DISSERTATION

MODULAR RECONFIGURABLE ROBOTS CAPABLE OF PROGRAMMABLE SHAPES AND MOTIONS

Submitted by

Zhe Chen

Department of Mechanical Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2025

Doctoral Committee:

Advisor: Jianguo Zhao

Kirk McGilvray
Azer Yalin
Anthony Maciejewski

ABSTRACT

MODULAR RECONFIGURABLE ROBOTS CAPABLE OF PROGRAMMABLE SHAPES AND MOTIONS

For robots to successfully locomote in different environments, it's better to equip them with multiple modes of locomotion. Multimodal locomotion for existing robotic systems is generally realized by integrating multiple mechanisms into a single robot. These reconfigurable robots are generally cumbersome or challenging to control and actuate. The adaptability of these robots is also limited after fabrication. Also, in the field of soft robotics, shape morphing structures actuated by external stimuli are utilized to enable reconfigurable and multi-functional robots. Since the soft robots feature continuous and large deformation, the behavior of these robots is not accurately predictable or controllable.

We propose to realize the reconfigurable robots by stacking multiple reconfigurable modules in series and using a few tendons for actuation. The shape and motion of the reconfigurable modules can be adjusted upon tendon actuation. These modular reconfigurable robots offer advantages such as ease of fabrication, extensive workspace, predictable behavior, and simplified control.

First, we present a novel reconfigurable module inspired by the Kresling origami pattern. The origami module is equipped with joints that can independently transition between soft and rigid states, enabling the module to adapt its behavior during actuation. To understand the reconfiguration capability, we numerically analyze the programmable shapes and motions of a single origami module. We develop a reconfigurable robot with four legs, each made from four serially connected modules. The robot can walk, crawl, and inch using the same mechanical structure.

We also realize a reconfigurable robot based on reconfigurable bistable module. Bistable modules have the ability to rapidly switch between multiple stable states and hold any stable state without requiring external energy input. Traditional bistable modules generally have a fixed struc-

ture after fabrication, leading to a fixed behavior when actuated. To make the bistable modules reconfigurable, we investigate adjusting the behavior of the bistable modules with shape morphing beams and adjustable springs. A forward model to predict the module's behavior is presented and validated. We also address the inverse problem to achieve a desired behavior by choosing proper parameters of the reconfiguring process. By stacking two modules in series, we realize a reconfigurable arm capable of generating different trajectories and a reconfigurable crawler which can crawl in different directions.

Both the reconfiguring parameters (e.g., stiffnesses of the joints of the origami modules) and the control algorithm (tendon displacements) will influence the resulting shape and motion of the robots. We manipulate the origami-module based and bistable-module based reconfigurable robots by empirically choosing the appropriate reconfiguring parameters and control algorithm. For more complicated task, the reconfiguring parameters and control algorithm should be co-optimized, since the reconfiguring parameters would influence the kinematics or dynamics of the robots, resulting in varying control algorithms. Here, we employ reinforcement learning based co-optimization framework on the robot to obtain an optimal combination of reconfiguring parameters and corresponding control policy. The task investigated in this work is to make the end effector of a manipulator composed of multiple serially connected origami modules reach a specific goal point, with optional obstacle avoidance. We further deploy the learned parameters and control policy on a two-module prototype performing a reaching task to validate the effectiveness of the proposed co-optimization framework.

This work demonstrates a unified approach to designing and controlling modular reconfigurable robots, paving the way for future robotic systems capable of autonomous adaptation and multifunctional operation.

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to the friends and colleagues whose support has been instrumental in making this work possible.

First and foremost, I am deeply thankful to my advisor, Dr. Jianguo Zhao, for his exceptional guidance, unwavering support, and significant investment of time and energy in my research. His intellectual brilliance and dedication have been invaluable throughout my PhD journey. He has a tremendous passion for research and consistently stays updated with the latest advancements in the field of robotics. Dr. Zhao is among the most hardworking and focused individuals I have ever had the privilege to know. I aspire to emulate his excellence in my future career.

I am also deeply grateful to my dissertation committee members — Dr. Anthony Maciejewski, Dr. Kirk McGilvray, and Dr. Azer Yalin — for their dedication and support as members of my committee, their thorough review of this work, and their invaluable feedback.

I thank all previous and current members of the Adaptive Robotics Lab (in no particular order): Dr. Haijie Zhang, Dr. Jiefeng Sun, Brandon Tighe, Ben Pawlowski, Billy Hsiao, Ajai Singh, Sydney Spiegel, Bryce Jones, Clint Middlemist, Elisha Lerner, Andrew Stefka, Josh Chrisler, Alex Leadbetter, Mahmud Saikot, and Sudheera Kariyawasam. You have created a wonderful work environment and have been great friends. Brandon Tighe, Ben Pawlowski, and Dr. Jiefeng Sun provided tremendous assistance in the origami robots project. I am also grateful to Dr. Jiefeng Sun for his insightful discussions and valuable suggestions on the bistable modules. I am greatly thankful to Dr. Hao Zhang and Li Chen from the University of Massachusetts Amherst for their valuable and insightful suggestions on the co-optimization for reconfigurable robots.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Biological creatures (animals or insects) can successfully locomote in unstructured environments using multiple modes of locomotion through actively changing their morphology (i.e., body/leg configurations) [1, 2]. For instance, a frog can slowly walk on the ground, rapidly jump from the lotus, swim in the water, or even climb up on trees, all with the same body and legs but different configurations.

Inspired by biological creatures, roboticists have designed various reconfigurable robots with multimodal locomotion capabilities [3]. Examples include jumping and wheeling [4, 5], flying and rolling [6, 7], walking and jumping [8], etc. Nevertheless, unlike their biological counterparts, most existing reconfigurable designs generally require different mechanical mechanisms to accomplish different locomotion modes. Recently, in the field of soft robotics, shape morphing structures actuated by external stimuli are utilized to enable reconfigurable and multi-functional robots. Since soft robots feature continuous and large deformation, their behavior is not accurately predictable or controllable.

We propose to realize reconfigurable robots by stacking multiple reconfigurable modules in series and using a few tendons for actuation. These modular reconfigurable robots offer advantages such as ease of fabrication, extensive workspace, predictable behavior, and simplified control. To enable these capabilities, we investigate two distinct reconfigurable modular mechanisms: origami-inspired modules and bistability-based modules. For origami-inspired modules, reconfigurability is achieved through variable-stiffness joints that can independently transition between soft and rigid states. For bistability-based modules, reconfigurability is realized by adjusting the energy landscapes of the bistable modules. Since both mechanisms have parameters that can be reconfigured on the fly, and these reconfiguration parameters influence the kinematics and dynamics of the robots, we develop a co-optimization framework using reinforcement learning techniques to simultaneously optimize both the reconfiguration parameters and control policies for specific

tasks. To provide context for our work, we review the state-of-the-art in three related research areas. We begin with origami module based reconfigurable robots, followed by bistable module based reconfigurable robots, and conclude with co-optimization approaches for robot design and control.

1.1 Origami Module Based Reconfigurable Robots

Using origami for reconfigurable robots has been recently investigated due to its advantage of shape-changing ability, fast fabrication, and bi-stability [9], [10]. Zhakypov et al. developed an origami robot, Tribot, which can switch between jumping and crawling without altering the mechanism [11, 12]. Miyashita et al. used origami to develop exoskeletons for different modes of locomotion under magnetic actuation [13]. Baek et al. leveraged compliant origami to mimic the folding wings of beetle for both jumping and gliding [14].

Among the many fascinating origami patterns, the Kresling pattern stands out due to its modular design, bistable behavior, and capacity for significant deformation. We can easily connect multiple Kresling modules in series to realize a chained mechanism, e.g., a robotic arm. Kaufmann et al. [15] constructed a robotic arm by assembling Kresling origami modules, the articulation of which can be reconfigured. A pipe climbing robot has been built based on the same origami modules [16]. Jin et al. [17] developed an origami-inspired soft actuator based on the Kresling pattern and built a reconfigurable crawling robot using that soft actuator. Pagano et al. [18] designed a crawling robot based on the Kresling pattern which is actuated by a DC motor. Lee et al. [19] developed an underactuated robotic gripper consisting of multiple modules connected in series, actuated by a cable routed through the modules. Wu et al. [20] used a magnetic field to actuate an origami-inspired manipulator, laying a foundation for developing untethered multi-functional robots. Nevertheless, most of these designs generally cannot reconfigure the mechanical structure or responses of the origami on the fly.

Element with electrically adjustable stiffnesses is a promising way to reconfigure the behavior of the origami modules on the fly, since the behavior of the module is affected by the stiffnesses

of its components (e.g., creases, facets). For instance, Firouzeh et al. [21] designed an underactuated gripper which can realize multiple grasping modes by changing the stiffness of various joints with heating elements. Zappetti et al. [22] developed a variable-stiffness tensegrity structure which can change its stiffness by heating and melting the low melting point alloys encapsulated in the variable-stiffness cables. Yang et al. [23] built a variable-stiffness robotic finger, in which shape memory polymer is used to modulate the stiffness of the finger with the Joule heating changing the temperature of the shape memory polymer material. Wang et al. [24] also used shape memory polymer as the variable-stiffness material to build a soft finger which can realize various trajectories by selectively changing the stiffness of different segments of the finger. Although extensive work on variable stiffness elements exists, it remains underexplored on how to incorporate variable stiffness joints into mechanical mechanisms to generate programmable shapes or motions [25], enabling reconfigurable modules and robots.

1.2 Bistable Module Based Reconfigurable Robots

Multistable structures, characterized by their ability to rapidly switch between multiple stable states, represent an emerging area of study in robotic or mechatronic systems. These structures, after switching to a stable state, can maintain the new stable configuration without any external energy input. A special type of multistable structure is a bistable module with two stable states, which has been observed widely in nature. For instance, the Venus flytrap's leaves are bistable, which can snap shut in 0.1 s to capture prey [26]. Similarly, the rapid beak closure of hummingbirds is another instance of natural bistability, enabling these birds to catch flying insects efficiently [27].

Researchers have employed bistable modules to realize reconfigurable robots with multiple functionalities [8, 28, 29]. Compared to origami module based reconfigurable robots, bistable module based robots have the advantages of fast response [30] and enlarged output force [31, 32]. Furthermore, bistable module based reconfigurable robots can maintain a specific shape without requiring external energy input.

For multistable or bistable modules, an important characteristic is the energy landscape (EL): how the stored strain energy varies with respect to the deformation (e.g., bending angle). The EL can determine both the stable configurations and the dynamic responses of the module. Despite substantial recent research, existing multistable modules are generally fixed after being fabricated, leading to a predetermined EL [29, 30], constraining their adaptability, reconfigurability, and application scope.

To extend the reconfigurability of bistable modules, researchers have recently investigated how varying specific parameters of the modules will influence their ELs, thereby generating different behaviors (stable configurations or dynamic responses) when actuated [33–35]. For instance, researchers realized adjusting the ELs of the bistable modules by changing the initial bending angles of the flexible beams [36], controlling the voltages applied to the twisted and coiled polymer fibers or manually altering the prestretch of the elastic cords [37], or tuning the magnetic field applied to the bistable module [38].

Despite recent progress in adjusting the ELs for bistable modules, two issues remain largely unaddressed. First, most of the existing work, with notable exceptions [37, 38], cannot adjust the ELs on the fly. Current approaches typically require manual alterations, such as the manual modification of the initial lengths of spring elements [32, 39, 40], or the manual adjustment of the initial bending angle of the bistable module by changing the dimension of its bending beam [41, 42]. It would be more advantageous if we could tune the EL on the fly to enhance the reconfigurability of bistable modules or robots. Second, there are few systematic explorations on how to achieve the desired ELs by choosing proper parameters of the bistable modules. Instead, most existing work predominantly conducts simulations or experiments to observe how different parameters will influence the ELs [43–46].

1.3 Co-Optimization of Design and Control

For reconfigurable robots, it is essential to co-optimize both the design (including reconfiguration parameters) and the control algorithm, as variations in design can result in different kinematics,

dynamics, or functionalities of the robots, thus significantly affecting their control strategy. Robots with different designs may have entirely different control strategies. In some cases, certain tasks may even become infeasible with unfavorable designs. Consequently, designing a robot requires simultaneous optimization of both its physical design and control algorithm. To achieve this, a co-optimization process should be employed, aiming to obtain an optimal combination of design and control tailored to the specific environment and task requirements.

Traditionally, co-optimization methods are developed based on the dynamics model of the robots [47–51]. The optimization of design or reconfiguring parameters can be merged with the optimization of control algorithm [48, 50]. Also, an iterative process can be used to update design parameters and control algorithm separately and alternatively [51].

In recent years, reinforcement learning (RL) has emerged as a powerful alternative to traditional model-based methods, which often depend on detailed mathematical models and intricate algorithm design. Reinforcement learning leverages empirical data to design and optimize control strategies for robots and manipulators. RL enables robots to learn control policies through interaction with their environment. By maximizing cumulative rewards based on trial and error, RL tries to learn a optimal control policy, generally implemented as a neural network, to map the state input to action output to achieve specific tasks [52] [53]. Despite challenges like computational intensity and lack of interpretability, data-driven methods, particularly RL, are now widely applied in tasks such as robotic grasping, motion planning, and adaptive control, showcasing their potential to handle complex, real-world challenges in robotics. RL technique is especially useful for complex robotic systems where dynamics are difficult to model accurately, such as soft robots [54] [55] or multi-degree-of-freedom manipulators [56] [57] [58].

Hoeller et al. [59] proposed a reinforcement learning approach to training four-legged robots which can navigate challenging scenarios that are reminiscent of parkour tasks. Though the policy was trained from simulation data only, it can be successfully transferred to real world legged robots. Nagabandi et al. [60] developed deep dynamics model, based on model-based RL, to achieve a suite of dexterous manipulation tasks, such as in-hand reorientation and handwriting.

Recently, researchers have also explored implementing co-optimization using reinforcement learning methods [61–66]. In addition to ease of implementation, these studies show that RL-based co-optimization methods can identify optimal designs and control algorithms, outperforming traditional co-optimization methods. In this work, we try to employ RL-based co-optimization techniques to co-optimize the reconfigurable robots for some given tasks.

1.4 Outline and Contribution of this Research

Chapter 2: Origami Module Based Reconfigurable Robots

We present a novel reconfigurable origami-inspired module with shape-morphing and variable-stiffness joints. By adjusting the stiffnesses of the joints in the module, we can achieve different motions and shapes upon tendon actuation. When multiple modules are connected in series, a greater variety of final shapes can be generated and a more extensive workspace can be achieved. We establish a kinematic model for predicting motions using geometric relationships and the minimum potential energy method. The developed model is validated through experimental results using a single module. Finally, we develop a reconfigurable locomotion robot comprising four legs and a central body, where each leg consists of four modules connected in series. We demonstrate walking, crawling, and inching locomotion modes on the same robot without altering its structural design or actuation mechanism.

Chapter 3: Bistable Module Based Reconfigurable Robots

We present methods to tune the energy landscape of a bistable beam-based module to reconfigure its static shapes and dynamic responses on the fly. The two tuning strategies we investigate—adjusting the beam's initial bending angle and varying the offset of the spring's extension—successfully modify the resting angles and energy barriers of the bistable modules. A forward model is developed and validated to predict the module's behavior given the tuning parameters. We also solve the inverse problem to obtain desired resting angles or energy barriers by calculating the corresponding tuning parameters. To demonstrate practical applications, we deploy

the tunable module in three scenarios: as a kicker that imparts different energies to an object, as a reconfigurable arm that adjusts its resting shape, and as a crawling robot that locomotes in both directions using the same actuation mechanism. Our method establishes a framework for designing reconfigurable robotic systems by leveraging the unique capabilities of multistable structures for enhanced performance and versatility.

