Impact of training environment on co-optimized crawling soft robots in simulator and real world

Written by:

Shiyao Ni

Supervised by: Audrey Sedal, Prof.

Department of Mechanical Engineering

McGill University

Montréal, Québec, Canada

Dec 5, 2023

A thesis submitted to McGill University in partial fulfilment of the requirements of the Undergraduate Honours Program.

Abstract

Soft robots have “mechanical intelligence”: the ability to passively exhibit behaviors that would otherwise be difficult to program. Exploiting this capacity requires consideration of the coupling between design and control. Co-optimization provides a way to reason over this coupling. Yet, it is difficult to achieve simulations that are both sufficiently accurate to allow for sim-to-real transfer and fast enough for contemporary co-optimization algorithms. In this work, the effectiveness of co-optimization were compared to robot with trained policy only and different order-reduced models were used to evaluate their robustness during sim-to-real transfer. The generalization of the framework to new terrains, and the effectiveness of domain randomization as a means to improve sim-to-real transfer were also studied in this.

Abrégé

Les robots souples possèdent une ”intelligence mécanique” : la capacité d'exhiber passivement des comportements qui seraient autrement difficiles à programmer. Exploiter cette capacité nécessite de prendre en compte le couplage entre la conception et le contrôle. La co-optimisation offre un moyen de raisonner sur ce couplage. Cependant, il est difficile d'obtenir des simulations à la fois suffisamment précises pour permettre un transfert sim-réel et assez rapides pour les algorithmes de co-optimisation contemporains. Dans ce travail, l'efficacité de la co-optimisation a été comparée à celle de robots avec une politique d'entraînement uniquement, et différents modèles réduits ont été utilisés pour évaluer leur robustesse lors du transfert sim-réel. La généralisation du cadre à de nouveaux terrains, et l'efficacité de la randomisation de domaine comme moyen d'améliorer le transfert sim-réel ont également été étudiés dans ce contexte.

Acknowledgements

First and foremost, I wish to extend my appreciation to my thesis supervisor, Professor Audrey Sedal, whose support and guidance were unwavering throughout this research project. Her expert instructions and suggestions played a pivotal role in the successful completion of this work.

I am also deeply grateful to Dr. Charles Schaff and Professor Matthew R. Walter for their generous guidance in navigating the co-optimization framework and for their assistance in setting up the sim-to-real transfer experiments.

Furthermore, I would like to express my sincere thanks to all the members of the MACRObotics Group at McGill University and the Robot Intelligence through Perception Lab at the Toyota Technological Institute at Chicago. Their assistance with writing and computing job setup greatly contributed to the progress of this research.

Lastly, but by no means least, I extend my appreciation to my family and friends for their unwavering support throughout my academic journey.

Contents

Abstract i

Abrégé ii

Acknowledgements iii

Table of Contents vi

List of Figures ix

List of Tables x

1 Introduction to Soft Robotics **1**

2 Background and Prior Work **6**

3 Simulation and Machine Learning Framework **10**

3.1 Model Order Reduction 12

3.2 Machine Learning 14

3.2.1 Reinforcement Learning 14

3.2.2 Design Distribution 15
3.2.3 Domain Randomization 17
3.3 Co-optimization 18

4 Sim-to-Real Transfer **22**
4.1 Test Setup and the Construction of the Robot 23
4.2 Experiment procedure 24
4.2.1 Evaluation 25
4.2.2 Test Surface and Robot Starting Position 26
4.3 Sources and Mitigation of Error 28
4.3.1 Pressure Regulator Operational Range and Calibration 28
4.3.2 Friction and Dust 29
4.3.3 Measurement Error 30
4.4 Evaluation of Sim-to-Real Results 30

5 Results **32**
5.1 Results 32

6 Discussion and Conclusion **38**
6.1 Discussion 38
6.1.1 Effect of Co-Optimization on Sim-to-Real Transfer 38
6.1.2 Effect of Tolerance of Reduced Models on Sim-to-Real Transfer . . . 40

6.1.3 Effect of Domain Randomization on Sim-to-Real and Robot Performance 41

6.2 Conclusion 43

List of Figures

1.1 A soft grasper designed by Filip Ilievski et al. [1] grasping an egg 2

3.1 This plot illustrates the trade-off between positional errors introduced by the reduction and the time required to simulate each animation for a 3x3 grid search over the two MOR tolerances. The red dot, (1), represents the low-tolerance model, while the pink dot (2) represents the high-tolerance model. 13

3.2 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. 19

4.1 **Left**: A picture of the 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 the 3D-printed disk and a molded PneuNet actuator cut in half to display the internal structure. 23

4.2 The linear pressure scaling used to align the behavior of a single PneuNet in simulation and reality. In simulation, the pressures output by the policy were scaled prior to applying onto real-world experiments. This helps to reduce discrepancies between simulation and reality caused by modelling error. . . . 25

4.3 The starting position of the robot on silicone mat 27

5.1 The co-optimization framework jointly learns the design and control of crawling soft robots (top 2) 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. (a) Robot trained with τ_{hi}, μ_{rand}, Seed 0. (b) trained with τ_{low}, μ_{fix}, Seed 0. (c) Baseline: design and control developed by expert. 33

5.2 (a)-(f) Sim and real reward for final trained robots in the four simulation settings $\tau_{lo/hi}$ and $\mu_{fix/rand}$; and the L1 robot design and the expert baseline design with trained policy only. Mean is red square; median orange line; IQR shaded box. (g) Mean normalized sim-to-real transfer error for same six robots. 35

5.3 (a) Mean normalized reward at $\mu = \{0.65, 0.7, ..., 1.3\}$ for all co-optimized robots in τ_{low} environment. (b) Same mean normalized reward for robots co-optimized with τ_{hi}, evaluated in τ_{hi} environment. 37

List of Tables

3.1 List of all training environments. (DR stand for friction domain randomized and fixed stand for trained on specified friction level; Co-Opt stand for co-optimized training and design stand for the specific design that the controller policy was trained on; seed stand for the seed used to initialize both the design distribution and control policy) 20

4.1 List of all training categories. (DR stand for friction domain randomized and fixed stand for trained on specified friction level; Co-Opt stand for co-optimized training and design stand for the specific design that the controller policy was trained on) 30

List of Acronyms

AI	artificial intelligence.
DR	domain randomization.
FEA	finite element analysis.
FEM	finite element method.
MDP	Markov Decision Process.
ML	machine learning.
MOR	model order reduction.
RL	reinforcement learning.

Chapter 1

Introduction to Soft Robotics

Soft robots are constructed with soft materials. The compliant nature of the robot allows soft robots to be adaptable and deformable. This adapability leads to the robot responding to contact and control inputs in a complex manner, exhibiting behaviors that have demonstrated their effectiveness across various domains. The design of soft robots revolves around the control policy that governs their motion, enabling them to possess a form of "mechanical intelligence"[2]. This intelligence entails the interaction between materials and mechanisms with the environment in ways that facilitate the completion of desired tasks. For instance, Fig. 1.1 demonstrates a soft grasper grasping an egg; the compliant nature of silicone allowing the robotic grasper to conform to the shape of the egg, adding additional contact surface and reducing contact pressure. However, this "mechanical intelligence" can

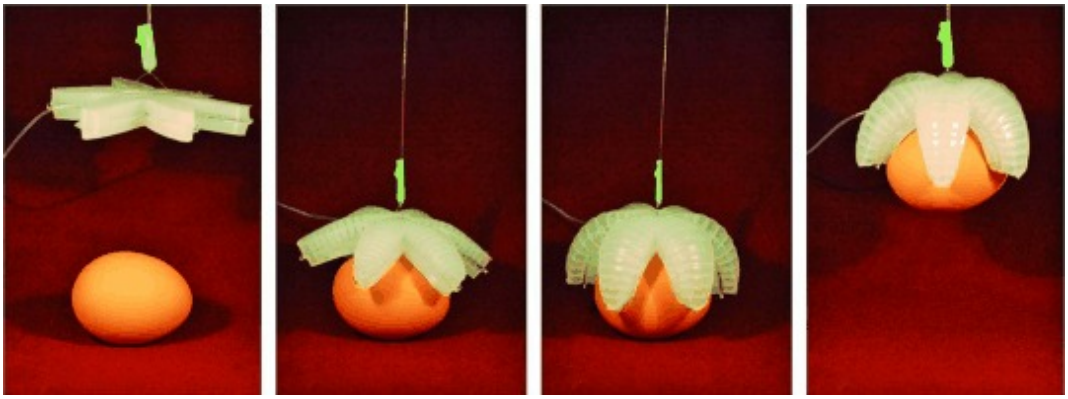

Figure 1.1: A soft grasper designed by Filip Ilievski et al. [1] grasping an egg

only be achieved and maximized if the controller of the robot matches its design. Therefore, the methods to co-optimize both the morphology and control of the soft robot provides a promising approach. Although there exists extensive research on joint design and control optimization [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] for rigid robots, less exists for soft robots.

The complete framework for the simulation and sim-to-real transfer of the design and control of modular soft robots for locomotion tasks with 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 are proposed by Schaff et al. [16] and explored in this work. The algorithm utilizes this policy to efficiently concentrate its search on top-performing designs. In order to promote the development of "mechanical intelligence", an open-loop controller is learned, forcing the expression of complex behaviors through the resulting soft body.

