symmetric, the gait relies on the subtle differences in friction to enable timing of stick-slip interactions for forward motion. As a result, error accrual due to backward sliding that was already evident in the learned designs is of an even higher magnitude here.

While there are many other differences between simulation and reality, such as dynamics associated with the pressure regulators and damping of high frequency motions, we find that the modelling errors associated with stick-slip transitions has the largest effect on the transfer performance of our designs. Stick-slip transitions are notoriously difficult to model [36], and often require smoothness approximations for numeric stability. Overall we still see mostly successful transfer for all five optimized designs.

VII. Conclusion

This work describes a complete framework for the simulation and co-optimization of the design and control of soft robots capable of sim-to-real transfer. We present an algorithm for co-optimization and a framework to create reconfigurable, reduced-order models for soft robotics. Experiments demonstrate that our framework learns design-gait pairs that outperform an expert-designed baseline in a soft robot locomotion task. We further characterize the successful qualitative and quantitative transfer of these learned pairs from simulation to reality.

Soft robots are compliant-bodied and mechanically intelligent. As a result, their design and control spaces are not well-separated. For many tasks, it is difficult if not impossible to explore these design-control spaces through prototyping or analysis alone—simulation is needed to build and evaluate designs and controllers in a tractable manner. Our work takes a step in this direction by showing that it is possible to combine reinforcement learning techniques with finite element simulation to deliver fast and physically accurate co-optimization for soft robotics.

This work has some limitations. Modelling errors, due to friction, linear elasticity, etc., caused us to linearly warp simulation parameters to obtain numerical stability and realism. Even so, these errors led to a degradation of transfer performance for some designs. Future work will include improved simulation techniques as well as investigations into domain adaptation and domain randomization methods as a means of improving sim-to-real transfer. Given the success of this method for simple locomotion, we plan to explore adaptations of this framework to different design and control spaces as well as more complex tasks such as manipulation.

ACKNOWLEDGMENTS

We thank Olivier Goury and Hugo Talbot for assistance implementing SOFA and the MOR module, and Arthur MacKeith for additional simulation support.

REFERENCES

[1] J. M. Bern, P. Banzet, R. Poranne, and S. Coros, "Trajectory optimization for cable-driven soft robot locomotion," in *Proceedings of Robotics: Science and Systems (RSS)*, 2019.

[2] J. M. Bern, Y. Schnider, P. Banzet, N. Kumar, and S. Coros, "Soft robot control with a learned differentiable model," in *Proceedings of the IEEE International Conference on Soft Robotics (RoboSoft)*, 2020.

[3] J. Bhatia, H. Jackson, Y. Tian, J. Xu, and W. Matusik, "Evolution gym: A large-scale benchmark for evolving soft robots," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[4] J. Bongard, "Morphological change in machines accelerates the evolution of robust behavior," *Proceedings of the National Academy of Sciences*, vol. 108, no. 4, pp. 1234–1239, 2011.

[5] G. Bravo-Palacios, A. Del Prete, and P. M. Wensing, "One robot for many tasks: Versatile co-design through stochastic programming," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1680–1687, 2020.

[6] D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan, "Modeling and control of soft robots using the Koopman operator and model predictive control," in *Proceedings of Robotics: Science and Systems (RSS)*, 2019.

[7] F. Chen and M. Y. Wang, "Design optimization of soft robots: A review of the state of the art," *IEEE Robotics & Automation Magazine*, 2020.

[8] T. Chen, Z. He, and M. Ciocarlie, "Hardware as policy: Mechanical and computational co-optimization using deep reinforcement learning," *arXiv:2008.04460*, 2020.

[9] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, "Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding," in *Proceedings of the Annual Conference on Genetic and Evolutionary Computation (GECCO)*, 2013.

[10] 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, 2017.

[11] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access*, 2021.

[12] U. Culha, S. O. Demir, S. Trimpe, and M. Sitti, "Learning of sub-optimal gait controllers for magnetic walking soft millirobots," in *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[13] R. Deimel, P. Irmisch, V. Wall, and O. Brock, "Automated co-design of soft hand morphology and control strategy for grasping," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[14] K. M. Digumarti, C. Gehring, S. Coros, J. Hwangbo, and R. Siegwart, "Concurrent optimization of mechanical design and locomotion control of a legged robot," in *Mobile Service Robotics*, 2014, pp. 315–323.

[15] D. Drotman, S. Jadhav, D. Sharp, C. Chan, and M. T. Tolley, "Electronics-free pneumatic circuits for controlling soft-legged robots," *Science Robotics*, vol. 6, no. 51, 2021.