Chapter 4: Co-Optimization of Design and Control

For modular reconfigurable mechanisms, different reconfiguration parameters lead to distinct robot configurations, which in turn influence the kinematics and dynamics of the robot. When optimizing a control algorithm for specific tasks of the robot, the reconfiguration parameters should be incorporated into the optimization process, which remains a significant challenge. We propose a co-optimization framework that simultaneously optimizes both reconfiguration parameters and control policies for specific tasks using reinforcement learning techniques. We implement this framework using two modeling approaches: the kinematic model, based on forward kinematics and minimum potential energy, and the physics-based model, using the MuJoCo physics engine. We apply the co-optimization process to reconfigurable origami manipulators composed of multiple serially-connected origami modules, simultaneously optimizing the stiffness values of the joints and control policies for reaching tasks. Through extensive training, we achieve near-perfect success rates for reaching tasks, including scenarios with payloads mounted on the top plate. The learned stiffness values and control policies are validated through deployment on a physical two-module prototype, demonstrating effective sim-to-real transfer despite positioning errors attributed to simplified joint representations and unmodeled friction forces. This work establishes a comprehensive data-driven framework for co-optimizing mechanical design and control of modular reconfigurable robots, offering a systematic alternative to manual tuning and enabling extension to more complex tasks such as multi-modal locomotion.

Chapter 2

Origami Module Based Reconfigurable Robots

Existing reconfigurable robots are generally cumbersome or challenging to control and actuate. The adaptability of these robots are also limited after fabrication. Also, the newly developed soft reconfigurable robots with deformable bodies feature continuous and large deformation, making it hard to accurately predict or control the behavior of these robots.

We propose a novel design of reconfigurable robots by stacking multiple reconfigurable origami modules in series and using a few tendons for actuation. The shape and motion of the reconfigurable origami modules can be adjusted on the fly by tuning the stiffnesses of the joints in the modules. The resulting modular reconfigurable robots offer advantages such as ease of fabrication, extensive workspace, predictable behavior, and simplified control.

First, we present a novel reconfigurable module inspired by the Kresling origami pattern (top right of Fig. 2.1). The origami module is equipped with Shape-morphing and Variable-stiffness Joints (SVJs, shown in bottom right of Fig. 2.1) that can independently transition between soft and rigid states, enabling the module to adapt its behavior during actuation. To understand the reconfiguration capability, we numerically analyze the programmable shapes and motions of a single origami module. We develop a reconfigurable robot (left of Fig. 2.1) with four legs, each made from four serially connected modules. The robot can walk, crawl, and inch using the same mechanical structure, without altering its mechanical structure.

2.1 Working Principle

The origami-inspired module with SVJs can be controlled to generate programmable shapes or motions on the fly. In this section, we explain the working principle through an SVJ, a single module, and multiple modules. Then we discuss the implementation details for a single module.

An SVJ is presented in our recent work [25], but we briefly describe its working principle here. It is made from a thermoplastic material (e.g., PLA) embedded inside an elastic tube (Fig.

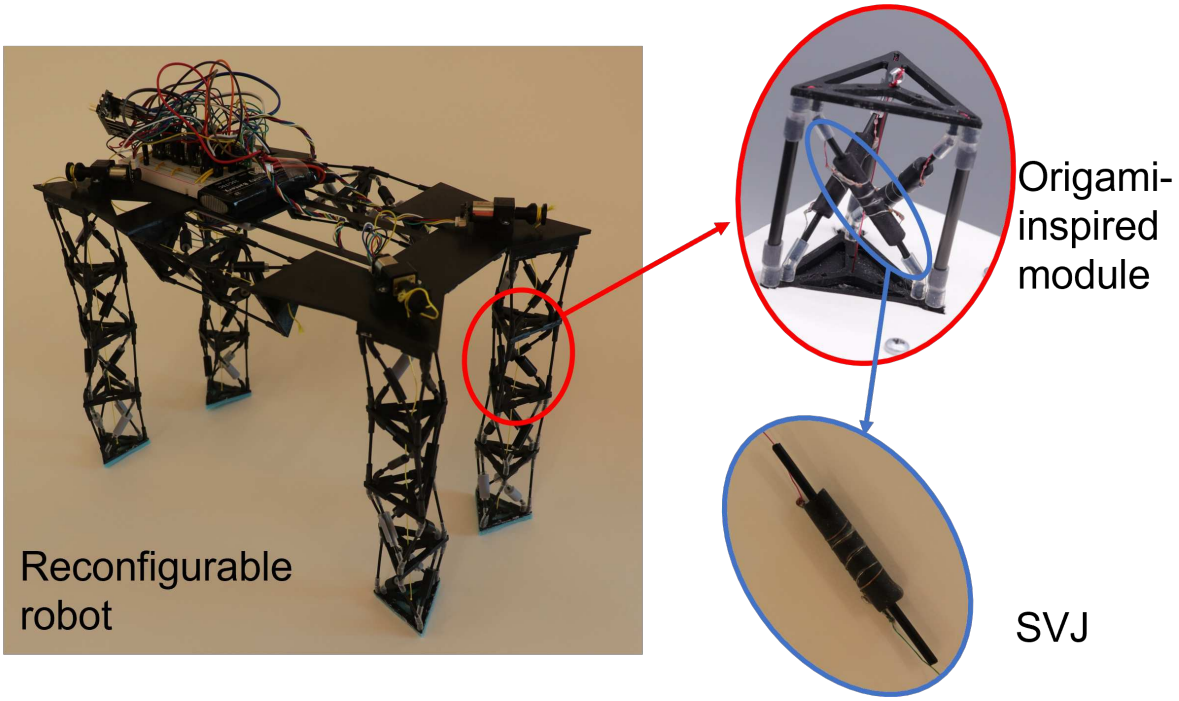


Figure 2.1: A Reconfigurable Robot Using Origami-Inspired Modules with Variable Stiffness Joints

2.2a). A resistance wire wrapped outside the tube can heat up the material through Joule heating (not shown in Fig. 2.2a). With such a design, an SVJ has two characteristics: variable stiffness and shape morphing. For variable stiffness, its bending stiffness can be controlled based on its temperature: once the SVJ's temperature exceeds PLA's glass transition temperature, the SVJ becomes soft, making the SVJ bendable. For shape morphing, it can be heated, bent to a new shape, and maintain that shape without additional energy input after cooling down (top to bottom in Fig. 2.2a). Because of the elastic tube, a bent SVJ can also return to the original shape after heating up and cooling down again (bottom to top in Fig. 2.2a).

Our module is inspired from the Kresling pattern, which has a top and a bottom surface that are connected by several pairs of mountain and valley creases. We implement the surface with rigid plates (top and bottom, shown in blue in Fig. 2.2b). We replace the mountain and valley creases with three vertical and diagonal links, respectively. Note that a Kresling pattern generally have five or six pairs of mountain and valley creases, but in our module, we used three to simplify the structure. Each vertical or diagonal link is connected to both the top and bottom plates through

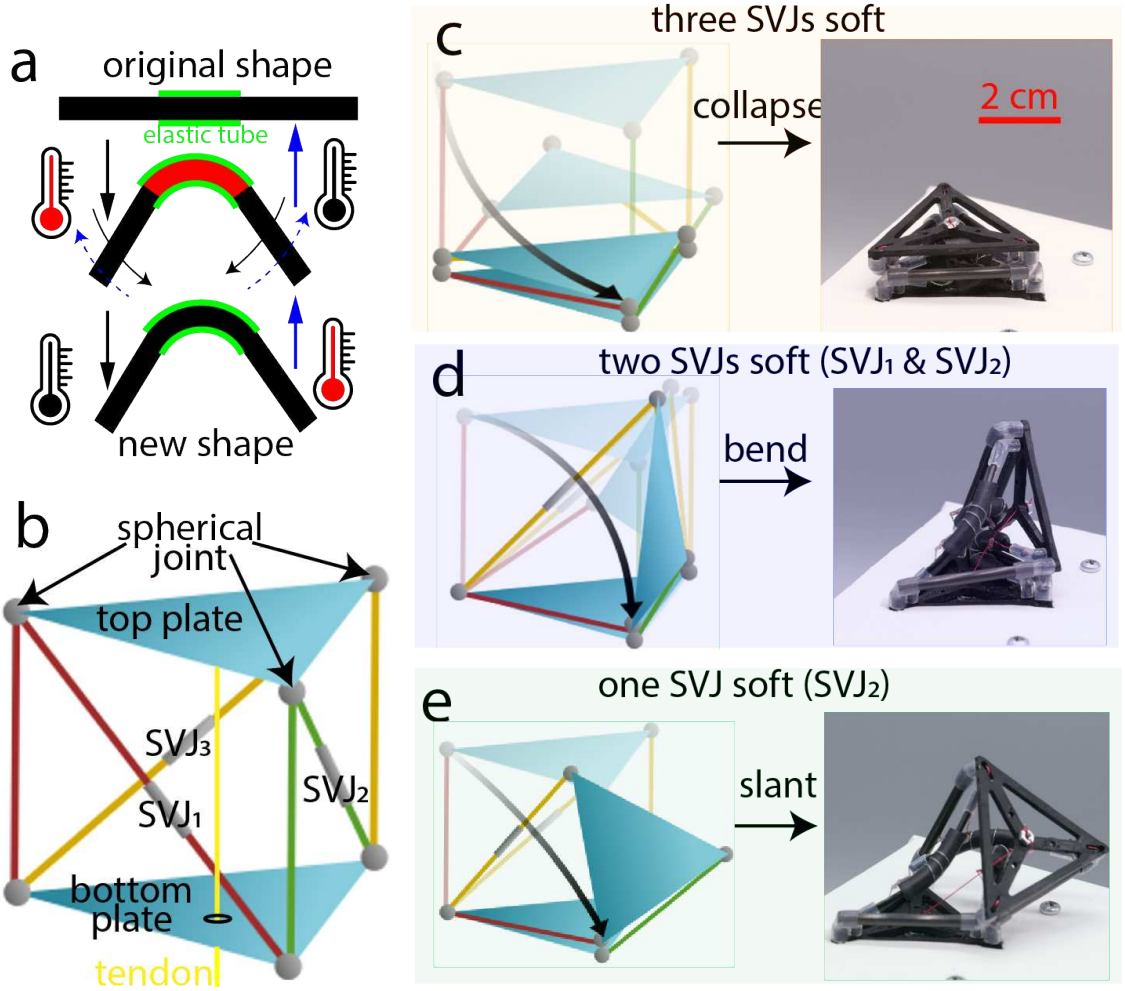


Figure 2.2: Working principle for the origami-inspire module with SVJs. (a) Working principle for an SVJ; (b) Architecture of a module; (c), (d), and (e) illustrate three different motions.

a spherical joint. Each diagonal link has an SVJ in the middle (labeled as SVJ_1, SVJ_2, and SVJ_3 in Fig. 2.2b). Assuming the bottom plate is fixed, we can generate motions by pulling downward a tendon attached to the top plate's center and routed through a hole at the bottom plate's center, resulting in the top plate's motion with respect to the bottom plate. Depending on which SVJs are soft, we can achieve three typical motions: collapse, bend, and slant.

- **Collapse (Fig. 2.2c):** If all the three SVJs are soft, all the three diagonal links can bend. In this case, the top plate will rotate and move downward but keep parallel with the bottom plate. Eventually, the top plate will contact the bottom plate, thus the module is collapsed.

- **Bend (Fig. 2.2d):** If two SVJs (e.g., SVJ_1 and SVJ_2) are soft, only two diagonal links can bend. The top plate will rotate to eventually become perpendicular to the bottom plate.

- **Slant (Fig. 2.2e):** If only one SVJ (e.g., SVJ_2) is soft, then the top plate would be slanted and only one vertex of the top plate would contact a corresponding vertex of the bottom plate.

Based on these different motions, the final shapes for a single module can be different (shown using a prototype in Fig. 2.2). To generate the different shapes, we first heat up desired SVJs to make them soft, control the tendon to pull the top plate to follow a desired motion (collapse, bend, or slant), and finally cool down SVJs so that the module can maintain that shape without any further actuation or energy input.

More diverse shapes and motions can be accomplished if we connect multiple modules in series and selectively and independently control the stiffness of each SVJ in each module. Here, we use an example of two modules connected together to explain the principles. A tendon attached to the topmost plate is routed along the holes at the centers of middle plates to the fixed bottom plate. In this case, if all SVJs in the top module are soft, whereas all SVJs in the bottom one are rigid, pulling the tendon will collapse the top module. On the other hand, if all SVJs in the bottom module are soft but all SVJs in the top one are soft, the bottom module will collapse when the tendon is pulled. More programmable motions can be generated if we consider all the three

different motions (collapse, bend, and slant) for each module. Details on programmable shapes and motions will be discussed in section 2.3 and 2.2, respectively.

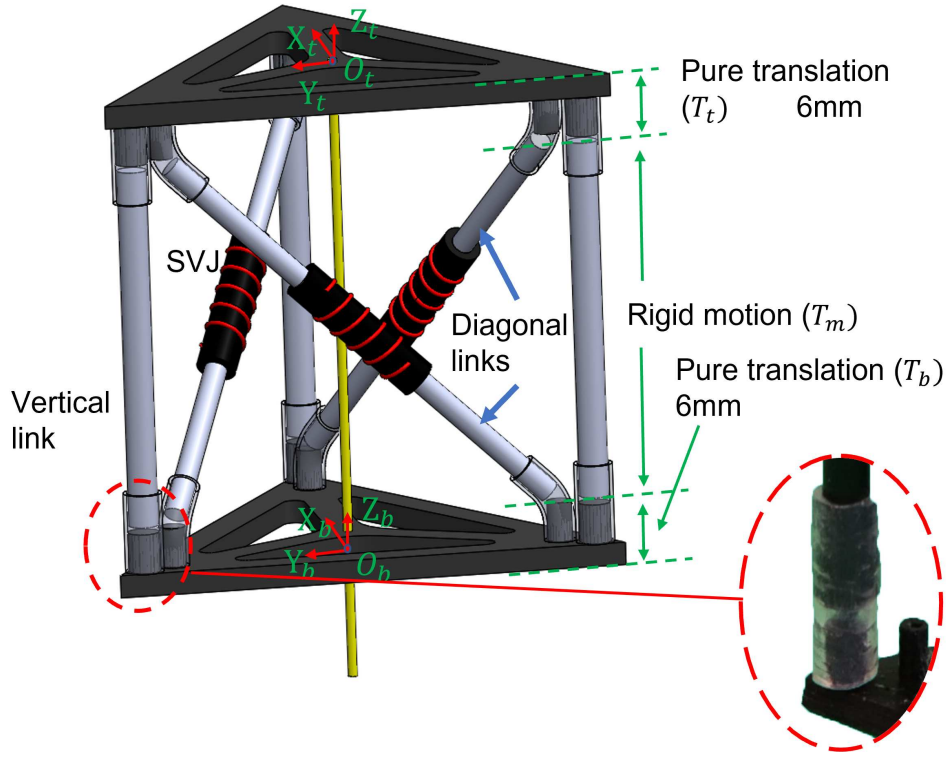


Figure 2.3: Solid model of the module with an enlarge portion of the tube-based joint shown on the lower right.

We briefly describe the main aspects for implementing SVJs (details in [25]). The diagonal link is made from thermoplastic PLA filament (diameter 1.75 mm, Overture). An elastic silicone tube (BKVP51135-71.5; Vanguard) is placed in the middle. A resistance wire with a resistance of 3.1 Ω (shown in red in Fig. 3.2) is wrapped around the silicone tube. Using the resistance wire, it takes 20 s to heat up the SVJ to soften it with a voltage of 3 V. This heating process would consume an energy of 58 J. The heating up and cooling down process can be carried out continuously and repeatedly, thus enabling the reconfiguration of the module on the fly.