One crucial pre-requisite to achieve successful co-optimization training is a simulator that is both fast enough for the learning agent to explore a large enough set of design and control pairs and accurate enough for the final design and control to be real world physical realizable and for the physics to endure sim-to-real transfer experiment. However, modelling soft bodies is computationally intensive job. The fast and accurate simulation of soft robots remains as an option question, and the few existing co-optimization approaches for soft robotics suggest different simulation strategies [17, 18, 19]. These simulators have different degrees of realism and their ability of creating soft robots that can transfer through the reality gap is unclear.

With a focus on achieving sim-to-real transfer, finite element analysis (FEA) is used, a widely accepted standard for accurately simulating deformable materials. In this work, it is demonstrated that given sufficient computational resources, FEA simulations enable the direct transfer of co-optimized soft robots. However, it's important to note that high-fidelity FEA simulations can be significantly slower than real-time, making them impractical for learning-based methods.

In order to improve the computational efficiency of FEA simulations while maintaining their accuracy, researchers have recently employed a technique proposed by Goury and Duriez [20]. They introduced a model order reduction method within the open-source FEA simulation framework SOFA [21, 22]. While this approach enhances simulation efficiency, it does come with a significant upfront cost that can hinder the ability to learn across different robot designs. To address this limitation, an adaptable reduction framework capable of reducing

modular components were proposed to be used, which can then be combined to create reduced-order models of soft robots with varying morphologies. This proposed model-order reduction technique can be fine-tuned to different levels of fidelity by adjusting reduction tolerances, allowing us to strike a balance between simulation speed and model accuracy.

However, given the high sample complexity of modern data-driven optimization methods, the speed-fidelity trade-off becomes increasingly important. How much can model fidelities affect the training in simulation as well as robustness during sim-to-real transfer still remains unclear. Therefore, in this work the sim-to-real transfer of robot designs and policies learned at varying levels of reduced model fidelity are compared and studied.

In addition to evaluating the effect on different fidelity models, domain randomization [23] is applied to some training sessions to be evaluated as an alternative way of improving realism of the simulation. It is aimed to assess if there is an enhancement in sim-to-real transfer when utilizing a lower-fidelity model and whether it results in adaptive gaits for new terrains. In the field of rigid robotics, domain randomization has been shown to be effective for sim-to-real transfer when employing reinforcement learning [24, 25, 26]. Yet, the physics of soft robotics differs from rigid robots, the conclusion on how domain randomization strategies may generalize still remain unclear.

Although the approach disscussed prior is general, the work focused on easily manufacturable PneuNet actuators [27], which were used to create crwaling and walking robots [28, 29]. The

proposed approach is experimentally validated by learning combinations of PneuNets and their controllers, resulting in the attainment of faster gaits. Importantly, the framework demonstrates the ability to successfully transfer optimized design-control pairs to real-world scenarios on a variety of flat terrains characterized by both high and low friction levels with the robot legs.

Chapter 2

Background and Prior Work

The challenge of jointly optimizing the physical structure and control of a rigid robot has a well-established history in robotics research. Neural networks were often employed to optimize both the design and control of the robot [5, 30, 31, 32]. Another common approach involved assuming access to a parameterized model of the robot's dynamics and optimizing these parameters in conjunction with control parameters [6, 7, 10, 33, 34, 35, 36]. With the rise of efficient high-fidelity physical simulators, joint optimization methods based on reinforcement learning have emerged, enabling the learning of effective design-controller pairs for rigid bodies without prior knowledge of the dynamics [12, 13, 15, 37]. However, the challenge lies in the transfer of these learned solutions from simulation to the real world. This work specifically focuses on

utilizing high-fidelity simulators, which have been demonstrated to facilitate the transfer of knowledge to rigid robots [24]. In contrast, achieving the necessary level of fidelity for soft-bodied robots typically demands computationally intensive simulators, making learning-based co-optimization infeasible. This is the challenge addressed in this work.

Compared to rigid robotics, the exploration of jointly optimizing the design and control of soft robots is a less developed area. Among the existing research, the majority of it primarily focuses on simulation. Many of these approaches involve reasoning over design and control spaces that encompass both discrete and continuous parameters. For instance, voxel-based soft robots (VSRs) [38]consist of discrete voxels, yet the input frequency to each voxel is treated as continuous. Spielberg et al. [18] introduced an autoencoder-based method capable of optimizing the placement of a large number of such voxels for real-world locomotion, achieving results more than 2 times faster than the baseline.

Cheney et al. [39] utilize an evolutionary neural strategy to generate designs for VSRs that can move in simulation. Kriegman et al. [40]describe an approach to deform the structure of VSRs when subjected to damage while maintaining the original control policy's validity. Ma et al. [41] employ a material point method-based simulation and gradient-based optimization techniques to co-optimize the shape and control of simulated swimming robots. Deimel et al. [42] leverage 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 promising for the field of soft robotics, and recent simulation-based benchmarks have

enabled a rigorous comparison of co-optimization methods [43, 44]. However, most of above work is limited to simulation only, and the transferability to real world still remains unkown. Indeed, experiments conducted on voxel-based soft robots demonstrate significant disparities between their behavior in simulation and reality [45]. Challenges related to modeling friction and stick-slip behavior [28, 46] have been identified as core reasons for the difficulties encountered in achieving successful transfer.

Recent advancements in the field of soft robotics have introduced techniques aimed at bridging the gap between simulation and real-world application. Zhang et al. [47] present an system identification method tailored to differentiable soft robot simulators. Meanwhile, Dubied et al.[48] propose a differentiable finite-element model for soft robotics, which they experimentally validate using canonical mechanical problems such as a cantilevered beam. It is worth noting that these approaches focus on fixed robot designs rather than exploring a space of designs, which distinguishes them from our work. In the realm of rigid robotics, Tobin et al. [23] demonstrate that applying domain randomization to a policy's visual inputs in simulation enables the policy to directly transfer to the real world in a zero-shot manner. Similarly, Tan et al. [24] employ domain randomization over the parameters of the dynamics model to transfer the control policy of a quadruped robot trained in simulation to the real world. Similar approaches have been leveraged to facilitate sim-to-real transfer in various robot learning domains [25, 26]. However, in the context of soft robotics, characterized by higher degrees of freedom (DoF) and a broader range of relevant physical

phenomena, the applicability of domain randomization for sim-to-real transfer remains relatively unexplored. Centurelli et al. [49] demonstrate that a policy trained on randomly generated trajectories can control a dynamic soft robot arm. Li et al. [50] utilize domain randomization by introducing noise to both actions and observations, showcasing control of a soft arm in path-following. Meanwhile, Tiboni et al. [51] incorporate randomness into the material properties of the soft robot's body, demonstrating, in simulation, control of a soft arm and a crawling robot akin to the expert baseline in this study. Prior findings highlight the need for a study on the impact of co-optimization and domain randomization on sim-to-real transfer performance using order-reduced models of varying fidelity.

Chapter 3

Simulation and Machine Learning Framework

This research project is based on the co-optimization framework proposed by Schaff et al. [16]. The framework comprises two parts, developed in Python 3 and Python 2 respectively. The Python 3 packages play a vital role, serving as the central commander. They oversee the reinforcement learning processes and manage the distribution of design components within the framework. Additionally, they are integral in optimizing the sequence of training simulations to enhance the efficiency of the training sessions.

Concurrently, the Python 2 packages operate in tandem, executing commands from the Python 3 modules. Their main task is to initiate and manage simulation scenes within the

SOFA FEM simulator, ensuring streamlined coordination throughout the framework.

The SOFA framework is a dynamic FEM simulator that uses finite element models to computer the deformation of nodes and their interaction forces. It accounts for factors like Coulomb friction and internal air pressure, while also emphasizing the acceleration of each node and the associated inertial forces.

SOFA can import multiple parts into the simulator and assemble them by adding specific constraints between components as needed. Given that the proposed design space consists of one central disk with eight potential positions for attaching PneuNet actuated legs, SOFA emerged as the ideal simulator for this project. Several reduced models were initially processed using the Model Order Reduction (MOR) technique, detailed in section (3.1). These models were pre-positioned within the space, undergoing translation, rotation, and orientation adjustments, to facilitate direct connection to the central disk. Appropriate constraints were established at contact points to ensure numerical stability and physical accuracy. This setup simplifies the co-optimization process, enabling machine learning agents to easily initiate simulation environments with their preferred robot design.

Among the plethora of FEM simulators available, SOFA stands out as an open-source option, offering high accuracy at the expense of computational speed. Furthermore, SOFA does not support GPU acceleration, adding to computational time. Given that reinforcement learning demands numerous iterations (1 million steps in this research), it is crucial to ensure the

models are sufficiently fast for computation on a cluster equipped with a 64-core AMD EPYC 7502 CPU.