[16] C. Farhat, P. Avery, T. Chapman, and J. Cortial, "Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency," *International Journal for Numerical Methods in Engineering*, vol. 98, no. 9, pp. 625–662, 2014.

[17] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, et al., "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, 2012, pp. 283–321.

[18] B. Gamus, L. Salem, A. D. Gat, and Y. Or, "Understanding inchworm crawling for soft-robotics," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1397–1404, 2020.

[19] M. Geilinger, R. Poranne, R. Desai, B. Thomaszewski, and S. Coros, "Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, 2018.

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

[21] D. Ha, "Reinforcement learning for improving agent design," *Artificial Life*, vol. 25, no. 4, pp. 352–365, 2019.

[22] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Joint optimization of robot design and motion parameters using the implicit function theorem," in *Proceedings of Robotics: Science and Systems (RSS)*, 2017.

[23] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[24] J. Hiller and H. Lipson, "Dynamic simulation of soft multimaterial 3D-printed objects," *Soft Robotics*, vol. 1, pp. 88–101, 2014.

[25] D. P. Holland, E. J. Park, P. Polygerinos, G. J. Bennett, and C. J. Walsh, "The soft robotics toolkit: Shared

resources for research and design," *Soft Robotics*, vol. 1, no. 3, pp. 224–230, 2014.

[26] T. Howison, S. Hauser, J. Hughes, and F. Iida, "Reality-assisted evolution of soft robots through large-scale physical experimentation: A review," *arXiv:2009.13960*, 2020.

[27] 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 *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[28] W. Huang, I. Mordatch, and D. Pathak, "One policy to control them all: Shared modular policies for agent-agnostic control," *arXiv:2007.04976*, 2020.

[29] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee, et al., "Review of machine learning methods in soft robotics," *PLOS ONE*, vol. 16, no. 2, 2021.

[30] S. Kriegman, A. M. Nasab, D. Shah, H. Steele, G. Branin, M. Levin, J. Bongard, and R. Kramer-Bottiglio, "Scalable sim-to-real transfer of soft robot designs," in *Proceedings of the IEEE International Conference on Soft Robotics (RoboSoft)*, 2020.

[31] S. Kriegman, S. Walker, D. Shah, M. Levin, R. Kramer-Bottiglio, and J. Bongard, "Automated shapeshifting for function recovery in damaged robots," in *Proceedings of Robotics: Science and Systems (RSS)*, 2019.

[32] V. Kurin, M. Igl, T. Rocktäschel, W. Boehmer, and S. Whiteson, "My body is a cage: The role of morphology in graph-based incompatible control," *arXiv:2010.01856*, 2021.

[33] K. Lee, S. Kim, S. Lim, S. Choi, M. Hong, J. Kim, Y.-L. Park, and S. Oh, "Generalized Tsallis entropy reinforcement learning and its application to soft mobile robots," in *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[34] H. Lipson and J. B. Pollack, "Automatic design and manufacture of robotic lifeforms," *Nature*, vol. 406, pp. 974–978, 2000.

[35] J. I. Lipton, R. MacCurdy, Z. Manchester, L. Chin, D. Cellucci, and D. Rus, "Handedness in shearing auxetics creates rigid and compliant structures," *Science*, vol. 360, no. 6389, pp. 632–635, 2018.

[36] J. Luo and K. Hauser, "Robust trajectory optimization under frictional contact with iterative learning," in *Proceedings of Robotics: Science and Systems (RSS)*, 2015.

[37] P. Ma, T. Du, J. Z. Zhang, K. Wu, A. Spielberg, R. K. Katzschmann, and W. Matusik, "DiffAqua: A differentiable computational design pipeline for soft underwater swimmers with shape interpolation," *arXiv preprint arXiv:2104.00837*, 2021.

[38] R. Morimoto, S. Nishikawa, R. Niiyama, and Y. Kuniyoshi, "Model-free reinforcement learning with ensemble for a soft continuum robot arm," in *Proceedings of the IEEE International Conference on Soft Robotics (RoboSoft)*, 2021.

[39] T. Morzadec, D. Marcha, and C. Duriez, "Toward shape optimization of soft robots," in *Proceedings of the IEEE International Conference on Soft Robotics (RoboSoft)*, 2019.