A single module can be fabricated as follows (Fig. 3.2). The plates are 3D printed with PLA with two protrusions at each vertex. Each diagonal link is connected to the protrusions of both the top and bottom plates via silicone rubber tubes (LDVP51135-11; Vanguard) which serve as

spherical joints (see inset in Fig. 3.2). The vertical links are hollow carbon fiber tubes (PCT125024; Goodwinds), allowing for the routing of electrical wires inside the links. Distances between the bottom surface of the bottom plate and the bottom spherical joints are 6 mm (Fig. 3.2). The length of vertical links and diagonal links are 38.5 mm and 48.5 mm, respectively. The length of each edge of the equilateral triangular plate is 45 mm. These dimensions are chosen to ensure the module can bend and collapse properly. More specifically, for the collapsing motion, we want the top plate to rotate 120 degrees to fully align with the bottom plate. For the bending motion, we want the eventual orientation of the top plate to be perpendicular to the bottom plate. For the slanting motion, we expect the vertex on the top plate to contact with the corresponding vertex on the bottom plate. Note that they can be scaled up or down to increase or decrease the size of the module. A tendon (shown in yellow in Fig. 3.2) is fixed to the top plate's center. It goes through the hole at the bottom plate's center to connect to a capstan driven by a DC motor (placed beneath the bottom plate, not shown in Fig. 3.2). When pulled by the capstan, the tendon will exert a downward force perpendicular to the top plate that will slant, bend or collapse the module depending on which SVJs are soft. The silicone tubes on the module are elastic and can store elastic energy during the motion. If the tendon is released and SVJs are kept soft, the deformed module will return to the original shape by the tubes' elastic energy.

The diagonal links in both Fig. 3.2 and 2.2 are directed from top left to bottom right. We call this type of module the counterclockwise (CCW) module since they would rotate CCW when collapsed. If the diagonal links are directed from top right to bottom left, then the modules would rotate in the clockwise (CW) direction when collapsed. We call this type of module the CW module. If a CW and a CCW module are connected in series, then the topmost plate will collapse linearly without rotation, since the rotations of these two opposite modules cancel out.

2.2 Programmable Motions

Our origami-inspired module can generate programmable motions: the top plate of a module can continuously move when SVJs maintain the soft state. In this section, we analyze and numer-

ically calculate the motion of the top plate of the module. We also conduct experiments using a single module to validate the theoretical results. Note that the motion without using SVJs (i.e., all the diagonal and vertical links are rigid) has been investigated before [67, 68]. In this case, the module will serve as a truss structure that exhibits different behaviors with different dimensions, such as monostability, bistablity, or tunable stiffness. Therefore, we will not discuss the motion without SVJs in this work.

2.2.1 Modeling of the Motion

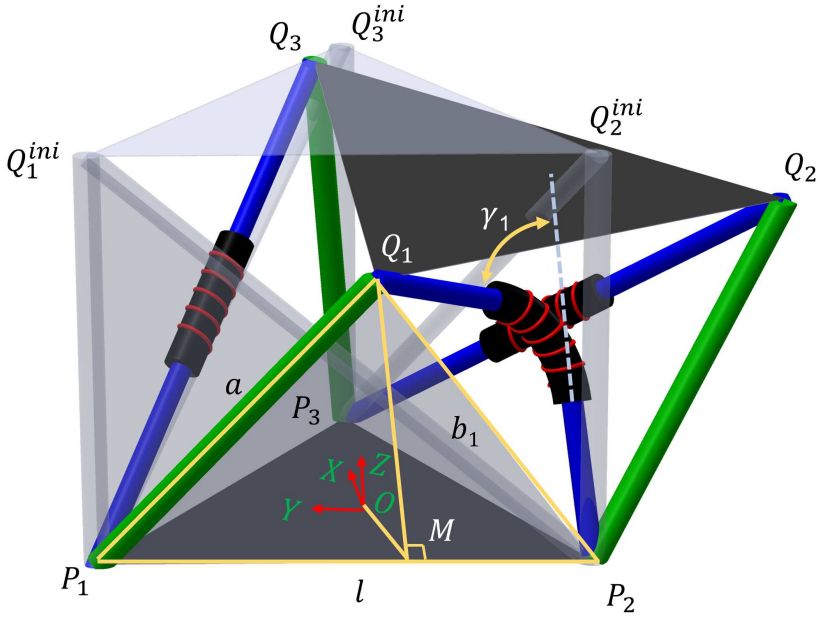


Figure 2.4: A module moves from an initial state (light gray) to a another state.

To predict the motion, we need to obtain the position and orientation of the top plate (T_m) given a certain displacement Δ of the tendon, assuming the bottom plate is fixed. It is difficult to directly solve T_m given Δ, especially when more than one SVJ is soft since the module is underactuated with only one actuation. Therefore, we use a general procedure as follows. First, we express T_m using the unknown bending angles γ_i, $i = 1, 2, 3$, for each of the three SVJs (γ_1 is illustrated in Fig. 2.4).

$$T_m = f(\gamma_1, \gamma_2, \gamma_3) \tag{2.1}$$

Then, we can express the tendon's displacement Δ using the top plate's position from T_m, i.e.,

$$\Delta = g(T_m) \tag{2.2}$$

Then, we can numerically solve the unknown bending angles γ_i from Eqs. (2.1) and (2.2), given a specific value of Δ. Finally, we can use Eq. (2.1) to calculate T_m based on γ_i. We detail the general procedure in the following.

We first solve the relationship between T_m and γ_i, i.e., Eq. (2.1). For the tubes connecting the links and the plates, we can simplify them as spherical joints (e.g., points P_i and Q_i ($i = 1, 2, 3$) in Fig. 2.4). The diagonal links (e.g., P_2Q_1) and the vertical links (e.g., P_1Q_1) are connected to the plates at different but very close points (distance 3.8 mm). For simplicity, we assume they are connected to the plates at the same point. Assume the SVJ on link P_2Q_1 is soft. When we pull the tendon fixed to the top plate's center, the original straight link $P_2Q_1^{ini}$, with an initial length of b_{ini}, will be bent by an angle of γ_1. The length between Q_1 and P_2, denoted as b_1, is a function of γ_1:

$$b_1 = \sqrt{\frac{b_{ini}^2}{2} - \frac{b_{ini}^2}{2}\cos(\pi - \gamma_1)} \tag{2.3}$$

During the motion, the surface $\Delta P_1Q_1P_2$ rotates about the fixed axis P_1P_2 for an angle denoted as θ_1 (the angle between $\Delta P_1Q_1^{ini}P_2$ and $\Delta P_1Q_1P_2$). The position and orientation of the top plate T_m can be related to the unknown values of γ_1 and rotating angles θ_1 through the following procedure.

A coordinate frame $OXYZ$ is established at the bottom plate with the origin O at the centroid of the bottom plate, OX axis along $\overrightarrow{OP_3}$, OZ axis perpendicular to the plane. Let $l = |\overrightarrow{P_1P_2}|$ be the length of the side of the equilateral triangular plate, $a = |\overrightarrow{P_1Q_1}|$ be the length of the vertical links, u_1 be a unit vector pointing from P_1 to P_2. We obtain the coordinate of vertex Q_1 by drawing a line Q_1M perpendicular to the side P_1P_2. In this case, we have

$$\overrightarrow{OQ_1} = \overrightarrow{OM} + \overrightarrow{MQ_1} \tag{2.4}$$

$\overrightarrow{OM}$ can be obtained through

$$\overrightarrow{OM} = \overrightarrow{OP_1} + \overrightarrow{P_1M} = \overrightarrow{OP_1} + \overrightarrow{P_1P_2}\left(\frac{a^2 + l^2 - b_1^2}{2l^2}\right) \tag{2.5}$$

where $\overrightarrow{OP_1}$ and $\overrightarrow{P_1P_2}$ are constant vectors. $\overrightarrow{MQ_1}$ can be obtained by using its direction and length

$$\overrightarrow{MQ_1} = R_{(u_1,\theta_1)}[0,0,1]^T\sqrt{a^2 - \left(\frac{a^2 + l^2 - b_1^2}{2l}\right)^2} \tag{2.6}$$

where $R_{(u_1,\theta_1)} = I + \sin\theta_1 S(u_1) + (1 - \cos\theta_1)S^2(u_1)$ is the rotation matrix obtained by applying the Rodrigues' rotation formula with S being the skew symmetric matrix operator. The physical meaning of $R_{(u_1,\theta_1)}[0,0,1]^T$ is to rotate the initial direction of Q_1M ($[0,0,1]^T$) about u_1 by an angle of θ_1.

The coordinates of the other two vertices (Q_2 and Q_3) of the top plate can be calculated similarly. After obtaining the coordinates of Q_1, Q_2, and Q_3, we can apply the following three geometrical constraints for the isosceles triangle of the top plate

$$\overrightarrow{Q_1Q_2} = \overrightarrow{Q_2Q_3} = \overrightarrow{Q_3Q_1} = l \tag{2.7}$$

to eliminate the intermediate variable θ_i to directly relate Q_i to the three unknown bending angles γ_i. After eliminating θ_i, we can get the position and orientation of the top plate (T_m) using the following equation.

$$Q_i = T_m Q_i^{ini}, \quad i = 1,2,3 \tag{2.8}$$

where Q_i^{ini} is the initial coordinates for Q_i. Eq. (2.8) can be rearranged to get an implicit form of equation (2.1), finishing our first step of the process to express T_m using the unknown bending angles γ_i.

Let d_x, d_y, and d_z be the transnational component in T_m, we can get the displacement of the tendon Δ.

$$\Delta = \sqrt{(d_x^{ini})^2 + (d_y^{ini})^2 + (d_z^{ini})^2} - \sqrt{d_x^2 + d_y^2 + d_z^2} \tag{2.9}$$

where d_x^{ini}, d_y^{ini} and d_z^{ini} are the initial values of d_x, d_y and d_z, respectively. Equation (2.9) represents the detailed form of Eq. (2.2). With Eqs. (2.3) to (2.9), we can numerically solve the unknown bending angles γ_i given a tendon displacement Δ.

If only one SVJ is soft, then there is a unique solution for this unknown bending angle given a specific value of the tendon displacement. If more than one SVJs are soft, however, the module becomes underactuated with more unknowns (e.g., two or three bending angles) than the actuation. In this case, we can apply the minimum potential energy theory [69] to solve for the unknown bending angles given a tendon displacement. The tubes of each SVJ can be considered as torsional springs when a SVJ is soft and bent during the motion. In this case, we can consider the diagonal link is made from two short links connected by a torsional spring (Fig. 2.4). The elastic energy E_i stored in the torsional spring is $E_i = \frac{1}{2}k_i\gamma_i^2$ where k_i is the spring's effective stiffness and can be found through a bending test, γ_i is the bending angle of the SVJ. The minimum potential energy method states that among the infinite many combinations of γ_i that satisfy the given tendon displacement, only the one combination that results in minimum potential energy of the module should occur in reality. Mathematically, we can formulate the problem as an optimization problem.

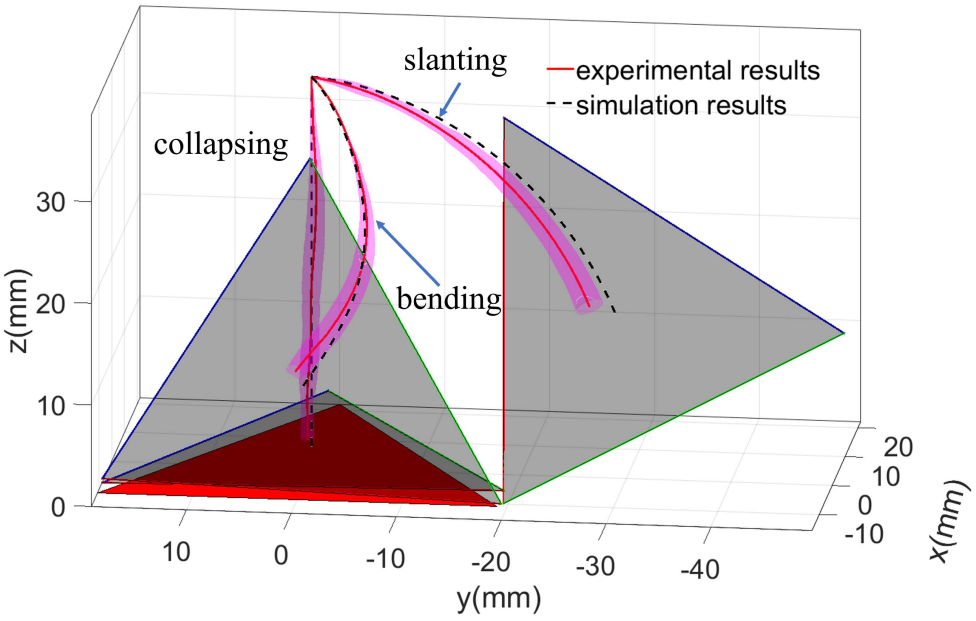
$$\begin{aligned} \underset{\gamma_i}{\text{argmin}} \quad & E = \frac{1}{2}\sum_{i=1}^{i=3} k_i\gamma_i^2 + \frac{1}{2}\sum_{j=1}^{j=12} k_j\sigma_j^2 \\ \text{s.t.} \quad & \Delta = \text{const.} \end{aligned} \tag{2.10}$$

where the first item represents the potential energy stored in the three SVJs, the second item represents the potential energy stored in the silicone tubes connecting the links and the plates (twelve bending angles σ_j), which can be obtained from the T_m of the module.

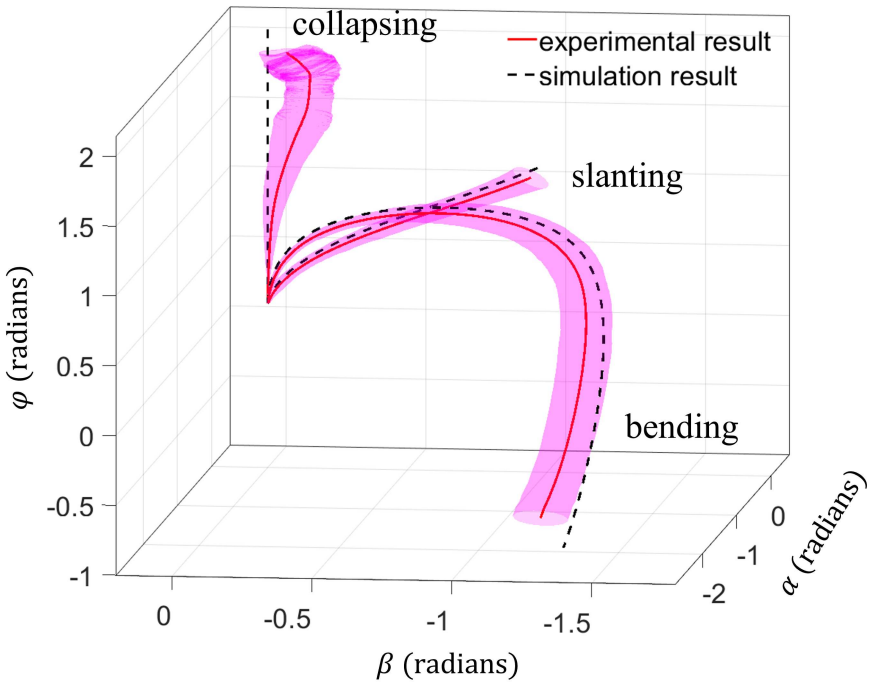
2.2.2 Model Validation

We numerically implement theoretical derivations using MATLAB and compare the results with experiments. In the simulation, we increase the tendon displacement with a step size of 0.1 mm. For each value of the tendon displacement, we can solve for the values of γ_i according to equations (2.3) to (2.9). Then we can get the posture (i.e., T_m) of the top plate from γ_i. For experiments, we use a motion tracking system (V120:Trio; OptiTrack) to track the top plate's motion by attaching three markers to the vertices of the top plate. We also use a test stand (ESM303; Mark-10) to pull the tendon at a steady speed (13 mm/min). For each type of motion, four experiments were conducted.

The experimental results and theoretical prediction for the three different motions (collapse, bend, and slant) are compared in Fig. 2.5 for the position and orientation, where the dashed lines represent the theoretical prediction. The solid lines and shaded areas represent the mean and standard deviation for four repeated experiments. Fig. 2.5a shows the position trajectory of the top plate's centroid. We can see that the solid lines representing the average experimental results and the dashed lines representing the theoretical results match well with a mean error of 2.4 mm. Also, some triangles denoting the final positions of the top plate during the three motions are shown in this figure to better visualize the results. Fig. 2.5b compares the Euler angles for both simulation and experiments. The dashed lines and solid lines here are also close to each other with a mean error of 8.64 degrees. It can be seen that in all these three cases, there exist some errors between the experimental results and theoretical predictions. This is possibly caused by the simplification on the geometry of the module. When multiple SVJs are soft, their stiffnesses may be slightly different (due to fabrication error), which would also influence the motion of the modules. For this underactuated module, a certain error at the beginning of the motion caused by some external perturbation or fabrication error may be enlarged during the motion, which explains the increase of the error during the motion. Additionally, at the ending part of the motion, the contact condition between the links and plates becomes complicated, which may also influence the trajectory of the motion.