3.1 Model Order Reduction

Model order reduction is a plugin in SOFA framework that utilizes mathematical technique to reduce a complex finite element model while preserving its key dynamic or static behavior in simulation. To achieve high-accuracy FEM simulation and ensure real-world transferability, high-fidelity finite element models are employed. Our PneuNet models comprise 2603 nodes excluding the cavity or collison mesh (cavity and collison mesh were used to compute deformation as well as contact between other objects in the simulator) to ensure numerical stability in the SOFA simulator and maintain accuracy. Nonetheless, FEM simulations with these high-resolution models can be significantly slower than real-time, hindering the feasibility of learning-based methods [16]. Goury and Duriez [20] introduced a model order reduction technique in SOFA that enhances computational efficiency without compromising accuracy. While the MOR technique boosts simulation speed, its high initial cost hampers the ability of machine learning agents to suggest new designs. A reconfigurable reduction framework first reduces a set of composable parts, which are then combined to create reduced order models of soft robots with varying morphologies. Therefore, the reinforcement learning agents (Sec. 3.2.1) can command the SOFA simulation interface to create robots with new morphologies using reduced model.

The MOR method's flexibility allows for tuning to yield models of varying fidelity, balancing computational speed with model accuracy.

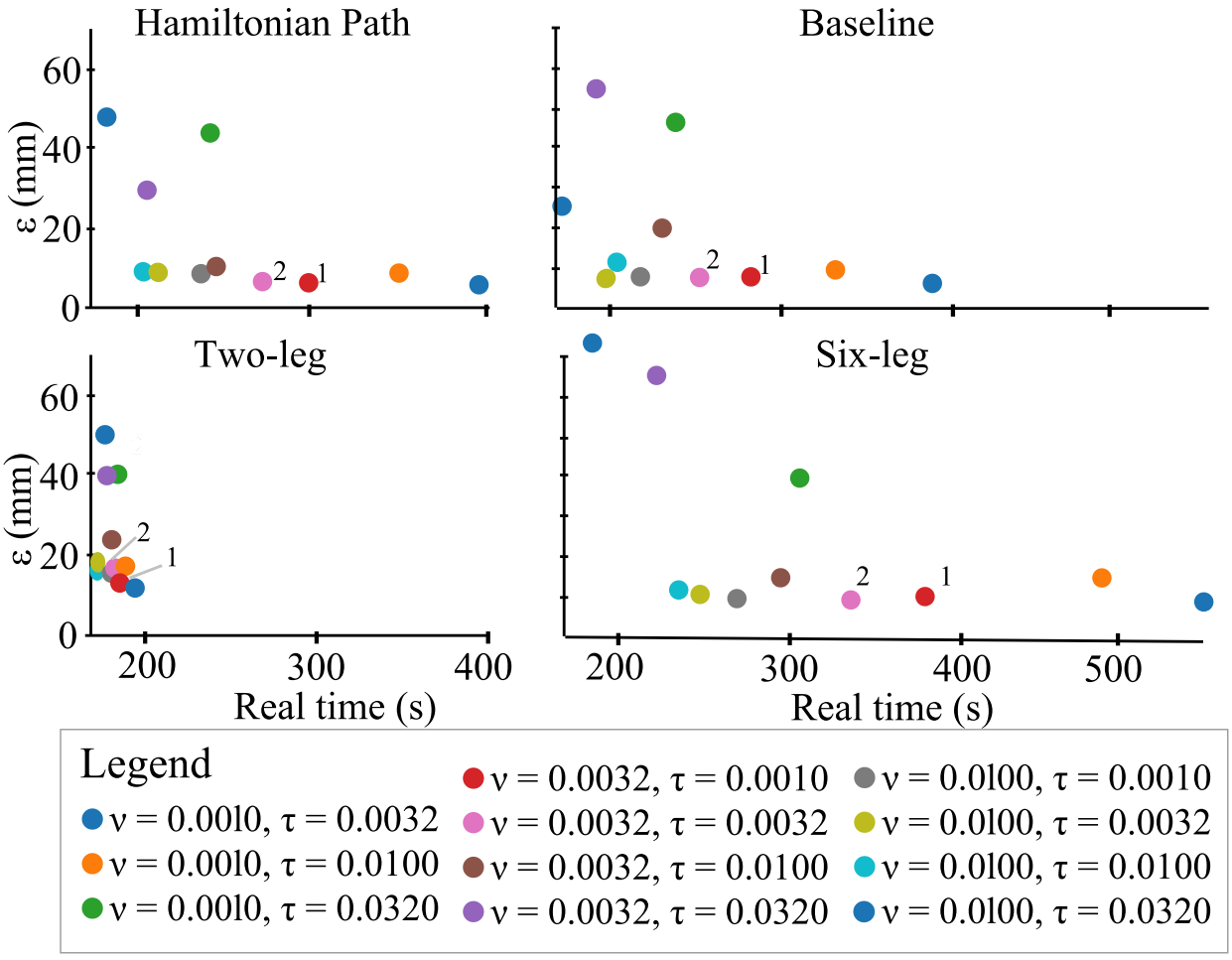

Figure 3.1: This plot illustrates the trade-off between positional errors introduced by the reduction and the time required to simulate each animation for a 3x3 grid search over the two MOR tolerances. The red dot, (1), represents the low-tolerance model, while the pink dot (2) represents the high-tolerance model.

Considering the high sample complexity of contemporary data-driven optimization methods, striking a balance between speed and fidelity is paramount. In previous work by Schaff

et al. [16], errors associated with several reduced order models were examined in fig. 3.1. Before the integration of domain randomization, co-optimization had already successfully designed and controlled a soft robot (L1) that outperformed the expert-designed baseline using a reduced model with MOR tolerances tolg = 0.0032 and tolm = 0.0010. With domain randomization, the co-optimization framework is anticipated to yield more accurate results. Thus, a model with slightly higher tolerances, offering less accuracy when operating on 2 legs, was selected to evaluate the ability of domain randomization on sim-to-real transfer even with less accuracy.

3.2 Machine Learning

In this section, the machine learning approach used to co-optimize the robots is explained. The co-optimization framework contains a utility model that enables the sofa simulator and RL agent to load and train directly on reduced models. An evolutionary algorithm that can automatically select and reconfigure robot with different design with new sample of designs every episode was used.

3.2.1 Reinforcement Learning

In this research project, reinforcement learning is the primary learning method employed to update the control policy of the robot. A reinforcement learning agent seeks to maximize the reward function through exploration and exploitation. For the specific task of this

project, which involves designing and controlling a soft robot crawler, the reward function is determined by the distance the robot travels in a 20-second span. Thus, the reinforcement learning agent aims to update its control policy to maximize this reward. Although the co-optimization framework can accommodate multiple RL algorithms, soft actor-critic[52] was used in this project. The agent is set to update its policy per time step, and the policy takes the design parameters, along with the four most recent actions as well as the current state of the robot as input, and outputs pressure targets for each regulator.

3.2.2 Design Distribution

After removing symmetries, there exist 6,972 standalone designs that their distribution are resampled based on their reward in previous steps at each training steps. The design samples are big in the beginning, before the entropy decaying process starts. The simulation interface will simulate SOFA environments in parallel to speed up the training by taking advantage of multiple cores of the CPUs. Ensuring a suitable design and control pair is essential for a robot to exhibit physical intelligence. Thus, co-optimization techniques are implemented in this research project. Apart from the reinforcement learning component mentioned in 3.2.1, which handles control policy updates, the robot's design pertains to its morphology and is represented by a vector of length 8. Each vector entry denotes a position on the central disk, and its value (from the set $\{0, 1, 2, 3\}$) indicates whether the disk slot is attached to a PneuNet and to which pressure regulator (numbered 1, 2, or 3) the PneuNet is connected. Equation 3.1

has demonstrated the structure of robot design vector space, with $\underline{D}$ representing the robot design and x_i representing the PneuNet actuator presence and controller number.

$$\underline{\mathbf{D}} = \begin{bmatrix} x_0 \\ \vdots \\ x_i \\ \vdots \\ x_7 \end{bmatrix} \qquad \text{where,} \qquad \begin{array}{l} \mathbf{x_i} \in [0, 1, 2, 3], \\ i \in [0, 1, \ldots, 7] \end{array} \tag{3.1}$$

Given the available configurations, there are 41,202 unique designs, which can be reduced to 6,972 by leveraging symmetry in regulator assignments. All these designs follow a linear entropy decay schedule from the 200K time step to the 950K time step. During the initial 200K time steps, all 41,202 designs are simulated based on the current control policy, and a design distribution is maintained that stores each design's performance, updated after each episode. Performance is gauged as a weighted average of all rewards each design receives post-simulation. After the 200K mark, designs with subpar performance are temporarily sidelined, while the active ones continue to be trained. Should an active design underperform and rank among the "eliminated" designs, a previously set-aside design could be reintroduced for training. It's anticipated that post the 950K time step, only one superior design will persist, undergoing training by the RL agent until the 1M time step. Equation 3.2 demonstrates the relationship between total number of active designs N_j and training step j. Step which decay

starts and ends is a hyperparameter that will be set before the start of training session.

$$N_j = \exp(h_j)$$

$$\mathrm{h} = \begin{cases} \ln(6972) & \text{if } j < j_{\text{decay start}} \\ \ln(6972) - \left(\frac{j - j_{\text{decay start}}}{j_{\text{decay ends}} - j_{\text{decay start}}} \cdot \ln(6972)\right) & \text{if } j_{\text{decay ends}} < j < j_{\text{decay start}} \\ 0 & \text{if } j > j_{\text{decay ends}} \end{cases} \tag{3.2}$$

3.2.3 Domain Randomization

Domain randomization is a technique prevalent in machine learning, especially for training intelligent systems. Within this approach, one or more simulation properties are randomized, diversifying the simulated environment. Consequently, the model is exposed to varied scenarios, resulting in a more resilient system during sim-to-real transitions.