[40] B. Mosadegh, P. Polygerinos, C. Keplinger, S. Wennstedt, R. F. Shepherd, U. Gupta, J. Shim, K. Bertoldi, C. J. Walsh, and G. M. Whitesides, "Pneumatic networks for soft robotics that actuate rapidly," *Advanced Functional Materials*, vol. 24, no. 15, pp. 2163–2170, 2014.

[41] S. Murata and H. Kurokawa, "Self-reconfigurable robots," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 71–78, 2007.

[42] J.-H. Park and H. Asada, "Concurrent design optimization of mechanical structure and control for high speed robots," *Journal of Dynamic Systems, Measurement, and Control*, vol. 116, no. 3, pp. 344–356, 1994.

[43] D. Pathak, C. Lu, T. Darrell, P. Isola, and A. A. Efros, "Learning to control self-assembling morphologies: A study of generalization via modularity," *arXiv:1902.05546*, 2019.

[44] C. Paul and J. C. Bongard, "The road less travelled: Morphology in the optimization of biped robot locomotion," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001.

[45] C. Paul, F. J. Valero-Cuevas, and H. Lipson, "Design and control of tensegrity robots for locomotion," *Transactions on Robotics*, vol. 22, no. 5, 2006.

[46] J. Pinskier and D. Howard, "From bioinspiration to computer generation: Developments in autonomous soft robot design," *Advanced Intelligent Systems*, 2021.

[47] H. Qi, K. Joyce, and M. Boyce, "Durometer hardness and the stress-strain behavior of elastomeric materials," *Rubber Chemistry and Technology*, vol. 76, no. 2, pp. 419–435, 2003.

[48] A. Rafsanjani, Y. Zhang, B. Liu, S. M. Rubinstein, and K. Bertoldi, "Kirigami skins make a simple soft actuator crawl," *Science Robotics*, vol. 3, no. 15, 2018.

[49] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, pp. 467–475, 2015.

[50] C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter, "Jointly learning to construct and control agents using deep reinforcement learning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[51] J. Seo, J. Paik, and M. Yim, "Modular reconfigurable robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 63–88, 2019.

[52] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *Proceedings of the National Academy of Sciences*, vol. 108, no. 51, pp. 20 400–20 403, 2011.

[53] K. Sims, "Evolving virtual creatures," in *Proceeding of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1994.

[54] A. Spielberg, A. Amini, L. Chin, W. Matusik, and D. Rus, "Co-learning of task and sensor placement for soft robotics," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1208–1215, 2021.

[55] A. Spielberg, B. Araki, C. Sung, R. Tedrake, and D. Rus, "Functional co-optimization of articulated robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[56] J. Talamini, E. Medvet, A. Bartoli, and A. De Lorenzo, "Evolutionary synthesis of sensing controllers for voxel-based soft robots," in *Proceedings of the Artificial Life Conference (ALIFE)*, 2019.

[57] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," in *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[58] O. Taylor and A. Rodriguez, "Optimal shape and motion planning for dynamic planar manipulation," *Autonomous Robots*, vol. 43, no. 2, pp. 327–344, 2019.

[59] N. V. Varghese and Q. Mahmoud, "A survey of multi-task deep reinforcement learning," *Electronics*, 2020.

[60] V. Vikas, E. Cohen, R. Grassi, C. Sözer, and B. Trimmer, "Design and locomotion control of a soft robot using friction manipulation and motor–tendon actuation," *IEEE Transactions on Robotics*, vol. 32, no. 4, pp. 949–959, 2016.

[61] M. G. Villarreal-Cervantes, C. A. Cruz-Villar, J. Alvarez-Gallegos, and E. A. Portilla-Flores, "Robust structure-control design approach for mechatronic systems," *Transactions on Mechatronics*, vol. 18, no. 5, pp. 1592–1601, 2013.

[62] I. Vitanov, A. Rizqi, and K. Althoefer, "Shape reconstruction of soft-body manipulator: A learning-based approach," in *Proceedings of the Annual Conference Towards Autonomous Robotic Systems (TAROS)*, 2020.

[63] J. Whitman, M. Travers, and H. Choset, "Learning modular robot control policies," *arXiv:2105.10049*, 2021.

[64] A. Zhao, J. Xu, M. Konaković-Luković, J. Hughes, A. Spielberg, D. Rus, and W. Matusik, "Robogrammar: Graph grammar for terrain-optimized robot design," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, 2020.

[65] Y. Zhu, J. Rossiter, and H. Hauser, "Learning in growing robots: Knowledge transfer from tadpole to frog robot," in *Proceedings of the International Conference on Biomimetic and Biohybrid Systems*, 2019.