(a) Comparison of positions



(b) Comparison of orientations using Euler angles

Figure 2.5: Comparison of experimental and simulation results for motion prediction

The same modeling and numerical analysis can be applied to modules with different dimensions or multiple modules connected in series (as shown in the following section), making the motions of such modules predictable.

2.3 Programmable Shapes

As discussed in the working principle, our origami-inspired modules also have the shape morphing capability: by selectively softening SVJs, we can move the top plate towards another configuration, and then cool down the SVJs to maintain that configuration. This capability can be used for reconfiguring the shapes for a leg made from several modules connected together. In this section, we analyze the number of shapes that can be generated.

To facilitate the discussion, we establish coordinate frames as follows (Fig. 2.3). A fixed frame $O_bX_bY_bZ_b$ is established at the bottom plate's bottom surface with the origin O_b located at the plate's center, axis Z_b pointing upward, X_b from O_b to a vertex of the bottom plate. Similarly, a body frame $O_tX_tY_tZ_t$ is established at the top plate's top surface with the origin O_t at the center. Initially, these two frames have the same orientations. For a single module, we can use the position and orientation (i.e., transformation matrix $T \in SE(3)$) of the body frame $O_tX_tY_tZ_t$ with respect to the fixed frame $O_bX_bY_bZ_b$ to indicate the final shape of the module.

As shown in Fig. 2.3, $T \in SE(3)$ can be obtained by sequentially multiplying three matrices T_b, T_m, and T_t.

$$T = T_bT_mT_t \tag{2.11}$$

where T_b and T_t are pure translations along the z axis, representing the thickness of the plates and heights of the protrusions (Fig. 2.3).

$$T_b = T_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

T_m represents the rigid motion of the top plate's bottom surface relative to the bottom plate's top surface. Using X-Y-Z Euler angle, T_m can be represented as

$$T_m = \begin{bmatrix} c_\beta c_\phi & -c_\beta s_\phi & s_\beta & d_x \\ s_\alpha s_\beta c_\phi + c_\alpha s_\phi & -s_\alpha s_\beta s_\phi + c_\alpha c_\phi & -s_\alpha c_\beta & d_y \\ -c_\alpha s_\beta c_\phi + s_\alpha s_\phi & c_\alpha s_\beta s_\phi + s_\alpha c_\phi & c_\alpha c_\beta & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

where α, β, and ϕ are the rotation about the x, y, and z axis, respectively. $\boldsymbol{d} = (d_x, d_y, d_z)$ represents the translation part, which is the vector from the center of the bottom plate's top surface to the center of the top plate's bottom surface. Since T_m is a function of α, β, ϕ, d_x, d_y, d_z, we can denote it as $T_m(\alpha, \beta, \phi, d_x, d_y, d_z)$.

For a single module, we will have eight different final shapes given which SVJs are soft. Here we are interested in the final shapes when the tendon reaches its maximum displacement, but there are infinitely many other shapes before this final shape. All shapes have the same T_b and T_t, and only T_m varies with different shapes. In the following, we use the values of T_m, instead of the values of T, to represent the shapes. When only one SVJ is soft, we can have three different shapes depending on which SVJ is soft, corresponding to the slanting motion. One of the three shapes, $T_{m,1}$, is shown in table 2.1. When two SVJs are soft, we can have another three different shapes corresponding to the bending motion. One of them is shown as $T_{m,2}$ in table 2.1. For all three SVJs are soft, we have one final shape $T_{m,3}$ corresponding to the collapsing motion. When no SVJ is soft, the module will stay at the initial shape $T_{m,4}$.

When connecting multiple modules in series, we can easily obtain the transformation matrix of the topmost plate relative to the bottom plate by multiplying the corresponding transformation matrix of each module sequentially. Since we have eight different shapes for a single module, for two modules connected together, we can generate $8^2 = 64$ different final shapes. To illustrate all possible shapes, we plot them in Fig. 2.6a using a black dot to show the position of the topmost plate's centroid. Fig. 2.6b shows the initial straight shape of the prototype. In this prototype,

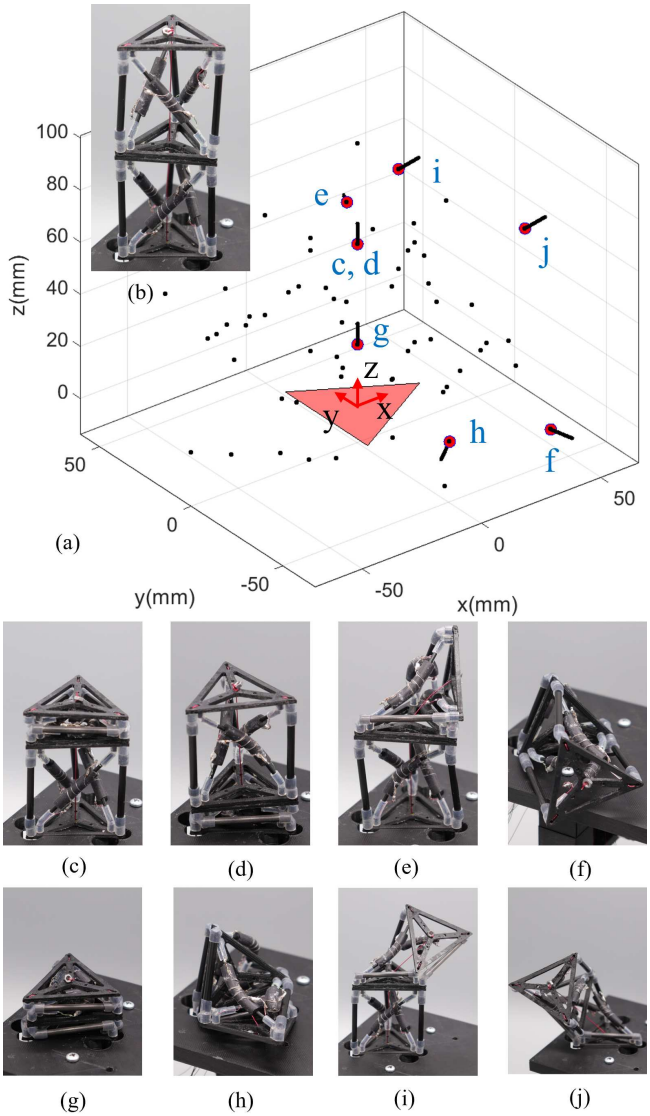


Figure 2.6: Programmable shapes of two modules connected in series.

Table 2.1: Possible final shapes for one module

T_m	α (radians)	β (radians)	ϕ (radians)	d_x (mm)	d_y (mm)	d_z (mm)
$T_{m,1}$	-1.01	0.12	0.83	-26.26	12.16	15.01
$T_{m,2}$	-3.05	-1.37	-0.98	-14.34	-0.82	11.97
$T_{m,3}$	0	0	2.06	0	0	0
$T_{m,4}$	0	0	0	0	0	38.5

the bottom module is a CW module while the top module is a CCW module. We also show eight typical shapes using the developed prototypes in Fig. 2.6c-j: top module collapsed, bottom module collapsed, top module bent, bottom module bent, both modules collapsed, both modules bent, top module slanted, and bottom module slanted, respectively. These eight shapes are indicated by red points in Fig. 2.6a, where the black lines emanating from each point represent the normals to the top plate. The red triangle denotes the fixed bottom module.

The number of final shapes will increase significantly with respect to the number of modules. With n modules connected in serial, we would have 8^n different final shapes.

2.4 Reconfigurable Robot

In this section, we combine the programmable motions and shapes to develop a reconfigurable robot with multiple locomotion modes. The robot has four legs, each made from four modules connected in series. In the following, we first discuss the shapes and motions for a single leg. After that, we discuss the whole robot and demonstrate the locomotion capability.

2.4.1 A Single Leg Consisting of Four Modules

To create a single leg, we connect 4 modules with SVJs in series. We choose 4 modules because four modules are necessary and sufficient to enable the three types of locomotion of the proposed robot. We number the modules as 1, 2, 3 and 4 from the base to the tip (Fig. 2.7a). The whole leg is actuated by a single tendon driven by a DC motor. To realize different locomotion modes, the leg needs to have a specific shape and motion (see the supporting video for the shape and motion of a single leg with SVJs).

For the walking locomotion, the leg is in a straight shape (shown as transparent in Fig. 2.7b), with module 1 permanently collapsed to decrease the leg length and modules 2-4 unchanged. To generate motion, module 2 on the leg is repeatedly slanted and released. The SVJ on module 2 can be kept soft by applying a smaller current (0.1 A) into the resistance wire during the motion. The slanting motion will raise the foot up (shown as nontransparent in Fig. 2.7b) and then, when

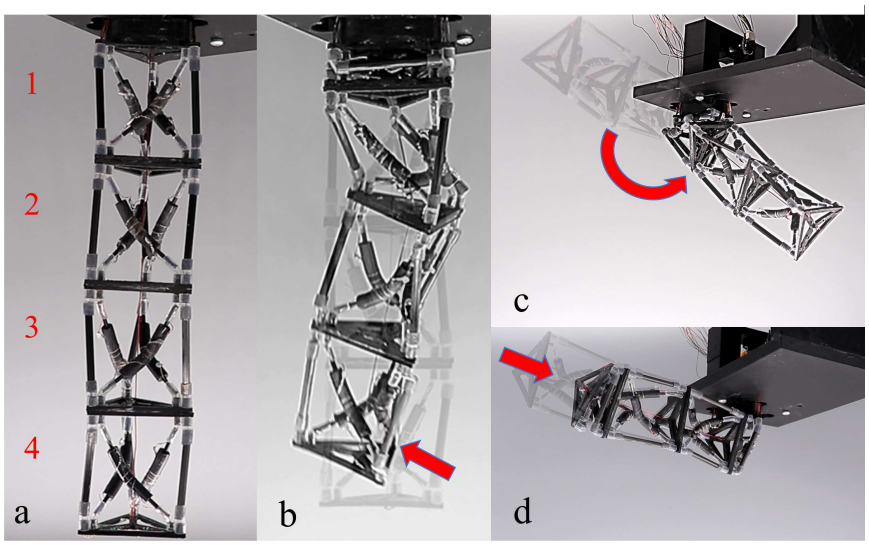


Figure 2.7: Shapes and motions of a single leg. (a) The initial straight shape. (b), (c), and (d) illustrate the motion for walking, crawling, and inching, respectively. The transparent and nontransparent show the leg's location at the beginning and the end of motion, respectively.

released, the elastic force of the soft SVJs will make the leg return to the straight shape and drive the robot forward.

For the crawling locomotion, we need to reconfigure the leg's shape into a configuration that is aligned with the body (shown as transparent in Fig. 2.7c). To do this, we can first soften two SVJs on the 1st module and pull the tendon to bend the module by $90°$, making the whole leg parallel to the ground. After that, we let those two SVJs cool down to make the 1st module keep the bent shape. Now we soften two SVJs on the 2nd module and repeatedly pull (nontransparent in Fig. 2.7c) and release (transparent in Fig. 2.7c) the tendon to realize the forward and backward motions of the leg used in the crawling gait.

For the inching locomotion, the leg's shape (transparent in Fig. 2.7d) is similar to that used in the crawling locomotion. Then we can soften all the three SVJs in the 4th module and then repeatedly collapse (nontransparent in Fig. 2.7d) and release (transparent in Fig. 2.7d) the module for the inching locomotion. Again, the three SVJs can be kept soft by a smaller current throughout the motion. When we release the tendon, the 4th module would try to return to the original straight

shape until the top plate is blocked by a vertical wall (not shown here). The wall blocking the top plate will provide a friction force to anchor the legs for inching motion.

2.4.2 Design of the Reconfigurable Robot

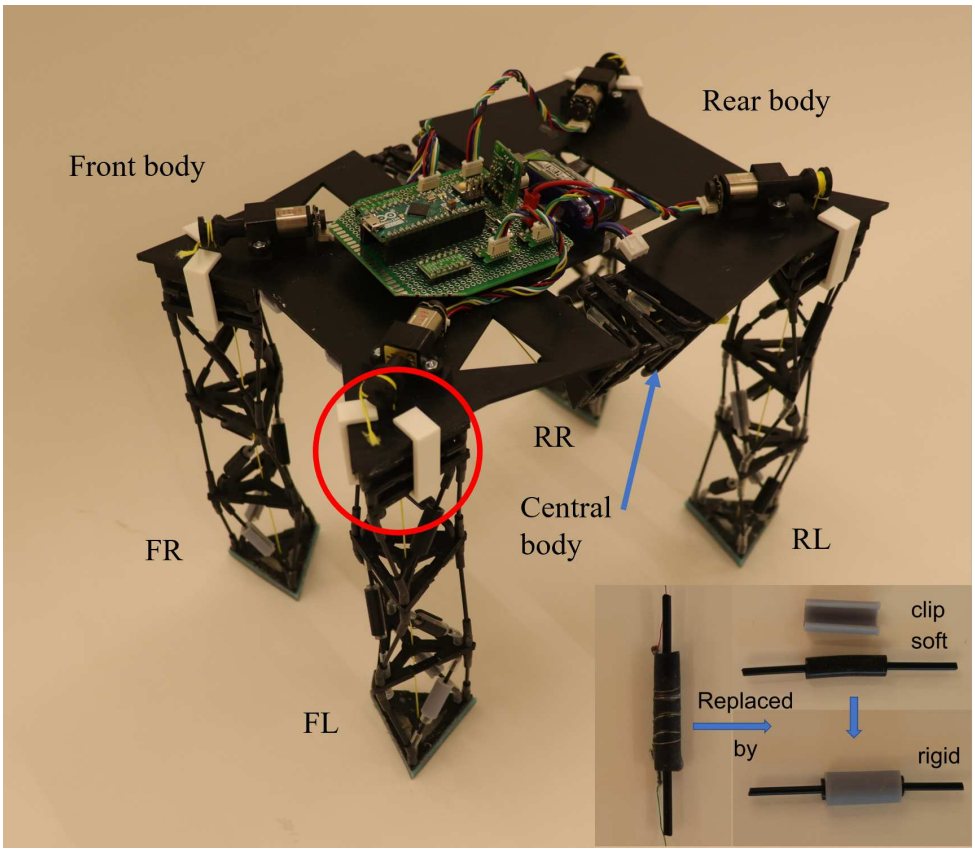


Figure 2.8: The reconfigurable robot and its multiple modes of locomotion.

To build the whole robot, we use a rigid clip (inset of Fig. 2.8) in lieu of the original SVJs for the sake of simplicity. In the original SVJs, a single link is enclosed by a silicone tube with a resistance wire. With the simplified SVJ, we use two short links connected by the soft silicone tube. In this case, the joint is soft since the two short links can rotate freely about each other with the soft silicone tube serving as a bending joint. If we manually apply the rigid clip on the silicone tube, then the joint is rigid in straight shape and cannot be bent. Another type of clip is used to

keep the tube rigid in bent shape. In this way, we can manually switch the joint between the soft state and the rigid state (either in the bent shape or the straight shape), with the help of the clips.

The robot (Fig. 2.8) consists of four legs, a front body, a rear body, a central body, and the electrical parts. The four legs are individually actuated and controlled by their corresponding motor-capstan-tendon systems to perform specified gaits. The central body is made from two pairs of two modules connecting the front and rear body on each side of the robot. Each pair of modules are in opposite directions (i.e., CW and CCW). We choose such a design to facilitate the inching locomotion, which requires a linear contraction and extension of the robot body. When the central body is actuated by a fifth motor-capstan-tendon system placed beneath the front body, the rotations of the two modules in each pair would cancel out and result in a pure contraction of the central body. In this way, we can contract the body for the inching locomotion. Fig. 2.1 shows the central body in its unactuated (extended) state, while Fig. 2.8 shows the robot with the central body contracted. Two carbon fiber plates are used as the sliding guide between the front and rear body.