Given domain randomization's characteristics, a lower-fidelity reduced model was also chosen to evaluate simulator performance and sim-to-real robustness. To effectively leverage the aforementioned co-optimization framework in training environments with randomized friction domains, significant enhancements were made to the Python 2 interface that oversees SOFA simulator scene initialization. A new function is implemented

in SOFA Python 2 for every new parallel sofa environments, enabling the automated generation of friction values between 0.65 and 1.30, with a 0.01 increment.

Subsequent to this, a function was designed to systematically record all friction levels during training, preserving them in a text file for in-depth future scrutiny. It's vital to note that these code modifications were hard-coded, a choice driven by the immutable communication layer between Python 3 and Python 2.

Additionally, parameters that do not change between environments and global parameters for reinforcement learning framework are given in Gin config and Yaml files, inialized once when the program onset.

3.3 Co-optimization

The structure of the co-optimization framework is comprehensively depicted in Fig. 3.2. Throughout the process, only the environments (Envs) shown in Fig. 3.2 are governed by Python 2 packages, while all other components are overseen by Python 3 packages. A bridge exists between the Python 2 and Python 3 interfaces, facilitating communication between the simulator and the reinforcement learning agents. The communication layer uses JSON scripts to exchange information including robot design number, final reward at each sofa parallel environment etc. between Python 2 and Python 3. The reward info is also used to update the design distribution.

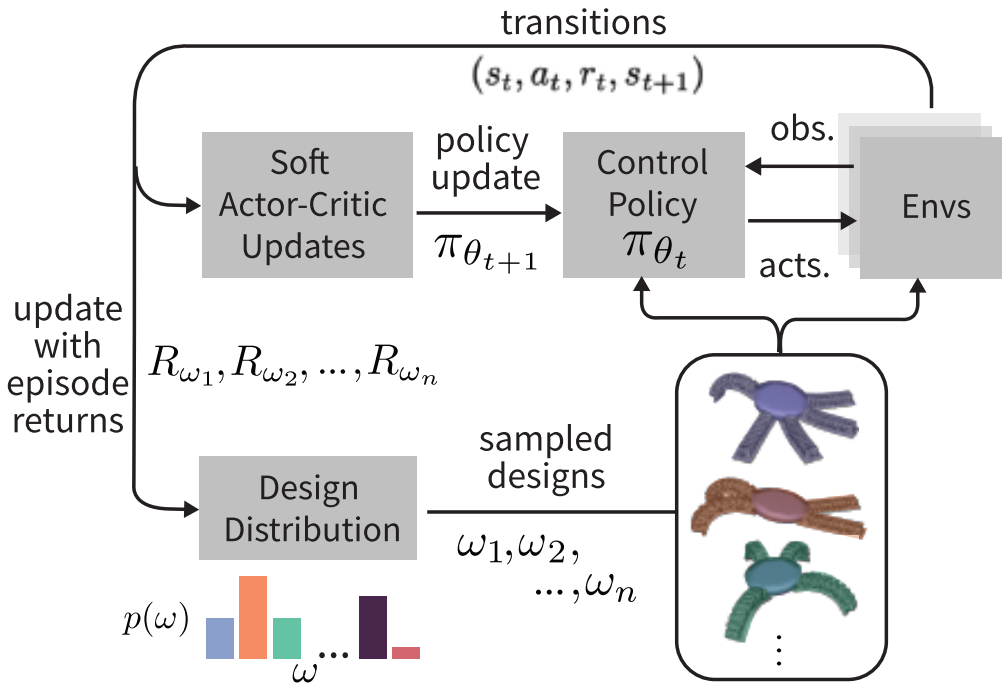

Figure 3.2: Our approach maintains a distribution over designs $p(\omega)$. At each iteration, the method samples a set of designs $\omega_1, \ldots, \omega_n$ and controls each using a shared, design-conditioned, policy π_θ. We train the policy using soft actor-critic on a mixture of data from different designs, and update the design distribution based on the episode returns of the sampled designs.

To investigate the relationship between co-optimization, reduced models, and domain randomization in terms of friction, training was done on the environments listed in Table 3.1.

We chose L1 and the baseline robot to be trained on policy only since L1 was the best-performing soft robot crawler from prior work [16], and by comparing row 1 and 5, a conclusion can be drawn regarding the effect of co-optimization. The baseline robot was selected for training solely on control policy to evaluate the potential of a robot with an

	Reduced Model	Co-Opt or Design	DR or fixed	Seed
1	High fidelity model	L1	fixed	seed 0
2	High fidelity model	L1	DR	seed 0
3	High fidelity model	Baseline	fixed	seed 0
4	High fidelity model	Baseline	DR	seed 0
5	High fidelity model	Co-opt	fixed	seed 0
6	High fidelity model	Co-opt	fixed	seed 18
7	High fidelity model	Co-opt	DR	seed 0
8	High fidelity model	Co-opt	DR	seed 18
9	Low fidelity model	Co-opt	fixed	seed 0
10	Low fidelity model	Co-opt	fixed	seed 18
11	Low fidelity model	Co-opt	DR	seed 0
12	Low fidelity model	Co-opt	DR	seed 18

Table 3.1: List of all training environments. (DR stand for friction domain randomized and fixed stand for trained on specified friction level; Co-Opt stand for co-optimized training and design stand for the specific design that the controller policy was trained on; seed stand for the seed used to initialize both the design distribution and control policy)

expertly designed morphology.

For each co-optimized simulation environment, two different seeds were used to initialize the environment to avoid selective presentation. In a non domain randomized, non co-optimized run, seeds are used to initialize the control policy in training only and if co-optimization is enabled, design distribution of the robot will be controlled by another seed; the randomized friction is controlled by seed in domain randomized runs. Although at the first 200K steps by default, all 6,972 will be trained together, they carry a weight by which the controller will have a different emphasize on each robot design and the weight will be controlled by the design distribution seed.

Owing to the challenges of implementing a seed for the frictional domain randomization, given that each Python 2 environment operates independently, three distinct training runs were initiated for co-optimized, friction domain randomized simulations with seed 0, namely row 7 and 11 in the Table 3.1.

Incorporating both the high fidelity and low fidelity models into the training list allows for conclusions to be drawn regarding the impact on models and the potential of domain randomization to enhance model accuracy in simulations.

Chapter 4

Sim-to-Real Transfer

After completing the training sessions, the best-performing model from each category is manufactured and tested. Robot bodies are manufactured using a 3-D printed disk and assembled with PneuNet actuators using zip-ties. PneuNet actuators are connected to different pressure regulators as indicated by their configuration. Pressure regulators are controlled by Raspberry Pi circuits, which include a back-end pressure monitor and a programmable power supply.

Prior to each test run, evaluations are conducted for each final design and control pair from the training sessions. The policy of the robot were recorded in the evaluations and can be visualized. Pressure input to each PneuNet is saved in the evaluation as Plotly plots, which can then be downloaded as JSON files with Plotly formats. The JSON files are then

processed to create an array of 20 entries per pressure regulator, indicating the pressure states at each second during the 20-second actuation period.

4.1 Test Setup and the Construction of the Robot

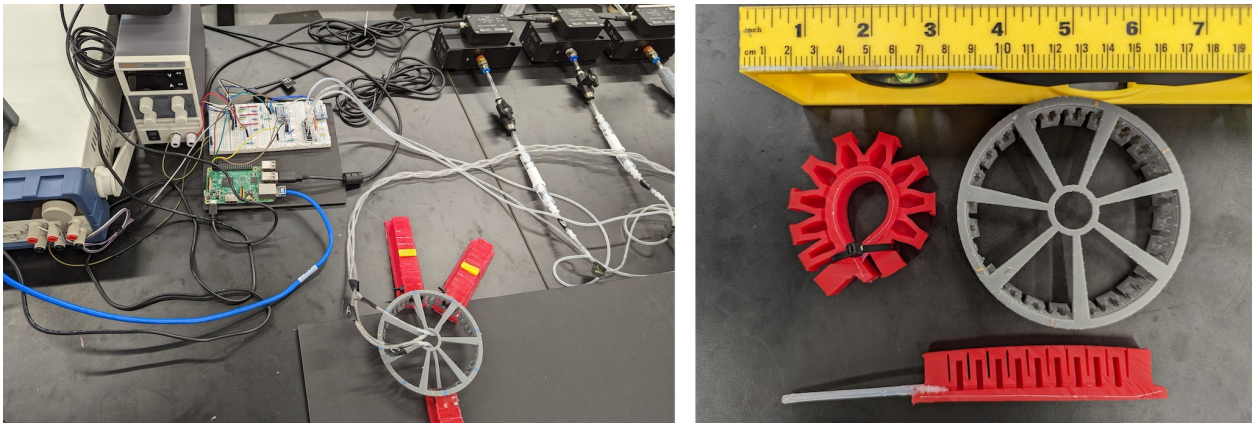

Figure 4.1: Left: A picture of the hardware set up consisting of a Raspberry Pi, three pressure regulators, two power supplies, three pressure sensor, and a breadboard connecting everything to the Pi. **Right**: A picture of the 3D-printed disk and a molded PneuNet actuator cut in half to display the internal structure.