APPENDIX A
LEARNING AND SIMULATION PARAMETERS

Table I lists the hyperparameter settings that we used for co-optimization. We represent each design as a concatenation of eight four-dimensional one-hot vectors indicating whether each PneuNet is present and which regulator it is connected to. Our policy is open-loop and receives the last four pressure actions as input. The actions are concatenated with the design representation to create a 44-dimensional input to the policy and value networks. Both the policy and value networks are four-layer feed-forward networks and each hidden layer has $[400, 200, 100]$ units, respectively. We train on a 32-core AMD EPYC 7502, with the experiment taking eight days to complete.

Hyperparameter	Value
Number of environments	96
Maximum timesteps	1M
Buffer size	100K
Batch size	512
Discount factor (γ)	0.95
Policy learning rate	0.0006
Q-function learning rate	0.0020
Optimizer	ADAM($\beta_1 = 0.9, \beta_2 = 0.999$)
Target network smoothing coefficient	0.01
Learning start	10K
Timesteps per SAC update	1
Timesteps per sampled design	20 (one episode)
Entropy linear decay start	200K
Entropy linear decay end	950K
Entropy target start	uniform
Entropy target end	0

TABLE I
SOFT ACTOR CRITIC AND CO-OPTIMIZATION PARAMETERS

Table II provides our simulation parameters. We use a linear elastic model and set Young's modulus for the silicone using known material parameters. The Young's modulus for the paper and disk were chosen to be significantly stiffer that the silicone. The Poisson ratios were set to maintain realism while avoiding numerical instabilities. The masses were measured from their real counterparts but then scaled by a factor of 2.5 to avoid numerical instability. The friction coefficient was chosen to qualitatively approximate the stick-slip behavior of the baseline design when all the PneuNets are inflated and deflated simultaneously.

Figure 11 shows the linear pressure scaling used to align the PneuNet bending response between simulation and reality. To calibrate the action scaling, we record the bending of a single PneuNet under pressures from 10 kPa to 100 kPa, in increments of 10 kPa. We then search for the pressures in simulation which achieve an equivalent bend and fit a linear function to the results.

Parameter	Value
Elastic model	linear
Young's modulus for PneuNet silicone	1160 kPa
Poisson ratio for PneuNet silicon	0.2
Young's modulus for PneuNet paper	2320 kPa
Poisson ratio for PneuNet paper	0.49
Young's modulus for disk	5000 kPa
Poisson ratio for disk	0.3
PneuNet mass	105 g
Disk mass	75 g
Friction coefficient	1.2
Gravity	9800 mN

TABLE II
SOFA SIMULATION PARAMETERS

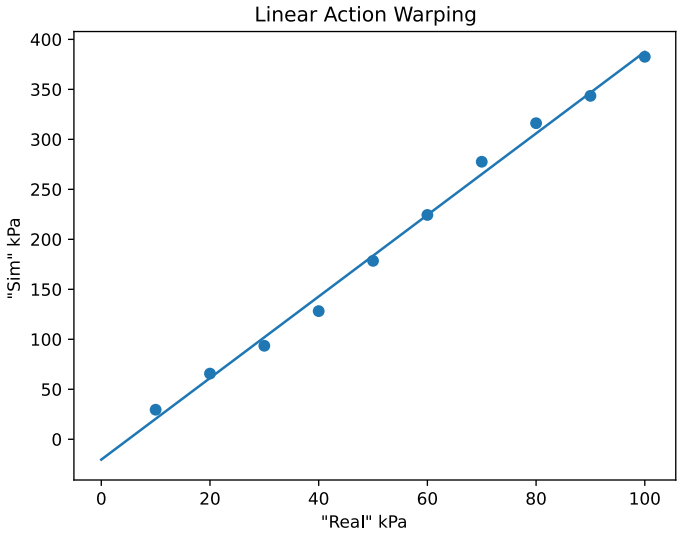


Fig. 11. The linear pressure scaling used to align the behavior of a single PneuNet in simulation and reality. In simulation, we scale the pressures output by the policy prior to applying we apply them. This helps to reduce discrepancies between simulation and reality caused by modelling error.

APPENDIX B
MODEL ORDER REDUCTION EVALUATION

The snapshot-POD reduction method has two tolerances that must be set, one for the error induced by the basis Φ, and one for the error induced by the hyperreduction. To explore the effect of these two parameters and to verify the validity of our reduced models, create a test set of four designs and animations:

1) the **Hamiltonian animation** uses the baseline four-legged design and iterates through the pressure extremes of the four PneuNets.
2) the **baseline animation** also uses the baseline design and a gait similar to the baseline gait from our experiments.
3) the **two leg animation** uses a design with two alternately inflating PneuNets at a 90° angle.
4) the **six leg animation** uses a design with six PneuNets that alternately inflate in two groups of three.