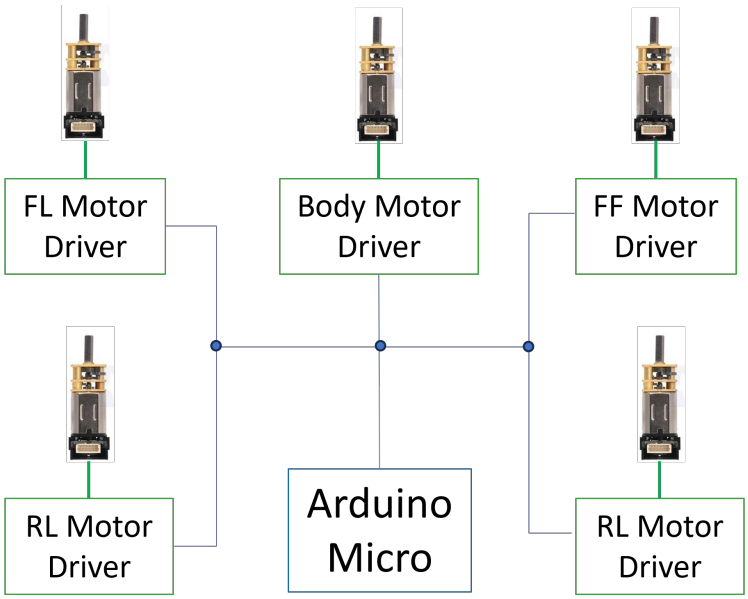


Figure 2.9: The circuit diagram of the locomotion robot.

An Arduino micro board is used as the microcontroller to control the motions of the five motors (3078; Pololu). Figure 2.9 illustrates the circuit diagram of the robot's motor control system, showing the connections between the Arduino micro board and the five motor drivers (front left, front right, rear left, rear right, and central body). Positions of the motors are fed back to the Arduino through a magnetic encoder (4760; Pololu) attached to the rear shaft of each motor. PID control is used to ensure a constant speed of the motor, with the help of the position feedback. The motor is stopped when the desired tendon displacement is reached. For each locomotion mode, the motor speed and displacement are experimentally optimized to improve the locomotion performance.

2.4.3 Locomotion Demonstration

Walking: For walking, we first collapse the central body to shorten the length of the robot. Then we collapse the first modules of each leg to lower the robot's center of gravity. These two steps help to make the walking gait steady and robust. On each leg, only the joint facing forward on the second module is soft. The gait design of walking locomotion is described in Fig. 2.10. The four legs are grouped into two pairs (row a in Fig. 2.10 Right). Front right (FR) leg and rear left (RL) leg form a pair and the other two legs (FL and RR) form the other pair. When we actuate the first pair (FR and RL), these two legs would move forward and upward as demonstrated in the second row. Then the tendons of the first pair are released and these two legs would touch down on the ground (rows b and c in Fig. 2.10 Right), enabling a friction force from the ground. After that, we actuate the other pair of legs (FL and RR) to move them forward and upward (row d in Fig. 2.10). Now, the elastic force of the bent links in the legs will push the robot forward, with the help of friction force from the ground. In this way, a cycle of the walking gait is finished, and the robot assumes the initial state again. The two pairs are actuated sequentially and repeatedly to realize a continuous walking locomotion. A sequence of images during the walking locomotion are shown in Fig. 2.10 Left. The walking speed is measured to be 28.1 mm/s for a motor speed of 18.6 mm/s and a cycle of motion to be 1.8 s.

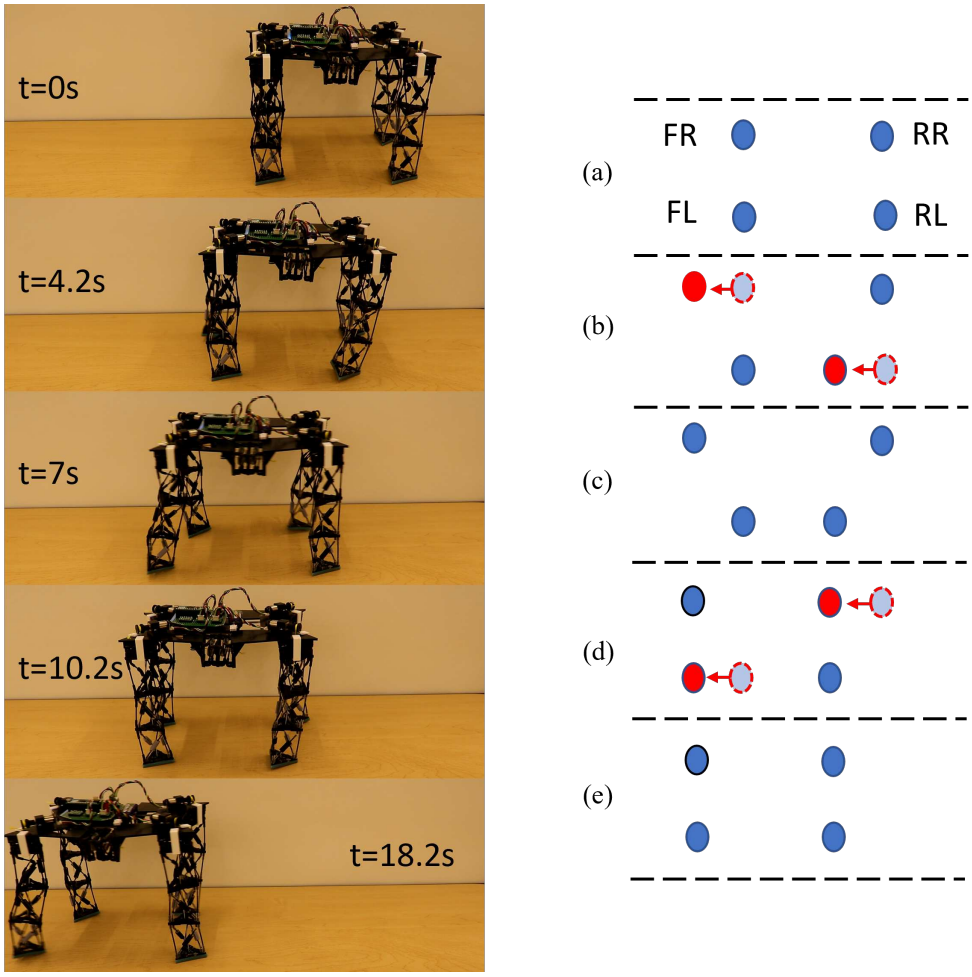


Figure 2.10: Walking locomotion of the reconfigurable robot. (Left) Time-stamped image sequence showing the robot during the walking locomotion. (Right) Schematic representation of the gait pattern, where blue circles indicate stance legs (ground contact) and red circles indicate swing legs (in motion)

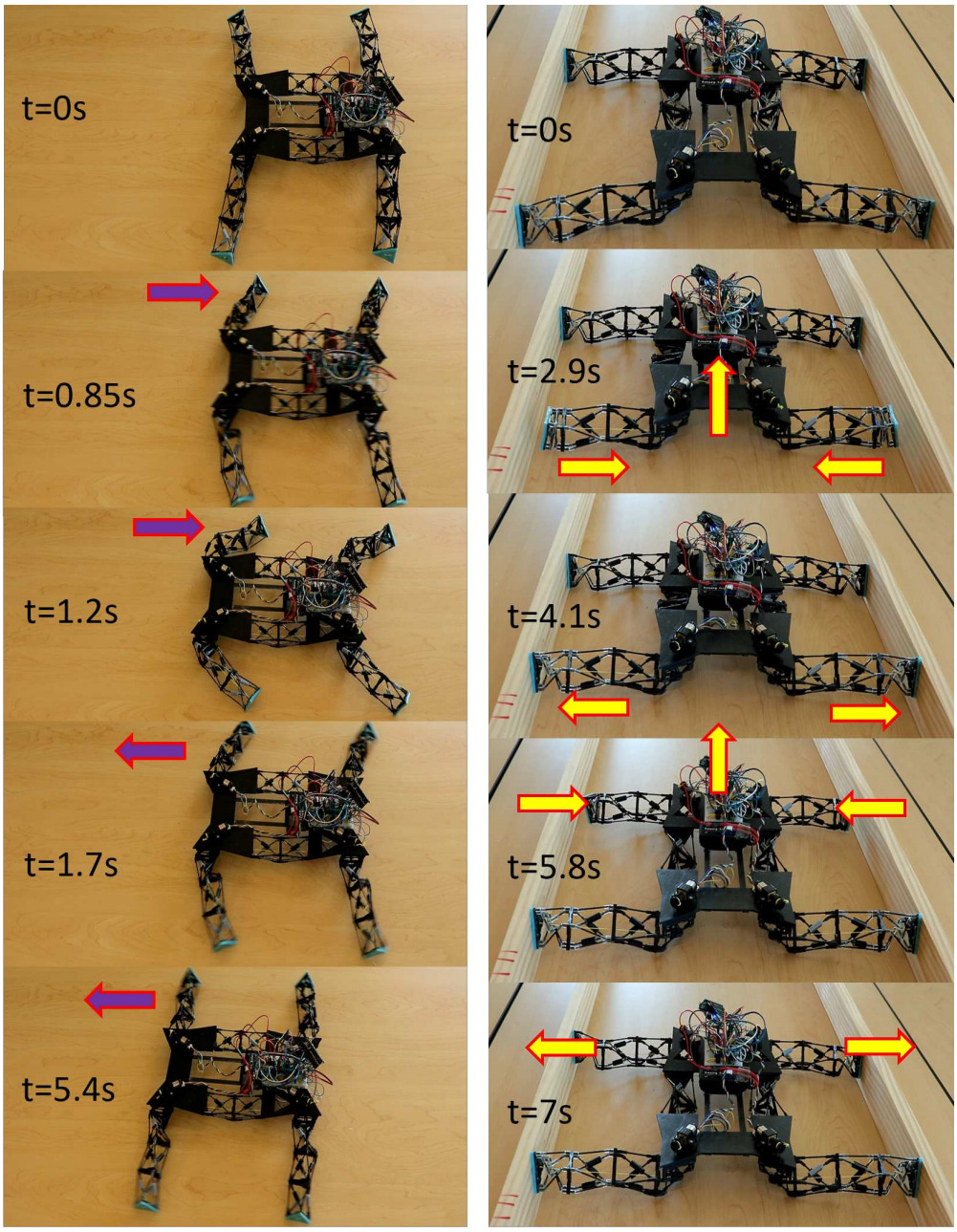


Figure 2.11: The reconfigurable robot in crawling and inching locomotion.

Crawling: For crawling, we first permanently bend the first modules of all the legs using the clips to let the robot lie flat on the ground and be ready for crawling. Then we repeatedly bend ($t = 0.85$ s) and release ($t = 1.7$ s) the second module of each leg to drive the robot forward. In the bending phase of the leg, the leg would try to move downward and backward (as shown in Fig. 2.7c). But this movement is blocked by the ground, and the resulting friction force would drive the robot's body forward. In the releasing phase of the leg, the friction force with the ground is smaller than that of the bending phase. This small force is not sufficient to slide the robot body back. Thus, the directional friction force enables the forward crawling locomotion. During the locomotion, the left and right legs experience symmetric motions. The sequence of the crawling locomotion is shown in Fig. 2.11a. The purple arrows in the figure show the moving direction of the legs. Arrows pointing to the right (left) indicate the bending (releasing) phase of the legs. We measured the crawling speed to be 21.3 mm/s for a motor speed of 42.3 mm/s and a cycle of motion to be 1.8 s. We can turn the robot left by actuating only the left legs or turn the robot right by actuating only the right legs. Potentially, it's possible to adjust the turning radius by fine tuning the displacements and timing between the left and right legs.

Inching: With the same mechanism, we can also achieve an inching locomotion between two parallel walls with a distance of 45 cm. Similar to the crawling motion, we permanently bend the 1st modules of all the legs to make the robot lie flat on the ground and get ready for inching locomotion. Then, we make all three links on the 4th module to be soft in each leg. When actuated, the 4th module should be collapsed. Also, all the joints in the central body are reconfigured to be soft. The sequence of the inching locomotion are shown in Fig. 2.11b, where the yellow arrows indicate the moving directions of the legs and central body. First, we simultaneously actuate the central body and two rear legs. When actuated, the two rear legs would get away from the wall. Since the front legs still contact with the walls and provide the friction force, the central body will drag the rear body forward ($t = 2.9$ s). Now, we release the two rear legs to let them contact with the wall again ($t = 4.1$ s). Then we collapse the front legs (detaching them from the walls) and release the central body. In this way, the elastic force of the central body will drive the front

body forward while the rear legs remain static ($t = 5.8$ s). At last, the front legs will be released to reverse the robot to the original shape. A complete cycle is now finished, and the robot moves forward about 60 mm.

2.5 Chapter Summary

This chapter presents a novel origami-inspired reconfigurable module with SVJs and demonstrate its application in reconfigurable robots. The origami module can be reconfigured on the fly to generate three different motions (collapse, bend, and slant) and shapes depending on which SVJ is soft. We enumerate and demonstrate the possible final shapes (8 for a single module). If we connect multiple modules in series, more final shapes can be generated (8^n).

We also establish a kinematic model for predicting motions using geometric relationships and the minimum potential energy method. The developed model is validated by experimental results using a single module.

Finally, we develop a reconfigurable robot consisting of four legs and a body made from the origami modules. With this reconfigurable robot, we demonstrated walking, crawling, and inching locomotion without altering the mechanical structure of the robot.

Chapter 3

Bistable Module Based Reconfigurable Robots

Multistable structures, characterized by their ability to rapidly switch between multiple stable states, represent an emerging area of study in robotic or mechatronic systems. These structures, after switching to a stable state, can maintain the new stable configuration without any external energy input. A special type of multistable structure is a bistable module with two stable states, which has been observed widely in nature. For instance, the Venus flytrap's leaves are bistable, which can snap shut in 0.1 s to capture prey [26]. Similarly, the rapid beak closure of hummingbirds is another instance of natural bistability, enabling these birds to catch flying insects efficiently [27].

Researchers have employed bistable modules to realize reconfigurable robots with multiple functionalities [8, 28, 29]. Compared to origami module based reconfigurable robots, bistable module based robots have the advantages of fast response [30] and enlarged output force [31, 32]. Furthermore, bistable module based reconfigurable robots can maintain a specific shape without requiring external energy input.

For multistable or bistable modules, an important characteristic is the energy landscape (EL): how the stored strain energy varies with respect to the deformation (e.g., bending angle). A typical EL is plotted in Fig. 3.1B as the black curve. The EL can determine both the stable configurations and the dynamic responses of the module. Despite substantial recent research, existing multistable modules are generally fixed after being fabricated, leading to a predetermined EL [29, 30], constraining their adaptability, reconfigurability, and application scope.