Figure 4.1 displays the experiment setup and the construction of the robot. Three pressure regulators modeled Enfield TR-010-gs were attached in series with a pressure chamber set at 400±100 kPa. 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 modeled 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 1/16 inch inner diameter silicone tube to

the robot. An external pressure sensor was connected to each regulator downstream to store and verify the operational pressure during experiments.

A modular assembly scheme, allowing for the construction of any robot from the design space, was proposed. A central polymer disk (Fig. 4.1 (right)) with uniformly distributed locations where PneuNet actuators can be attached to while routing the pneumatic cables away from the robot (Fig. 4.1 (left)) is 3D-printed.

Moulds for PneuNets, based on a modification of the design from the Soft Robotics Toolkit (`https://softroboticstoolkit.com/`), were also 3D-printed. The modification focused on filling the two end prismatic segments instead of leaving them hollow (Fig.4.1 (right)), allowing the first prismatic segment to be used as an attachment point to the disk. Smooth-On DragonSkin 30 silicone were used to fabricate the PneuNet actuators.

4.2 Experiment procedure

After completing all the training sessions, the test platform, including the pressure regulator and Raspberry Pi control circuit, was established. Regulator pressure data were obtained from evaluations towards the final training step.

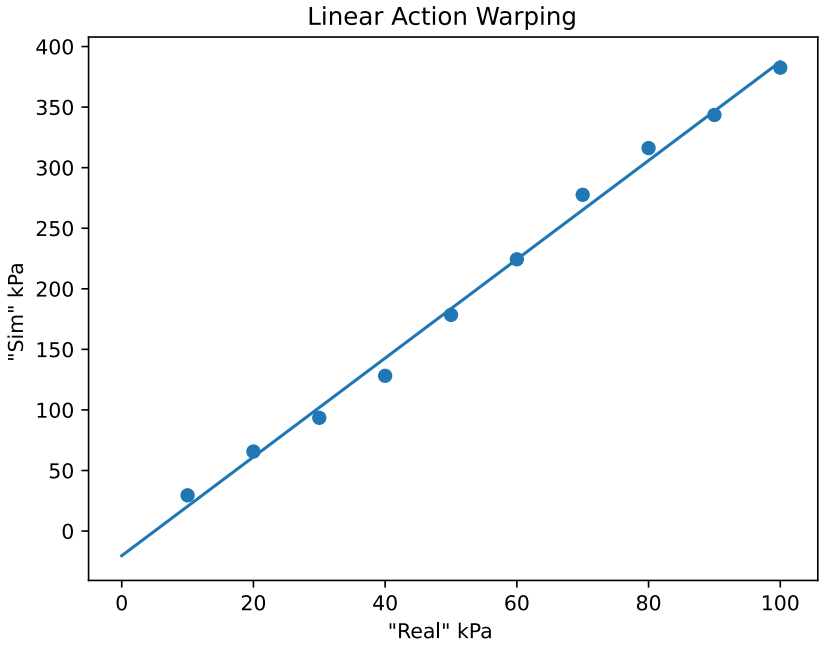

Figure 4.2: The linear pressure scaling used to align the behavior of a single PneuNet in simulation and reality. In simulation, the pressures output by the policy were scaled prior to applying onto real-world experiments. This helps to reduce discrepancies between simulation and reality caused by modelling error.

4.2.1 Evaluation

Each trained design and control pair in the simulation environment were evaluated to have their final reward in simulation, pressure profiles analyzed. The co-optimization framework introduced in sec 3.3 includes the feature to resume from break-point, it will save the current training results into checkpoints per 100K steps as default. The checkpoints will include the design distribution, control policy at the time and the active robot queue to be continually trained. The co-optimization framework (sec 3.3) also includes an evaluation launcher, which will read through all the checkpoints and record videos on unreduced model and reduced

model of user's choice per checkpoint. The evaluation launcher will also record the average reward as well as analyze the pressure command policy output. The pressure commands are shown in the sequence of the legs, it is in the format of 8 Plotly plots and one plot will be left blank if no leg is attached to the corresponded slot. Therefore, the design of the robot can be easier verified with the pressure command output and the recorded policy videos. The plot can be downloaded as JSON scripts with Plotly formats, which can be then analyzed and exported as arrays of 20 entries; each entry represents the pressure command at the corresponded time. The array can be easily entered into the Raspberry Pi pressure control interface for the regulator to control the regulator output. As the experiment focuses solely on the design and control pair from the final training step, the final checkpoints are extracted for evaluation.

Fig. 4.2 shows the linear pressure scaling used to align the PneuNet bending response between simulation and reality. To calibrate the action scaling, the bending of a single PneuNet under pressures from 10 kPa to 100 kPa were recorded in increments of 10 kPa. The pressures in simulation which achieve an equivalent bend were then searched in simulation and a linear function to the results were fitted.

4.2.2 Test Surface and Robot Starting Position

In this project, three surfaces were used to test the performance of domain randomization on surface friction. The three surfaces include a silicone mat, cardboard, and a table

surface. Among these surfaces, the silicone mat has the lowest friction coefficient, around 0.6, cardboard has a friction coefficient of about 1.44, and the table surface has a friction coefficient well beyond the μ_{rnd}, reaching 2.64. The friction coefficients of the surfaces were tested by recording the force required to slide the PneuNet actuator across them with mass applied perpendicular to the surface. The testing surfaces were selected such that the first two, with low friction, cover the lowest and highest friction levels in μ_{rand}. The table surface, being on the higher end of the friction spectrum, could potentially indicate whether domain randomization can generalize the control policy design, making the design and control pair less sensitive to surface roughness.

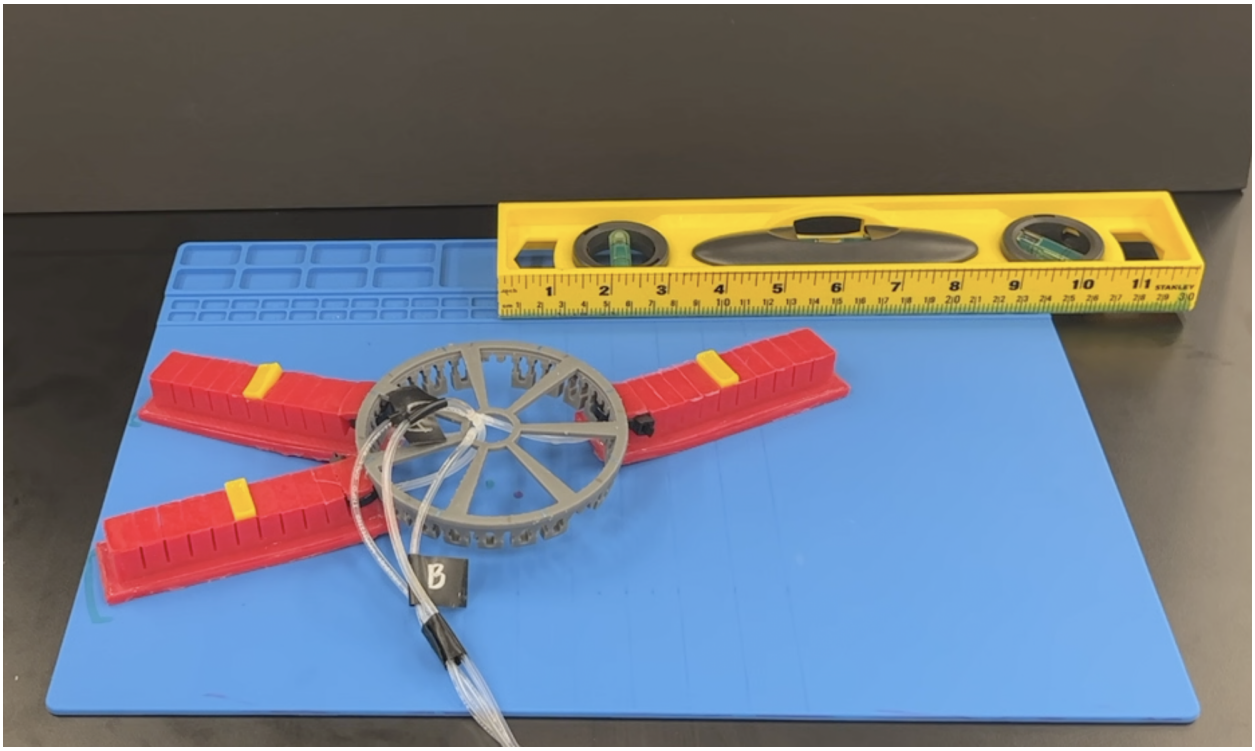

Figure 4.3: The starting position of the robot on silicone mat

Given the size constraint of the silicone mat relative to the robot (Fig.4.3), the robot is positioned at the edge of the silicone mat to prevent it from leaving the mat during the 20-second action duration. In Fig. 4.3, the starting position of the robot is clearly marked, with reward indication lines drawn for easier assessment of the reward. A starting position marker and reward indication lines are also drawn on the cardboard and table. All reward readings are taken with respect to the center of the disk.

4.3 Sources and Mitigation of Error

Several difficulties encountered during the experiment necessitated repeated experiments to obtain reliable data. This section focuses on these issues and explains the sources of error in the experiment.