For each design and animation, we compare the node positions of the mesh using the unredeced and reduced-order models. We perform a coarse grid search over the two MOR tolerances and

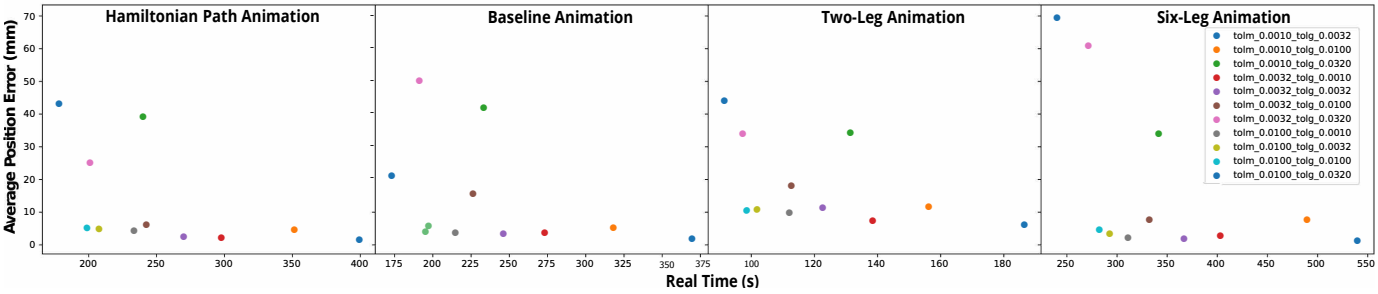


Fig. 12. This plot shows the trade off between positional errors introduced by the reduction and the time to simulate each animation for a 3x3 grid search over the 2 MOR tolerances. We choose a mode tolerance of 0.0032 and a hyperreduction tolerance of 0.0010 (red dot) for our experiments.

plot the simulation speed and accuracy of each reduction. Errors are computed as the L2 distance between the node positions of the reduced and unreduced model averaged over each node in the mesh and each timestep. Based on this analysis, we use a mode tolerance of 0.0032 and a GIE tolerance of 0.001 for our experiments.

Appendix C
Hardware and Fabrication Details

Figure 13 displays our experimental platform and robot design. We keep a pressure chamber at 400±100 kPa attached in series with three pressure regulators (Enfield TR-010-gs). The learned and baseline policies are executed on a Raspberry Pi where the pressure commands are converted to voltage commands and sent to a programmable power supply (BK Precision 9129B). The power supply then sends voltages to each of the three pressure regulators through a breadboard that converts the voltages into an acceptable range. Each pressure regulator is connected by a lightweight tube to the robot. We connect an external pressure sensor to each regulator to verify that the correct pressure is applied.

We created a modular assembly scheme in which any robot from the design space can be built. We 3D-printed a lightweight polymer disk (Fig. 13 (right)) with uniformly distributed locations where we can attach the PneuNet actuators while routing the pneumatic cables away from the robot (Fig. 13 (left)). We 3D-printed moulds for the PneuNets based on a modification of the design given in the Soft Robotics Toolkit [25] such that the two end prismatic segments are filled rather than hollow (Fig. 13 (right)), which allows the first prismatic segment to be used as an attachment point to the disk. We fabricated the PneuNet actuators using Smooth-On DragonSkin 30 silicone.

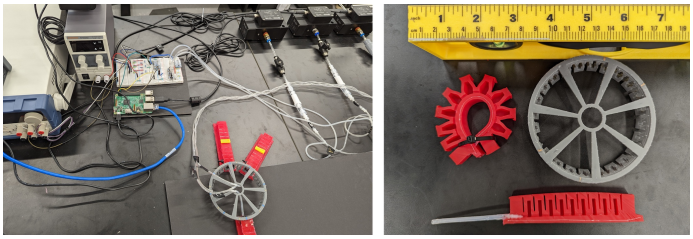


Fig. 13. **Left**: A picture of our hardware set up consisting of a Raspberry Pi, three pressure regulators, two power supplies, three pressure sensor, and a breadboard connecting everything to the Pi. **Right**: A picture of our 3D-printed disk and a molded PneuNet cut in half to display the internal structure.