To extend the reconfigurability of bistable modules, researchers have recently investigated how varying specific parameters of the modules will influence their ELs, thereby generating different behaviors (stable configurations or dynamic responses) when actuated [33–35]. For instance, researchers realized adjusting the ELs of the bistable modules by changing the initial bending angles of the flexible beams [36], controlling the voltages applied to the twisted and coiled polymer fibers

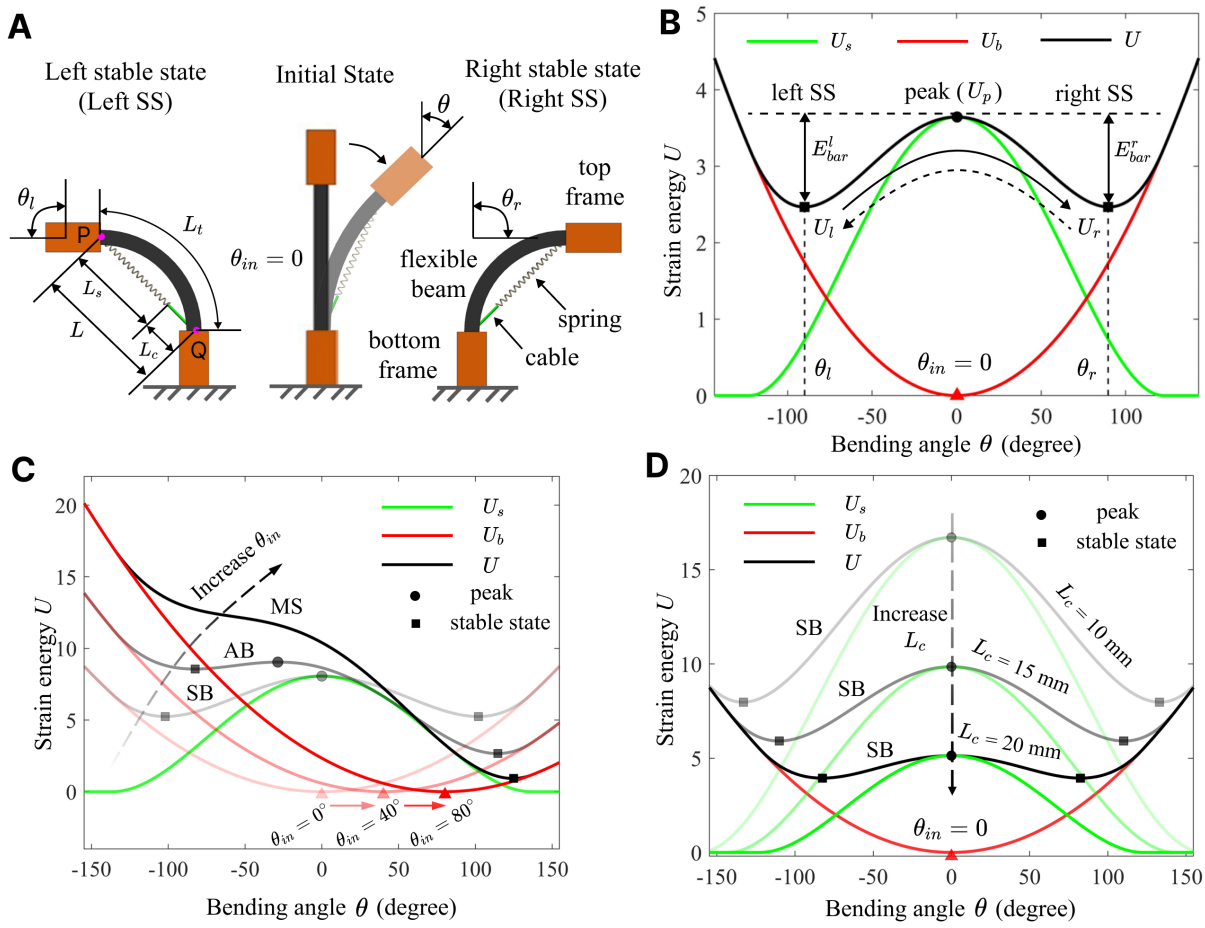


Figure 3.1: Working principle for tuning the energy landscape (EL) of a beam-based module. (A) A symmetric bistable module in its left stable, straight, and right stable states, respectively. (B) The EL of the symmetric bistable module. (C) When θ_{in} increases with $L_c = 17$ mm, the module changes from symmetric to asymmetric bistable and finally to monostable. (D) When L_c increases with $\theta_{in} = 0$, the spring's energy profiles will scale down, changing both the energy barriers and resting angles.

or manually altering the prestretch of the elastic cords [37], or tuning the magnetic field applied to the bistable module [38].

Despite recent progress in adjusting the ELs for bistable modules, two issues remain largely unaddressed. First, most of the existing work, with notable exceptions [37, 38], cannot adjust the ELs on the fly. Current approaches typically require manual alterations, such as the manual modification of the initial lengths of spring elements [32, 39, 40], or the manual adjustment of the initial bending angle of the bistable module by changing the dimension of its bending beam [41, 42]. It would be more advantageous if we could tune the EL on the fly to enhance the reconfigurability

of bistable modules or robots. Second, there are few systematic explorations on how to achieve the desired ELs by choosing proper parameters of the bistable modules. Instead, most existing work predominantly conducts simulations or experiments to observe how different parameters will influence the ELs [43–46].

We directly address these two issues with a beam-based bistable module. First, we introduce two strategies to tune the module's EL without manual alterations, adjusting its behavior on the fly. The online tuning strategies significantly enhance its reconfigurability. We then systematically investigate how to achieve the desired EL by selecting appropriate tuning parameters. We call this the inverse problem, which is solved based on the forward problem: predict the EL given the tuning parameters. By solving the inverse problem, we offer a more structured and guided approach to designing tunable processes for bistable modules.

We demonstrate the wide reconfigurability of this tunable module with three different reconfigurable robots: a kicker, a reconfigurable arm, and a crawling robot.

3.1 Working Principle for Tuning the ELs

In this section, we explain how to actively tune the ELs of modules with elastic instabilities. Without loss of generality, we use a beam-based module with a linear spring to illustrate the working principle. This module primarily comprises a flexible beam and an elastomeric spring (see labels on the right of Fig. 3.1A). The beam is connected to two rigid frames (top and bottom). One end of the spring is anchored to the top frame at point P, while the other end is connected to an adjustable cable, which is, in turn, attached to the bottom frame at point Q. The adjustable cable of length L_c is used to offset the extension of the spring. When the dimension and bending angle of the flexible beam are given, increasing the length of the adjustable cable results in a decrease in the length or extension of the spring. The beam has a rectangular hole at its center, allowing for the passage of both the spring and the adjustable cable as the beam undergoes bending motions. With an initially straight flexible beam, this module exhibits two symmetrical stable states (SS):

the left stable state (left SS) and the right stable state (right SS), each mirroring the other relative to the beam's straight configuration (Fig. 3.1A).

To analyze the stable states, we need to consider the module's strain energy. The total strain energy (U) of the module consists of two components: the strain energy of the flexible beam (U_b) and that of the spring (U_s). Since the flexible beam is long and slender, the strain energy of the bent beam due to shear stress and compressive stress is negligible compared with that due to normal stress. So U_b mainly comes from the bending energy of the flexible beam, while U_s comes from the spring's extension. We can represent the total energy U as a function of the beam's bending angle θ

$$U = U_b + U_s = \frac{1}{2}k_b(\theta - \theta_{in})^2 + \frac{1}{2}k_s\Delta^2(\theta) \tag{3.1}$$

where k_b is the beam's effective bending stiffness, θ_{in} is the beam's initial bending angle ($\theta_{in} = 0$ for an initially straight beam), and k_s is the spring's stiffness. $\Delta(\theta)$ represents the spring's extension as a function of θ, which can be expressed as

$$\Delta(\theta) = \max(0, L(\theta) - L_c - L_{in}) \tag{3.2}$$

where $L(\theta)$, varying with θ, is the distance between the two anchoring points P and Q, L_c is the length of the adjustable cable, L_{in} is the spring's nominal length. As can be seen in Eq. (3.2), L_c can be used to offset the extension of the spring for a given $L(\theta)$. Assuming the beam is uniformly deformed and takes on the shape of a circular arc, $L(\theta)$ is a chord PQ corresponding to this circular arc. The radius of the arc is expressed as $R(\theta) = L_t/\theta$, where L_t is the length of the beam. In this case, L can be obtained as a function of θ by the following equation

$$L(\theta) = 2R\sin(\frac{\theta}{2}) = 2\frac{L_t}{\theta}\sin(\frac{\theta}{2}) \tag{3.3}$$

With Eq. (3.1), we plot U, U_b, and U_s as functions of the bending angle θ, represented by the black, red, and green curves, respectively (Fig. 3.1B). The beam's strain energy, U_b, is a convex

parabola with its vertex located at $(\theta_{in}, 0)$. The spring's strain energy, U_s, is symmetric about the vertical axis $\theta = 0$. The resulting total EL of the module, $U = U_b + U_s$, has two local minima at θ_l and θ_r (shown as the black squares in Fig. 3.1B), indicating the left and right stable states (left SS and right SS) of the module. The module is in equilibrium when the bending angle θ equals θ_l or θ_r, and external energy is required to shift the module away from the stable states (θ_l or θ_r). The peak of this EL (U_p, shown as the black dot in Fig. 3.1B), indicating the module's maximal strain energy and its unstable straight state, is located at $\theta = 0$. The differences between the energy peak (U_p) and energies at the local minima (U_l or U_r) are termed the energy barriers (i.e., left energy barrier $E^l_{bar} = U_p - U_l$ and right energy barrier $E^r_{bar} = U_p - U_r$). We call this configuration symmetric bistable (SB) since the EL is bistable with two minima (θ_l and θ_r) and symmetric ($\theta_r = -\theta_l$).

To transit the module from the right SS to the left SS, we apply a force to bend the module to the left, overcoming the right energy barrier E^r_{bar}. Application of this external force will increase the strain energy stored in the module. Once the module surpasses the energy peak U_p, part of the stored strain energy would be instantly released, resulting in a rapid snap motion to the left SS. A similar mechanism is needed when switching from the left to the right SS, where an appropriate force must overcome the left energy barrier E^l_{bar} to initiate the transition.

By tuning the EL, we can adjust the positions of the two resting angles (θ_l and θ_r). If the magnitudes of θ_l and θ_r are adjusted to be different ($|\theta_r| \neq |\theta_l|$), the module is referred to as asymmetric bistable (AB). If one resting angle disappears, the module will become monostable (MS). Similarly, the value of each energy barrier (E^l_{bar} and E^r_{bar}) can also be adjusted by tuning the EL. An increase in E^r_{bar}, for instance, leads to a more rapid or forceful transition from the left to the right SS (solid curved arrow in Fig. 3.1B) since a greater amount of stored energy is released upon surpassing the energy peak U_p. Meanwhile, this increase in E^r_{bar} makes it more difficult to transit from the right to the left SS since more energy input is required to overcome a larger E^r_{bar}.

Since we have four independent tunable elements: k_b, θ_{in}, k_s, and L_c in Eqs. (3.1) and (3.2), in theory, we can simultaneously achieve all the four desired characteristics (θ_l, θ_r, E^l_{bar} and E^r_{bar}) by choosing proper values for k_b, θ_{in}, k_s, and L_c. In practice, however, it is quite challenging to

concurrently tune these four tunable elements in a tunable module. Additionally, it is generally not necessary to achieve all four characteristics at the same time (e.g., in some cases, only the resting angles are required, and the energy barriers are not critical). Therefore, in this study, we only investigate strategies to adjust two elements on the fly, θ_{in} and L_c, to tune the EL to achieve either two desired resting angles (indicating the stable states of the module) or two desired energy barriers (indicating the dynamic responses of the module). We choose to adjust θ_{in} and L_c because they are relatively easy to implement and prove effective in achieving the desired characteristics (implementation details are discussed in section 3.2).

We first illustrate how adjusting the beam's initial bending angle (θ_{in}) can influence the module's EL. Since $U_b = \frac{1}{2}k_b(\theta - \theta_{in})^2$, adjusting θ_{in} will influence the energy profile of the flexible beam. Specifically, adjusting θ_{in} can shift U_b horizontally since U_b is a convex parabola symmetric about vertical axis $\theta = \theta_{in}$. To illustrate this, we plot the module's ELs (Fig. 3.1C) for three different θ_{in} values: 0°, 40°, and 80°, while maintaining a constant L_c at 17 mm. For these three cases, the energy profile of the spring, U_s, stays the same. However, U_b will shift horizontally to the right when θ_{in} increases, thus changing the module's EL, U. When $\theta_{in} = 0^\circ$, the module's EL is SB. When $\theta_{in} = 40^\circ$, the EL changes to AB. Values of energy barriers are also changed. Compared with the first case ($\theta_{in} = 0^\circ$), E^l_{bar} decreases while E^r_{bar} increases. When $\theta_{in} = 80^\circ$, the peak disappears, eliminating the energy barriers and resulting in an MS configuration.

We then illustrate how adjusting the cable's length (L_c) will influence the module's EL. Since $\Delta(\theta) = \max(0, L(\theta) - L_c - L_{in})$ and $U_s = \frac{1}{2}k_s\Delta^2(\theta)$, adjusting L_c will influence the extension of the spring, thereby affecting the energy profile of the elastic spring. Specifically, changing L_c will scale up or down the energy profile of the spring. We plot the module's ELs (Fig. 3.1D) for three different L_c values: 10, 15, and 20 mm, while maintaining a constant θ_{in} at 0°. For these three cases, U_b stays the same, but U_s will scale down when we increase L_c, thus changing the module's EL, U. The module is SB for each case since both U_s and U_b are symmetric about the vertical axis ($\theta = 0^\circ$). In these three cases, the module with $L_c = 10$ mm has the largest energy barriers (E^l_{bar}

and E^r_{bar}). When L_c increases, the resting angles move closer to $0°$, until the energy peak of the EL (shown as the black dot) finally disappears, and the module becomes monostable.

3.2 Implementations of the Two Tuning Strategies

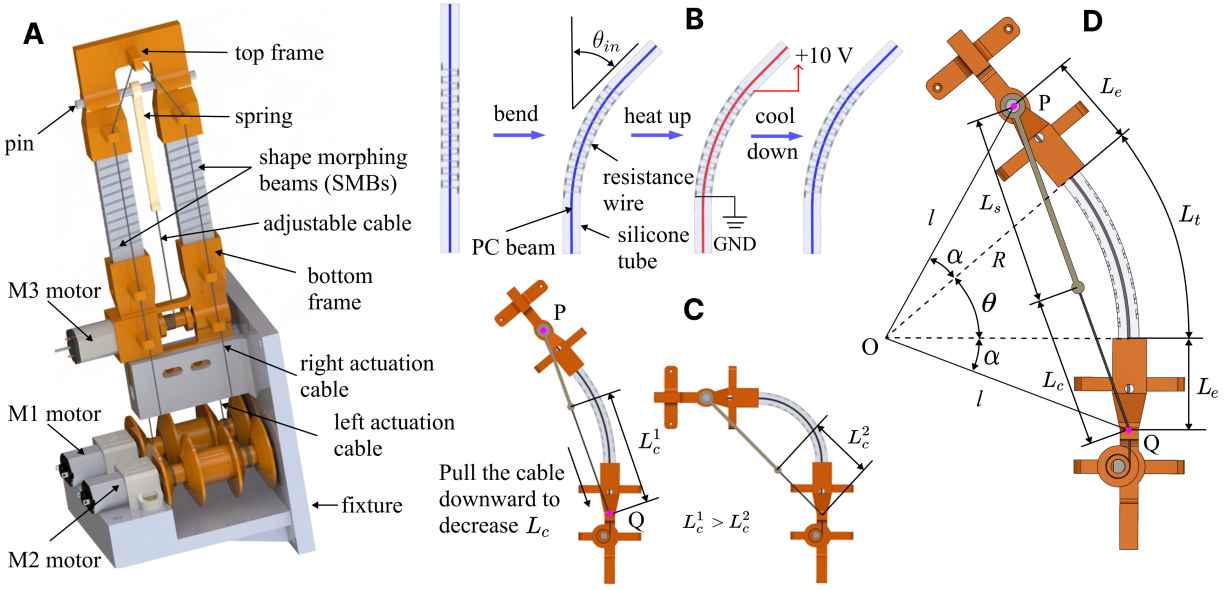


Figure 3.2: Implementation of the two tuning strategies and the forward modeling. (A) Solid model of the tunable module in its straight shape with key components: the Shape Morphing Beams (SMBs), the spring, and three DC motors (M1, M2, and M3). (B) The structure of the SMBs and the procedure to tune their initial bending angle θ_{in}. (C) The procedure to tune the cable's length L_c. (D) Geometric parameters of the module for developing a more accurate forward model.

In this section, we detail how we implement the tunable module and the two tuning strategies for the beam's initial bending angle θ_{in} and the cable's length (L_c). We also develop a more accurate forward model to predict the module's EL to realize three different configurations (SB, AB, and MS) by adjusting θ_{in} and L_c on the fly.