4.3.1 Pressure Regulator Operational Range and Calibration

Section 4.1 explained the detailed setup of the experiment; the pressure regulators used in the setup are Enfield TR-010-gs, (the data sheet: `https://www.enfieldtech.com/site/Product-Documentation/TR-Electronic-Pressure-Regulator-Datasheet.pdf`). These valves have an internal built-in pressure sensor and an operational pressure range from 0 to 10 bar. In this project, the commanded pressure input to the PneuNet actuators are between 0 to 0.9 bar, at the lower part of the range of operation; which is noisier than the center of the range. The proportional-integral-derivative gains of the valves were tuned, enabling

them to track pressure step changes within 1 second and control the pressure overshoot to a reasonable extent.

After tuning the PID controller gain, a systematic pressure difference were observed; the downstream Honeywell sensor consistently record a 20 to 30 percent higher pressure value. Therefore, a knock-down factor is also applied in the Raspberry Pi controller script to prevent overshooting. During the tuning process, unregulated exhaust from the controller sometimes creates unexpected turbulent flows, adding noise to the feedback pressure signal. The exhausts of all three valves are connected to a 1/16 to 1/4 barb to act as a muffler.

4.3.2 Friction and Dust

Subsection 4.2.2 explained how the friction coefficients of the testing surfaces are measured. As the research project is highly dependent on stick-slip behavior of the PneuNet actuators on the testing surfaces, slight variance in friction coefficient could have a significant impact on the experiment result. As dust continuously accumulates on the testing surfaces, PneuNet actuators inevitably collect more dust over time as the experiment progresses. While the intricate surface geometry of PneuNet actuators already makes dust removal a challenging task, this issue is further complicated by the ongoing settling of ambient dust on the actuators during their operation. To maintain a consistent level of dust, dry rags are used to clean off excess dust from PneuNet actuator surfaces before each experiment.

4.3.3 Measurement Error

The sim to real transition was evaluated by comparing simulated forward progress to experimentally measured forward progress, which is the reading of the centre of the disk projected on the yellow ruler (Fig. 4.3). The central hole on the disk is rather big, causing approximately an 0.5 cm random error on sim-to-real experiment reward readings per trial.

4.4 Evaluation of Sim-to-Real Results

Table 3.1 lists all the training environments. Among the 12 different configurations, entries 7 and 11 are each trained three times. Of the total 16 training sessions, they were categorized into the following categories regardless of the seed they were trained with.

	Reduced Model	Co-Opt or Design	DR or fixed
1	High fidelity model	L1	fixed
2	High fidelity model	L1	DR
3	High fidelity model	Baseline	fixed
4	High fidelity model	Baseline	DR
5	High fidelity model	Co-opt	fixed
6	High fidelity model	Co-opt	DR
7	Low fidelity model	Co-opt	fixed
8	Low fidelity model	Co-opt	DR

Table 4.1: List of all training categories. (DR stand for friction domain randomized and fixed stand for trained on specified friction level; Co-Opt stand for co-optimized training and design stand for the specific design that the controller policy was trained on)

By consolidating all 16 training sessions into the 8 categories shown in Table 4.1, and

calculating the mean, error bars, root mean square error and quartiles of simulation and real-life experiment rewards, Chapter 6 presents the results and conclusions on the effectiveness of domain randomization.

Chapter 5

Results

After all the sim-to-real experiments are done, the simulated and real world testing reward are gathered and analyzed. The results from Scaff et al. [53] were reproduced.

5.1 Results

Among all the final design and control pairs obtained in all the training configurations, two sets of design and control pairs were selected and displayed along side with the expert designed baseline robot in 5.1. Both trained robots have shown different use of mechanical intelligence and utilized slip-stick behavior of friction in a smart way.

Robot trained with τ_{high}, μ_{rand}, Seed 0 run 1: The robot has used a three legged movement pattern with a periodic gait(Since neural open-loop controller is used, the pressure

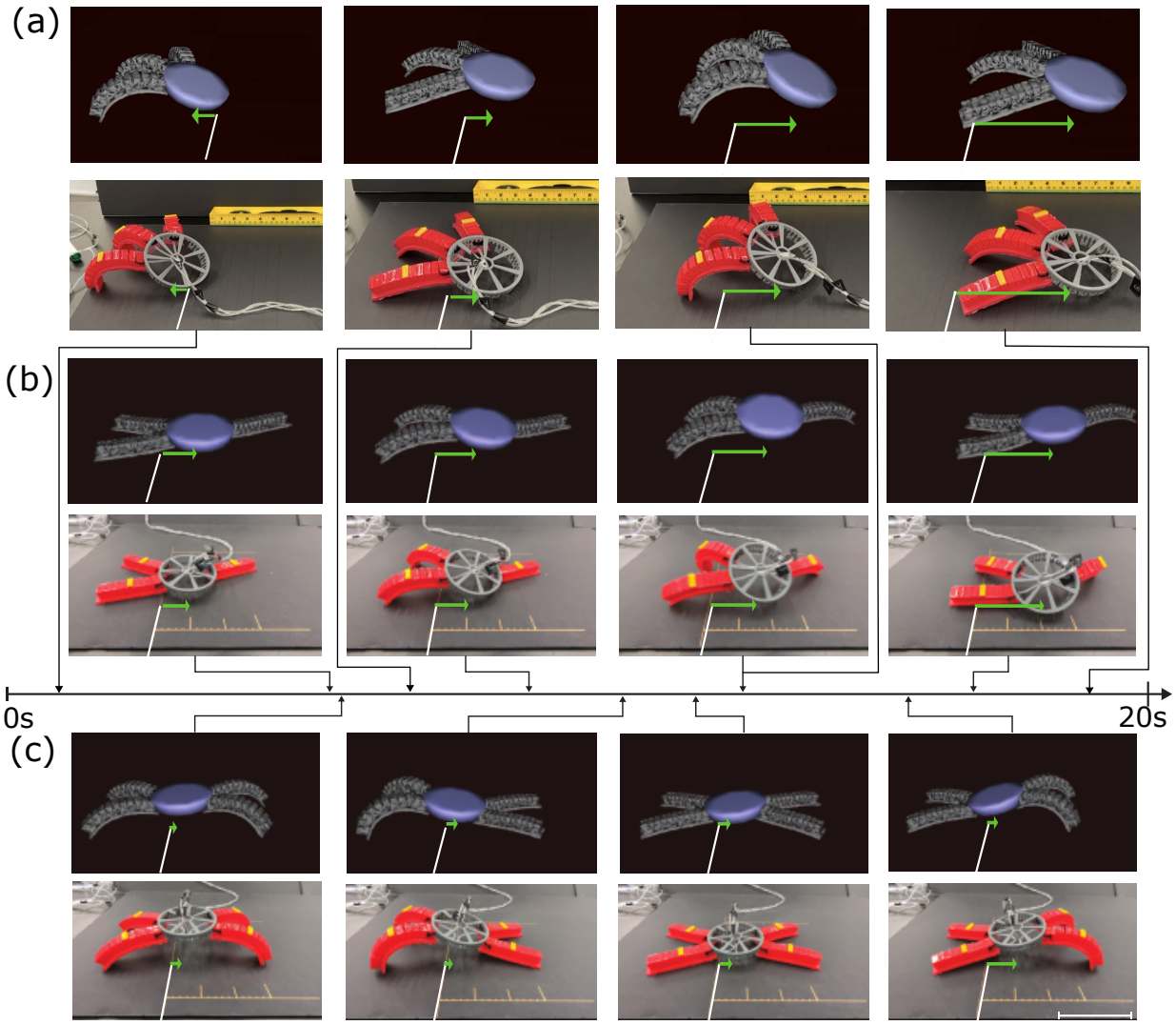

Figure 5.1: The co-optimization framework jointly learns the design and control of crawling soft robots (top 2) that outperform an expert-designed baseline (bottom). While trained exclusively in simulation, our learned robots are capable of zero-shot sim-to-real transfer, with the optimal design moving more than 2× faster than the baseline in the real world. (a) Robot trained with τ_{hi}, μ_{rand}, Seed 0. (b) trained with τ_{low}, μ_{fix}, Seed 0. (c) Baseline: design and control developed by expert.

profile from evaluation show slight difference, but the actuation pattern in general show periodicity). It has a rapid actuation frequency with a time period of 2 seconds. The central back leg acts as a pivoting point, supporting the robot and provide interesting frictional behavior change for the two outer legs allowing them to slip on the surface when contracting forward and stick on the surface when pushing forward. Slip-stick behavior causes the robot to have a boost in forward progression when the two outer legs are pushing.

Robot trained with τ_{low}, μ_{fix}, Seed 0 run 1: The robot employs a 'pivoting' behaviour that allows for large bursts of forward sliding. Due to asymmetry, the robot roll sideways 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. Although the gaits seems to be very effective across the range of friction, the actuation frequency is slower than the former robot with a time period of 3 seconds, causing the robot to have a lower overall reward at low friction surfaces.