3.2.1 Design of the tunable module

The developed module is made of several parts. We use a straight configuration of the module in Fig. 3.2A for a better illustration. First, it has two shape-morphing beams (SMBs) that can morph

to and then hold another curved shape, thereby adjusting their initial bending angles (details in the next paragraph). One end of these SMBs is connected to a 3D-printed top frame, and the other to a 3D-printed bottom frame, which is mounted to a fixture. An elastic spring made from elastomers (details to be discussed later) is placed in the middle of the two SMBs. One end of the spring is attached to a pin on the top frame, while the other end is attached to an adjustable cable that can be pulled or released by a DC motor M3 (3080, Pololu Corp) placed in the bottom frame. The fixture hosts two additional DC motors labeled as M1 and M2 (3080, Pololu Corp) placed on each side of the module. Actuating M1 will pull the left actuation cable to bend the module to the left, provided that the right actuation cable is slack. Similarly, actuating M2 can pull the right actuation cable to bend the module to the right. The bending angle is negative (positive) if it is bent to the left (right) by the left (right) actuation cable. The attainable range of the bending angle θ is (-192°, 192°), which is determined by the collision of the two frames. In practice, we only bend the module in the range (-160°, 160°), to reduce the influence of permanent plastic deformation of the bending beam. On both the top and bottom frames, several protrusions with holes are designed to facilitate the routing of the actuation cables.

Tuning θ_{in} is accomplished through the SMBs (Fig. 3.2B). Each SMB has three components: a 3D-printed beam made from thermoplastic material polycarbonate (PC), a customized silicone tube encasing the beam, and a resistance wire wrapped around the tube. The beam is initially straight (i.e., $\theta_{in} = 0°$, Fig. 3.2A). Now we can actuate either M1 or M2 to bend the SMBs to a desired angle θ_{in} to the left or right. Then, we heat up the tubes and the PC beams by applying a constant voltage (10 V) across the resistance wires (Fig. 3.2B). The silicone tubes serve a dual purpose: they facilitate uniform heating of the PC beams and ensure consistent bending behavior during the bending motion. After around 200 seconds, the PC beams reach a temperature higher than their glass transition temperature and become soft. Then we maintain the new shape for around 320 seconds while cooling down the SMBs until their temperature drops below the glass transition temperature, and the SMBs regain their rigidity. Finally we release the actuation cables. During this tuning process, we keep the spring slack because, with tension in the spring, the heated

beams may buckle due to the restoring force of the spring. In this way, we can actively adjust the initial bending angle of the module on the fly.

Tuning the length of the adjustable cable, L_c, is realized by using the cable system driven by motor M3. One end of the cable is attached to the lower end of the spring, while the other end is attached to a capstan on the shaft of M3. The cable is routed through a small hole in the bottom frame (see point Q in Fig. 3.2C and D) to keep the effective length of the cable L_C constant when the module undergoes bending motion. In this way, we can actuate M3 to adjust the value of L_c (Fig. 3.2C). M3 is also used to relax the spring before tuning the initial bending angle of the SMBs.

We choose polycarbonate (PolyMax PC, Polymaker Inc) for the beams since it can maintain its initial shape better, i.e., it can almost recover its initial shape after being bent by a large angle and undergoing a large strain, compared with other 3D printing materials (e.g., PLA, PETG, Nylon). We design the dimension of the beam to be $0.6 \times 8 \times 38$ mm, and fabricate the beam through 3D printing (Prusa i3 MK3). The customized silicone tube is molded using Dragon Skin 30 (Smooth-On Inc) with an outside dimension of $4.5 \times 11 \times 38$ mm. A dedicated mold for curing the silicone tube is printed with a FDM printer (Prusa i3 MK3). A beam with the same dimension as those used in the SMBs is inserted to the center of the mold to realize the rectangular hole in the cured silicone tube. After inserting the PC beam to the cured silicone tube, a resistance wire (RW0332, TEMCo Inc) with a resistance of 9 Ω is wrapped outside the tube to complete the fabrication of an SMB. The elastic spring is also molded using Dragon Skin 30 because it can withstand a large tensile strain. We choose the spring's dimension to be $4 \times 2.4 \times 32$ mm. We select the specific dimensions for the beam and the spring such that their strain energies during the bending process are approximately within the same range.

3.2.2 Forward problem: predict the tunable module's EL

With the detailed design, we now establish a more accurate mathematical model for the EL to investigate how tunable elements θ_{in} and L_c will influence the module's EL (i.e., resting angles or

energy barriers). Specifically, we want to solve the forward problem: given the values for θ_{in} and L_c, predict the module's EL.

In the implemented module, the total strain energy has three parts: the strain energies from the tubes U_t, the beams U_b, and the spring U_s.

$$\begin{aligned} U(\theta) &= U_t + U_b + U_s \\ &= 2 \times \frac{1}{2} k_t \theta^2 + 2 \times \frac{1}{2} k_b (\theta - \theta_{in})^2 + \frac{1}{2} k_s \Delta^2(\theta) \end{aligned} \tag{3.4}$$

where k_t is the effective bending stiffness of the two tubes, and other parameters are the same as in Eq. (3.1). Note that $U_t = \frac{1}{2} k_t \theta^2$ because the initial angle of both tubes is always zero, regardless of the value of θ_{in}. Now, since there exists a distance (L_e in Fig. 3.2D) between the end of the bending beam and the spring's attaching point (P or Q), Eq. (3.3) does not hold anymore. Based on the geometry shown in Fig. 3.2D, we have the following equation to relate L and θ.

$$L(\theta) = \begin{cases} 2l \sin(\frac{|\theta|}{2} + \alpha), & \text{if} \quad \theta \in (-\pi, 0) \cup (0, \pi) \\ 2L_e + L_t, & \text{if} \quad \theta = 0 \end{cases} \tag{3.5}$$

where $l = \sqrt{L_e^2 + L_t^2/\theta^2}$, L_t is the length of the tube, $\alpha = \arctan(|\theta| L_e / L_t)$. Eqs. (3.4) and (3.5) can be used to predict the EL of a tunable module given the values of the two tuning parameters θ_{in} and L_c.

We can obtain the derivative of $L(\theta)$ with respect to θ as

$$L'(\theta) = \left[-\frac{2L_t^2}{l|\theta|^3} \sin(\frac{|\theta|}{2} + \alpha) + l \cos(\frac{|\theta|}{2} + \alpha)(1 + \frac{2L_e L_t}{\theta^2 l^2}) \right] \frac{|\theta|}{\theta}$$

for the special case of $\theta = 0$, $L'(\theta) = 0$.

The configuration of the module (i.e., SB, AB, or MS) can be determined by examining the derivative of U with respect to θ. Specifically, if the equation

$$U'(\theta) = 0 \tag{3.6}$$

has three solutions θ_1, θ_2, and θ_3, $U(\theta)$ would have three extrema (one maximum point at θ_2 and two minimum point at θ_1 and θ_3), indicating the module is bistable. The maximum and minima represent the energy peak and stable states, respectively, as shown by the black dots and squares in Fig. 3.1B-D. Further, the module is SB or AB if the magnitudes of the two minima are the same or different. If Eq. (3.6) has one solution, then $U(\theta)$ has one minimum, indicating the module is MS. If Eq. (3.6) has two solutions, $U(\theta)$ is transitioning between bistable and monostable (the maximum point θ_2 and one minimum θ_1 or θ_3 are merged).

Carrying out the derivative, we have

$$U'(\theta) = 2k_t\theta + 2k_b(\theta - \theta_{in}) + k_s\Delta(\theta)L'(\theta) = 0 \tag{3.7}$$

where $\Delta(\theta) = \max(0, L(\theta) - L_c - L_{in})$.

A MATLAB script is developed to numerically solve Eq. (3.6) using the built-in fmincon function to obtain the bending angles (θ_1, θ_2, and θ_3) corresponding to the maximum and minimum points on the energy landscapes (black dots and squares in Fig. 3.1B-D). Based on these obtained bending angles, the energy peak, minimum energies at stable states, and energy barriers can be calculated using Eq. (3.4). In this way, the forward model can be used to both predict the EL of a module and solve for its resting angles and energy barriers.

To understand how the two tuning parameters θ_{in} and L_c influence the module's behavior, we use Eq. (3.6) to plot the bistable and monostable regions in the θ_{in}-L_c parameter space.

First, we experimentally obtain the parameters k_t, k_b, and k_s with bending tests and stretching tests. First, we bend the PC beam using Mark-10 test stand and record the forces and corresponding displacements of the force gauge on the test stand. The strain energy of the PC beam is obtained as the discrete integration of the force over the displacement. We also record a video of the bending motion and use Tracker to analyze the video to obtain the bending angle of the PC beam. Thus, we get the curve of the strain energy U_b versus the bending angle of the beam θ, as shown by the red curve in Fig. 3.3. By curve fitting the result with a quadratic polynomial $U_b = \frac{1}{2}k_b\theta^2$, we approximate the value of k_b to be 8.296 mJ. Then we insert the PC beam to a silicone tube and

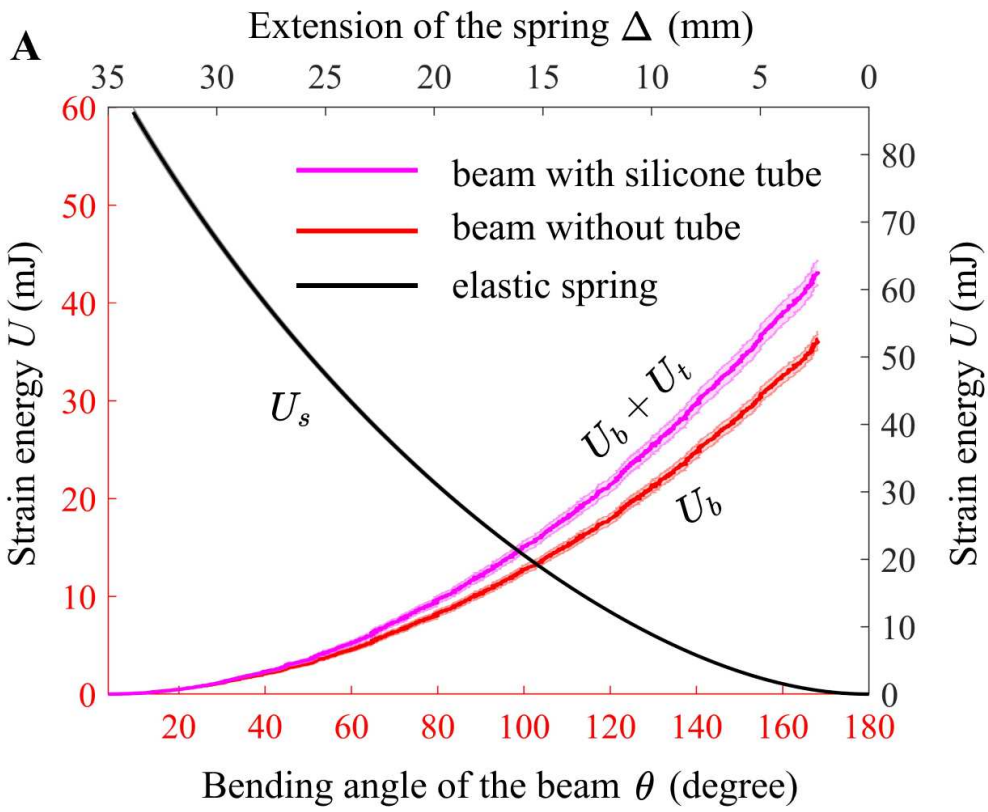


Figure 3.3: The parameters k_t, k_b, k_s are obtained experimentally with bending test or stretching test. Curve fitting is applied on the resulting curves of strain energy versus bending angle (or extension) to approximate the parameters.

repeat the bending test on the beam and tube assembly. The resulting curve of the combined strain energy $U_b + U_t$ versus the bending angle is curve fitted with a quadratic polynomial $U_b + U_t = \frac{1}{2}(k_b + k_t)\theta^2$. In this way, k_t is approximated to be 1.620 mJ. Lastly, we stretch the elastic spring with the Mark-10 test stand and record the forces and corresponding displacements of the force gauge. Similar to bending test, we use a quadratic polynomial $U_s = \frac{1}{2}k_s\Delta^2$ to curve fit the resulting curves of the strain energy of the spring versus its extension. k_s is found to be 0.1518 mJ/mm^2.

We have also limited the range of θ_{in} to $(-\pi, \pi)$ and L_c to $(8, 40)$. We choose this range for L_c because if L_c is too small, the spring force of the module around its straight shape would be very large, leading to the buckling of the beam during the transition. A thorough discussion on the buckling behavior of flexible beams can be found in [70]. If L_c is too large (e.g., 48 mm), then the spring will always be slack, and the module is always monostable.

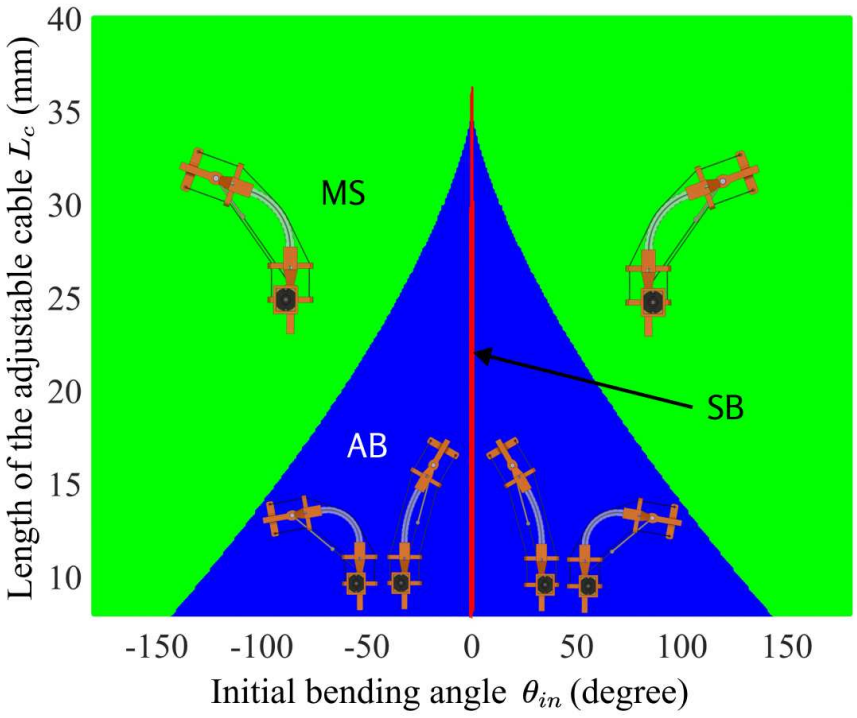


Figure 3.4: Configurations of the module (SB, AB, and MS) depend on the two tuning parameters (θ_{in} and L_c).

In Fig. 3.4, we see that all the regions are symmetric about the axis $\theta_{in} = 0$. The SB configuration is only possible when $\theta_{in} = 0$ and $L_c < 36.1$ mm. When $\theta_{in} = 0$ and $36.1 < L_c < 40$ mm (the spring's energy will decrease if L_c increases, see Fig. 3.1D), the spring's energy profile is insufficient to result in a maximum in the module's EL. So, the module is also monostable. For a given value of $L_c < 36.1$ mm, the module will go from bistable to monostable when the magnitude of θ_{in} increases, as indicated by Fig. 3.1C. At the boundary between bistability and monostability (i.e., the transition between bistable and monostable ELs), Eq. (3.6) has two solutions. The magnitude of θ_{in} at this boundary increases when L_c decreases. This is because, as shown in Fig. 3.1C, if the green curve is higher (corresponding to lower L_c), the red curve needs to shift more (corresponding to larger θ_{in}) horizontally to transition the module's EL from bistable to monostable. Based on the results in Fig. 3.4, we can choose specific values for θ_{in} and L_c of the prototype to enable three different configurations of the EL: SB, AB, and MS.