In this project, two different MOR models are trained with and without frictional domain randomization in the SOFA simulator. The resulting design and control pair are tested on 3 different surfaces in sim-to-real transfer experiments to investigate the effect of MOR tolerance τ and friction domain randomization on task performance and transfer error. Since frictional error accrues over time, the reward are normalized by the episode length (20 sec) and the results are shown in figure 5.2(a-d). In reality, error accumulated over five individual independent tests in which robots may receive different reward. In simulation, some variation

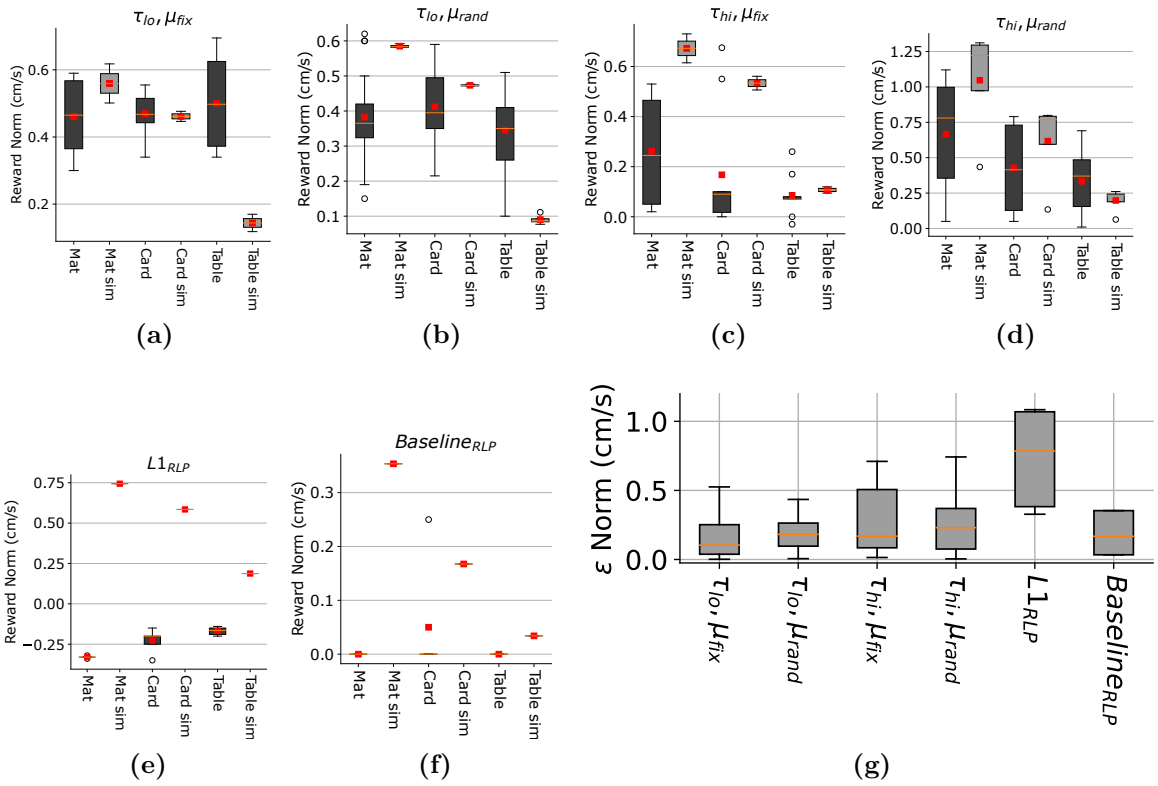

Figure 5.2: (a)-(f) Sim and real reward for final trained robots in the four simulation settings $\tau_{lo/hi}$ and $\mu_{fix/rand}$; and the L1 robot design and the expert baseline design with trained policy only. Mean is red square; median orange line; IQR shaded box. (g) Mean normalized sim-to-real transfer error for same six robots.

is due to robots trained under same category but using different random seeds.

The robot L1, when trained with τ_{low}, μ_{fix}, and the final robot trained under τ_{low}, μ_{rand} conditions, yield comparable rewards in a simulated environment. However, in real-world applications, the former exhibits superior transferability. The strategy used in the latter scenario (τ_{low}, μ_{rand}) is characterized by a smaller number of extended steps. While this approach minimizes the impact of friction, it potentially heightens the susceptibility to

perturbations in inertia, such as external moments induced by the tubes.

The highest overall reward in simulation is achieved by the training session with τ_{hi}, μ_{rand}. However, the real and simulated reward are separated greater than 0.25cm/s except in the case of robot co-optimized with $\tau_{high}, \mu_{rand}, s0$, the gap of which lies within 0.075cm/s.

Simulated and real reward for policies trained on fixed designs(the L1 design and the four-legged expert baseline design) are show in in figure 5.2(e-f). These were only trained with one random seed and therefore less variation in reward is shown. The transfer-ability are bad on all of the models although rewards in simulation are higher, often lies on the outside of 1.5x of the interquartile range.

Transfer error calculated through all trials for each setting of which robots were trained are shown in figure 5.2(g). Variations in transfer errors stem from disparities in the real rewards obtained throughout the five trials of each real robot experiment, as well as fluctuations in the simulated rewards of gaits trained under different random seeds. Additional sources of errors include disagreements between rewards recorded in real-world experiments and simulations. This type of error is predominant among all sources of errors, and it directly indicates whether the design and control pairs learned in simulation under each category transfer well enough to the real world.

In addition to the sim-to-real transfer results, the transfer from the higher MOR tolerance to higher-fidelity τ_{lo} environment in SOFA simulation environment were also analyzed. The

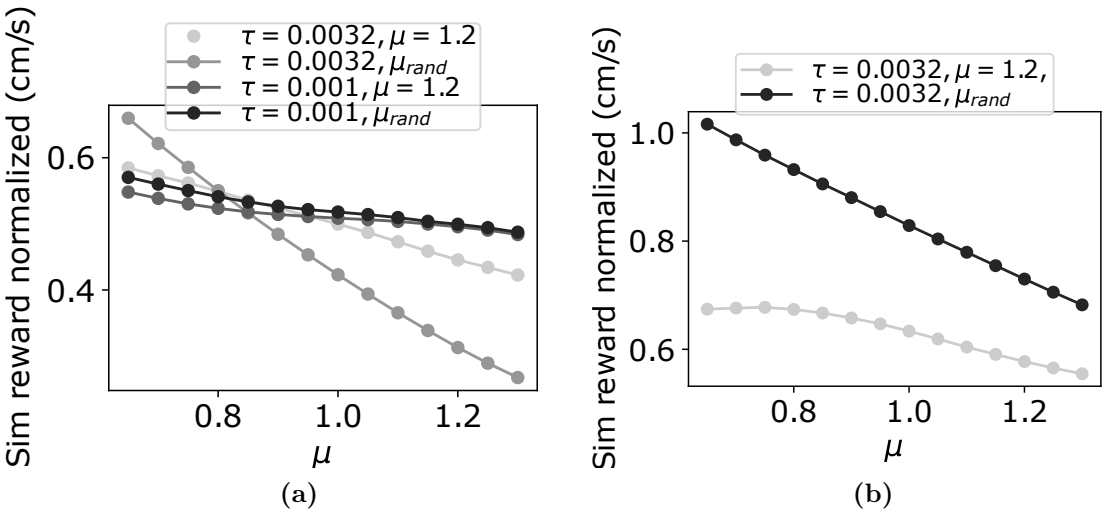

Figure 5.3: (a) Mean normalized reward at $\mu = \{0.65, 0.7, ..., 1.3\}$ for all co-optimized robots in τ_{low} environment. (b) Same mean normalized reward for robots co-optimized with τ_{hi}, evaluated in τ_{hi} environment.

reward normalized by episode length for robots trained in each environment are shown in figure 5.3. Despite the minor discrepancies in position error observed between the two tolerances in the heuristic animations, the gaits that are ultimately trained with τ_{hi} demonstrate a significant transfer error when applied to the τ_{low} simulation environment. Specifically, the robot trained under the conditions of τ_{hi} and μ_{rand} experiences a peak error that surpasses 0.4cm/s in environments characterized by low friction.) Moreover, Figure 5.3 highlights a consistent trend of increasing reward with respect to μ, with the sole exception being the scenario where τ_{hi} and μ_{fix} are evaluated in the τ_{hi} environment.

Chapter 6

Discussion and Conclusion

6.1 Discussion

6.1.1 Effect of Co-Optimization on Sim-to-Real Transfer

The effect of co-optimization is significant for improving the robustness of the robot when transferring from simulation to reality. Fig 5.1 has shown that the robot trained under τ_{low}, μ_{fix} (b) have obtained a design and control pair that achieves much higher reward (green arrow) in simulation and real world experiment compared to our expert designed baseline robot(c). The co-optimized robot (b) has discovered a interesting asymmetrical design which allows the controller to pivots the robot and allowing the robot to burst forward. Further, the robot trained under τ_{hi}, μ_{rand} (c), even further exceeds the performance of (b).

We further show that training in the joint space of design and control can be advantageous over training a controller on the expert-developed design or the L1 design are further demonstrated; here, policies trained through co-optimization achieve higher reward and stronger transfer in most cases. Although, it is possible that the longer training time from the co-optimized sessions or other parameter tuning caused this. In general, gaits trained through co-optimization are qualitatively different than those trained of fixed designs.

L1 coopt vs L1 non-coopt: By comparing the L1 robot (b) with the non co-optimized L1 robot, although they both achieved a rather good and comparable simulated reward, the sim-to-real transfer of the non co-optimized robot was unsuccessful. The lack of fore-leg actuation made the robot to stick on the surface and received a negative reward. This comparison have demonstrated the ability of open-loop neural controllers to learn more robust designs towards real-world friction and actuation perturbations and overcome model errors by exposing it to more robot morphologies.

Compared to voxel-based design spaces [39], the design space in this project is smaller. The effects that might arise by having a more refined design space (e.g., FEA element-level rather than actuator level, we note that modularization at the actuator level seems useful for practitioners with access to standard fabrication tools.) still remain unclear.

6.1.2 Effect of Tolerance of Reduced Models on Sim-to-Real Transfer

In spite of having a high tolerance, the robot trained under τ_{hi}, μ_{rand} with **seed 0** and **1st trial** (robot (b) from Fig. 5.1) achieved a good sim-to-real result (although the robots trained with the high tolerance models in general have a higher variance and less robust in sim-to-real transfer experiments) . However, it's worth noting that when the models were replaced with low-tolerance models and re-simulated, the real-world physics where not revealed. The reward in evaluation significantly drops when the design and control pair is evaluated on the high-fidelity model compared to the pair being evaluated on the low-fidelity model. The physics of the design-control pair succeeded sim-to-real transfer with high accuracy (lies within 0.075 cm/s). This discovery have suggested that by solely having a test shown in Fig. 3.1 is not enough, the causes of error in a non-convex multi-dimensional space of a reduced model in the simulation is also multi-dimensional and decoupling the sources of error is not feasible. A simplified 1-dimensional error metrics is used in the previous simulations to model the performance between different reduced models which resulted in similar amount of error between high tolerance and low tolerance models (Fig. 3.1). Yet, the sources of error might be different, resulting in a large gap of performance in the trained design and control pair in the simulation. This underscores the importance of thorough testing of reduced models before their use in simulators, as some high-tolerance models align better with real-world physics.

6.1.3 Effect of Domain Randomization on Sim-to-Real and Robot Performance

Previous discussion (Sec. 6.1.1) has shown that stick-slip transitions are the main source for both the performance and error of the robot rewards, domain randomization on the Coulomb friction parameter are performed to seek for improvement on the ability to carry out sim-to-real transfer. Randomization of friction coefficient μ are carried out to evaluate the performance of robots with different MOR models before and after the technique in simulation and in reality. During sim-to-real transfer experiments, the performance of the design and control pairs are tested for surfaces within the range of μ_{rand}(silicone mat, foam board) and outside it(table).

Domain randomization positively impacts co-optimization, enabling the discovery of robust and versatile designs and control strategies on τ_{hi} models. When trained under μ_{fix} environments, the transferability of the design and control pair is sub-par compare to models trained under μ_{rand} environments. Although gaits obtained from μ_{rand} training environments seems to be more sensitive to friction and having a rather big variance (1 final robot had complete different design compare to other 3 final robots trained under same setting, with lower simulated and real-world reward), it successfully discovered the robot (c) from Fig. 5.1.

Among all the combinations studied in this project, domain randomization on solely friction

coefficients does not seem to support transfer from simulation to real world. All training settings achieved comparable transfer error across all surfaces and the potential reasons are concluded by analyzing the learned gaits. One robot trained under τ_{lo}, μ_{rand} had learned a gait that is less sensitive to friction in the simulator, which appears to use longer, precarious steps that fail more easily in reality. The robots trained under τ_{hi}, μ_{rand} have achieved the lowest variance in reward across different surfaces by attempting a variety of step types. The model trained under τ_{hi} accrues significant error when transferring into the τ_{lo} environment. Although some training sessions have found robot gait that is faster compare to L1 robot in reality while testing on surfaces within the range of μ_{rand}, there also exist results that achieves low reward across all surfaces while transferring.

The effect of physical perturbations is most often demonstrated by changes in discrete set of values such as joint torques whereas perturbations of a soft robot could have effect on its motion anywhere along its continuous body. The larger state space of the soft system makes sim-to-real transfer more challenging compare to rigid bodies when using domain randomization over a single parameter. The selection of parameters to randomize is crucial and difficult; reducing the sensitivity to one parameter might also increase the sensitivity to another. With in the simulator, there exist 11 physical parameters that could be randomized and future work may include the exploration of randomization parameters.

6.2 Conclusion

This work demonstrated the impact of co-optimization, reduced models as well as domain randomization when using the framework proposed by Charles Schaff et al.[16] optimizing the control or co-optimizing the design and control of modular, PneuNet-based crawling soft robots capable of sim-to-real transfer. A model-free algorithm for co-optimization together with a method for creating reconfigurable, high-fidelity reduced-order models were presented, allowing the algorithm to efficiently optimize over designs with different topologies while preserving realism. A series of sim-to-real experiments are conducted to demonstrate the effectiveness of the co-optimization on robustness of transfer of the robots trained comparing to the trained only robots or expert designed baseline in a soft robot locomotion task. Two different reduced models were used in the training to evaluate the performance of domain randomization and it is discovered that the real-world fidelity of reduced models does not solely correlate with tolerance levels. High-tolerance models also unveiled physical phenomena in simulation that were not apparent in the low-tolerance models. Additionally, it has been found that although domain randomization indeed positively impacts co-optimization on high-tolerance model, allowing it to discover better design and control pairs in simulation and real-world compare to trained under fixed friction setting; its effectiveness towards sim-to-real transferability is not significant when domain randomized towards friction μ only.

Soft robotic being an emerging field. The compliant nature of soft robots made it hard to be designed and controlled. This work has demonstrated the power of reinforcement learning and evolutionary co-optimization algorithms on design and control of soft robotic crawlers. Although the effect of domain randomization is not significant on sim-to-real transfer, it is planned to explore adaptations of this framework with additional sim-to-real techniques (e.g., domain randomization of multiple variables) and to different design and control spaces. In future work, high-precision pressure regulators will be used across the experimental operating range to minimize pressure discrepancies and ensure better alignment with the pressure input curve in simulations.

In conclusion, our study underscores the potential impact of co-optimization methods in design and control of soft robotic crawlers. This revolutionary algorithm has the capability to greatly reduce the human effort required for creating robots tailored to specific tasks, offering the prospect of highly adaptive robots designed to meet human needs efficiently. Through the capability to automatically design versatile and adaptable soft robots, this work promises real-world applications that boast improved performance and functionality. These implications reach beyond the scope of this study, offering the exciting prospect of progress and innovation in the field of soft robotics.

Bibliography

[1] F. Ilievski, A. D. Mazzeo, R. F. Shepherd, X. Chen, and G. M. Whitesides, "Soft robotics for chemists," *Angewandte Chemie*, vol. 123, no. 8, pp. 1930–1935, 2011.

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

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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*, pp. 315–323, 2014.

[10] 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.

[11] 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.

[12] 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.

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

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

[15] 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.

[16] C. Schaff, A. Sedal, and M. R. Walter, "Soft robots learn to crawl: Jointly optimizing design and control with sim-to-real transfer," 2022.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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," *Soft tissue biomechanical modeling for computer assisted surgery*, pp. 283–321, 2012.

[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, 2017.

[23] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30, IEEE, 2017.

[24] 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.

[25] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A survey on learning-based robotic grasping," *Current Robotics Reports*, vol. 1, pp. 239–249, 2020.

[26] H. Ju, R. Juan, R. Gomez, K. Nakamura, and G. Li, "Transferring policy of deep reinforcement learning from simulation to reality for robotics," *Nature Machine Intelligence*, pp. 1–11, 2022.

[27] 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.

[28] 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.

[29] R. Shepherd, F. Ilievski, W. Choi, S. Morin, A. Stokes, A. Mazzeo, X. Chen, M. Wang, and G. Whitesides, "Multigait soft robot," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 108, pp. 20400–3, 11 2011.

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

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

[32] 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.

[33] 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.

[34] 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.

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

[36] 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.

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

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

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

[44] 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.

[45] 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.

[46] C. Majidi, R. F. Shepherd, R. K. Kramer, G. M. Whitesides, and R. J. Wood, "Influence of surface traction on soft robot undulation," *The International Journal of Robotics Research*, vol. 32, no. 13, pp. 1577–1584, 2013.

[47] J. Z. Zhang, Y. Zhang, P. Ma, E. Nava, T. Du, P. Arm, W. Matusik, and R. K. Katzschmann, "Sim2real for soft robotic fish via differentiable simulation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 12598–12605, IEEE, 2022.

[48] M. Dubied, M. Y. Michelis, A. Spielberg, and R. K. Katzschmann, "Sim-to-real for soft robots using differentiable fem: Recipes for meshing, damping, and actuation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5015–5022, 2022.

[49] A. Centurelli, L. Arleo, A. Rizzo, S. Tolu, C. Laschi, and E. Falotico, "Closed-loop dynamic control of a soft manipulator using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4741–4748, 2022.

[50] Y. Li, X. Wang, and K.-W. Kwok, "Towards adaptive continuous control of soft robotic manipulator using reinforcement learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7074–7081, IEEE, 2022.

[51] G. Tiboni, A. Protopapa, T. Tommasi, and G. Averta, "Domain randomization for

robust, affordable and effective closed-loop control of soft robots," *arXiv preprint arXiv:2303.04136*, 2023.

[52] 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.

[53] C. Schaff, A. Sedal, S. Ni, and M. R. Walter, "Sim-to-real transfer of co-optimized soft robot crawlers," *Autonomous Robots*, pp. 1–17, 2023.