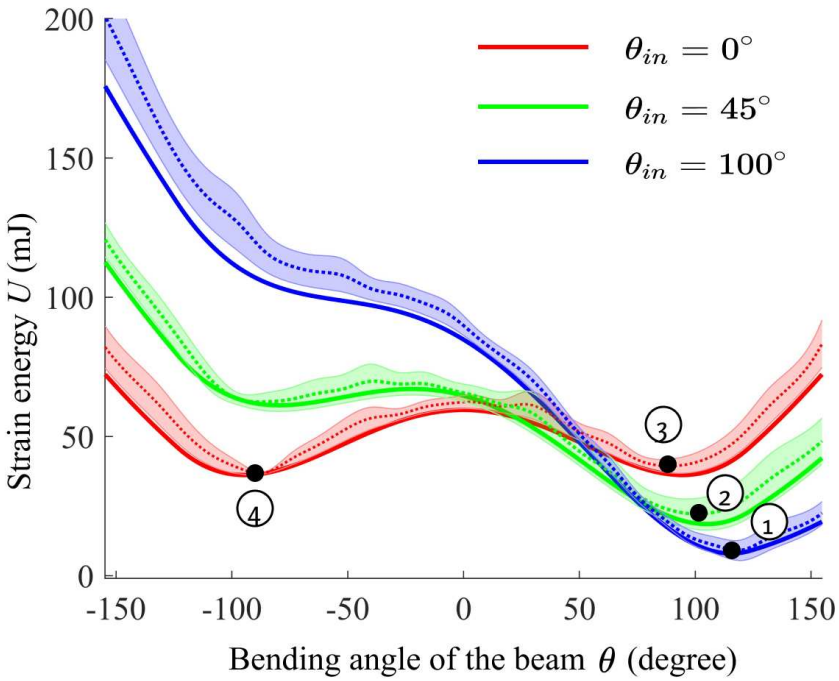


Figure 3.5: Comparisons between the measured and predicted ELs of the tunable module in three different configurations.

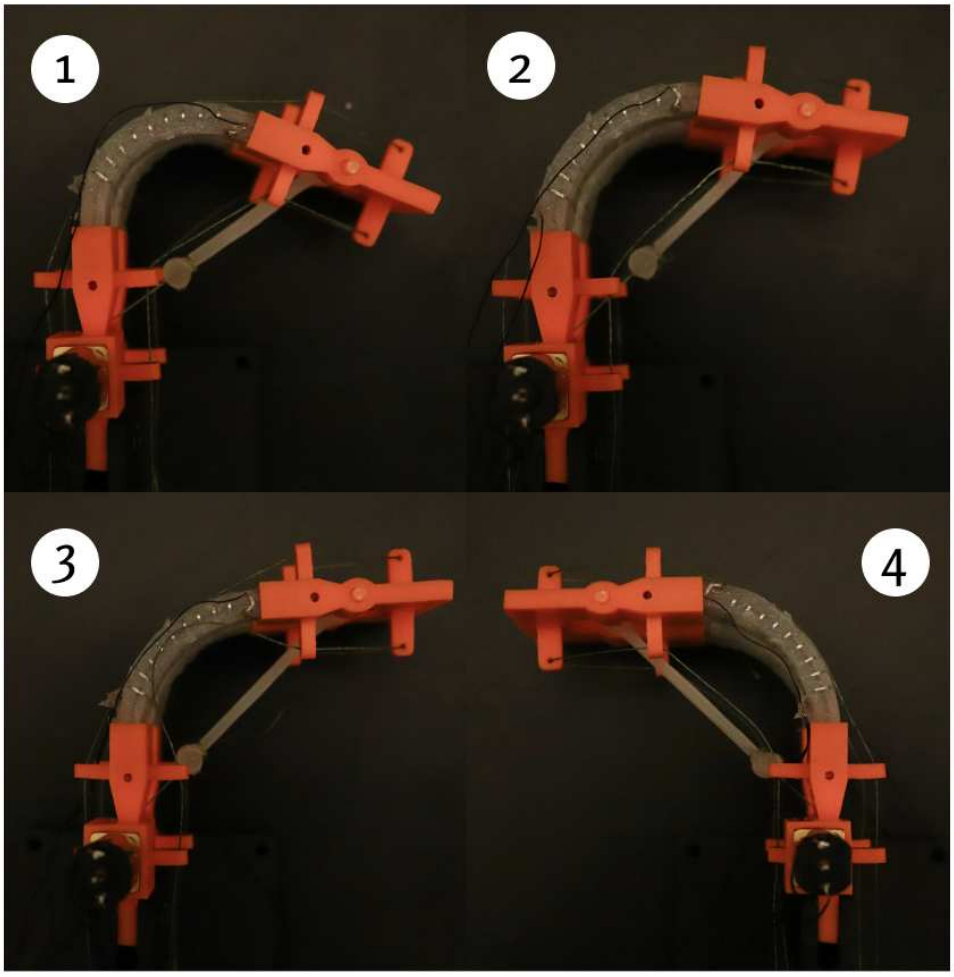


Figure 3.6: Selected pictures of the tunable module in its stable states.

To validate the model we have developed so that we can use it for the inverse problem in the next section, we experimentally obtain the ELs of modules with different configurations (SB, AB, and MS) and compare them with the predicted ELs. We conducted experiments on the same module with different θ_{in} and the same $L_c = 20$ mm. These correspond to three different configurations: SB, AB, and MS, respectively. Since it is difficult to directly measure the strain energy, we derive the strain energy through bending tests of the module to obtain the force on the actuation cable and displacement of the actuation cable. Then we calculate the change of strain energy through discrete integration of the force on the cable over displacement of the cable. For this discrete integration, we choose the initial condition (i.e., the initial energy) as the strain energy at the left stable state (or the right stable state for a monostable module), which can be calculated using Eq. (3.4).

For each module, the bending test is repeated three times to obtain three ELs. Both experimental and predicted ELs are plotted in Fig. 3.5, where the dotted and solid curves are the experimental and predicted ELs, respectively. Shaded regions around the dotted curves show the standard deviations of the measurements. From this figure, we can see that the strain energy model matches reasonably well with the experimental results with an average error of 6.61 mJ and average standard deviation of 8.87mJ. The experimental energies are greater than the predicted ones, possibly because of the friction force between the actuation cables and the silicone tubes. Four images representing typical stable states of the tunable module are illustrated in Fig. 3.6.

3.3 Inverse Problem: Obtain Desired EL Using the Two Tuning Strategies

Using the two tuning strategies, a more interesting problem is to achieve a desired EL. Though the most general case is to achieve precisely the same desired EL (i.e., the exact curve of the EL) for a tunable module, the most important characteristics for the EL are the resting angles (θ_l, θ_r) and the energy barriers (E^l_{bar}, and E^r_{bar}). Ideally, we can achieve both the desired resting angles and desired energy barriers at the same time by tuning the four elements (k_b, θ_{in}, k_s, and L_c)

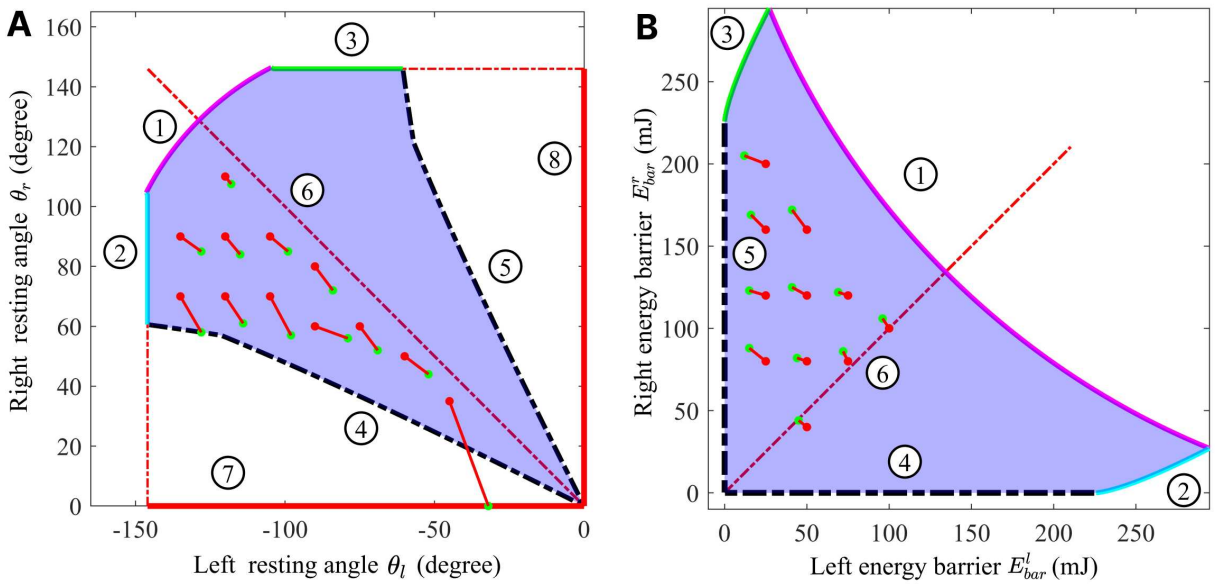


Figure 3.7: Feasible regions of resting angles or energy barriers when $-\pi < \theta_{in} < \pi, 8 < L_c < 40$. (A) Feasible region of resting angles (θ_l and θ_r). (B) Feasible region of energy barriers (E^l_{bar} and E^r_{bar}).

concurrently. However, we only implemented two independent strategies to adjust θ_{in} and L_c on the fly as discussed in section 3.2. Therefore, in this section, we aim to solve the simplified tuning problem: given desired resting angles (θ_l and θ_r) or energy barriers (E^l_{bar} and E^r_{bar}), solve for the values of θ_{in} and L_c. We call this the inverse problem for tuning as opposed to the forward problem discussed in section 3.2-B. We also validate the solution by implementing the two tuning strategies on a prototype and comparing the resulting resting angles (or energy barriers) with the desired ones.

3.3.1 Feasible regions of the resting angles or energy barriers

Before trying to realize the desired resting angles or energy barriers, it's necessary to first obtain the feasible regions for them so that we can choose the desired values in these regions. We first discuss the feasible region for the resting angles, which can be categorized into two cases: when the module is bistable, find the region for two variables θ_l and θ_r, whereas when the module is monostable, only the region for one variable needs to be determined.

When the module is bistable, the feasible region of resting angles θ_l and θ_r depends on two conditions: ranges of the tunable parameters (θ_{in} and L_c) and bistability of the module. We look at these two conditions to determine the values for θ_{in} and L_c corresponding to the boundaries of the feasible region. Using the forward model developed in section 3.2-B, we can obtain θ_l and θ_r resulting from those tuning parameters θ_{in} and L_c. Then θ_l and θ_r are plotted in a 2-dimensional figure to show the boundaries of the feasible region.

For the first condition, we can generate three boundaries: ①, ②, and ③ in Fig. 3.7A based on the ranges of tunable parameters.

$$\text{curve ①:} \qquad L_c = 8, -\pi < \theta_{in} < \pi \tag{3.8}$$

$$\text{curve ②:} \qquad \theta_{in} = -\pi, 8 < L_c < 40 \tag{3.9}$$

$$\text{curve ③:} \qquad \theta_{in} = \pi, 8 < L_c < 40 \tag{3.10}$$

For instance, for the range described by Eq. (3.8), we take sample sets (e.g., $\theta_{in} = \pi/4$ and $L_c = 8$ mm) from this range and use Eq. (3.6) to determine if the module for a certain sample set of θ_{in} and L_c is bistable or not. If bistable, we record the corresponding θ_l and θ_r for that sample set using the forward model. Plotting the resting angles for all the sample sets, we obtain boundary ①. The ranges described by Eq. (3.9) and (3.10) are utilized in a similar manner to get boundaries ② and ③. When $L_c = 40$, the module is always monostable, so this upper bound of L_c is irrelevant in Fig. 3.7A. For the second condition, we can generate two boundaries (④ and ⑤ in Fig. 3.7A), corresponding to the transition of the module from bistability to monostability, which occurs when Eq. (3.6) has just two solutions (see the supplementary material for the detailed procedure to obtain boundaries ④ and ⑤). These two sets of boundaries form a closed region (the blue region in Fig. 3.7A), inside which the input variables $\theta_{in} \in (-\pi, \pi)$ and $L_c \in (8, 40)$ and the module will be bistable. In other words, we can realize any values for θ_l and θ_r inside this region by choosing proper values of θ_{in} and L_c. Curve ⑥ is plotted here to show the SB configurations.

When the module is monostable, we can only obtain the region for one resting angle. In this case, we use θ_l (or θ_r) to designate the resting angle of the module if it is negative (or nonnegative). Curves ⑦ (representing θ_l) and ⑧ (representing θ_r) are also determined by the two conditions mentioned above. It can be seen that the region of θ_l (or θ_r) extends from 0 to the resting angle resulting from tunable parameters $\theta_{in} = -\pi$ (or π) and L_c, when the module transitions from bistable to monostable.

We then try to get the feasible region for the energy barriers E^l_{bar} and E^r_{bar}. When the module is bistable, similar to the case of resting angles, the feasible region of E^l_{bar} and E^r_{bar} also depends on two conditions: ranges of the tunable parameters (θ_{in} and L_c) and bistability of the module. As shown in Fig. 3.7B, the feasible region of E^l_{bar} and E^r_{bar} is enclosed by two sets of boundaries: one (①, ②, and ③) determined by the ranges of the tunable parameters (θ_{in} and L_c) and the other (④ and ⑤) determined by the transition of the module between bistable and monostable states. Curve ⑥ is also plotted here to show the SB configurations. Note that for monostable configurations, the energy barrier does not exist. Additionally, boundaries ④ and ⑤ in Fig. 3.7B lie on the horizontal and vertical axes, respectively, since one energy barrier is 0 when the module transitions between bistable and monostable states.

3.3.2 Obtain desired resting angles or energy barriers

After identifying the feasible regions for the resting angles and energy barriers, we can solve the inverse problem: given desired resting angles ($\overline{\theta}_l$ and $\overline{\theta}_r$) or energy barriers ($\overline{E}^l_{bar}$ and $\overline{E}^r_{bar}$) from the closed regions shown in Fig. 3.7, solve for the corresponding tuning parameters θ_{in} and L_c.

First we investigate the inverse problem for tuning the resting angles. When a module is at a resting angle, the derivative of its strain energy with respect to the bending angle is 0. So the desired resting angles $\overline{\theta}_l$ and $\overline{\theta}_r$ must both satisfy Eq. (3.6), leading to the following two equations with θ_{in} and L_c being the unknowns.

$$(k_s + k_b)\overline{\theta}_l - k_b\theta_{in} + k_s\text{max}(0, L(\overline{\theta}_l) - L_c - L_{in})L'(\overline{\theta}_l) = 0 \tag{3.11}$$

$$(k_s + k_b)\overline{\theta}_r - k_b\theta_{in} + k_s\text{max}(0, L(\overline{\theta}_r) - L_c - L_{in})L'(\overline{\theta}_r) = 0 \tag{3.12}$$

Eqs. (3.11) and (3.12) can be solved to determine the values of the tuning parameters θ_{in} and L_c. In this case, if we tune the module to have the solved θ_{in} and L_c, ideally, the resting angles of the resulting module should be $\overline{\theta}_l$ and $\overline{\theta}_r$.

Table 3.1: List of values for the inverse problem

parameters	$\overline{\theta}_l$	$\overline{\theta}_r$	θ_l	θ_r	$\overline{E}^l_{bar}$	$\overline{E}^r_{bar}$	E^l_{bar}	E^r_{bar}
1	$-135°$	$90°$	$-128°$	$85.0°$	25	200	12.3	205.8
2	$-135°$	$70°$	$-128.4°$	$59.3°$	25	160	16.0	169.4
3	$-120°$	$110°$	$-118.2°$	$107.5°$	25	120	15.5	123.5
4	$-120°$	$90°$	$-115.1°$	$84.8°$	25	80	15.5	88.4
5	$-120°$	$70°$	$-114.4°$	$61.2°$	50	160	41.7	172.8
6	$-105°$	$90°$	$-99.7°$	$85.2°$	50	120	41.2	125.6
7	$-105°$	$70°$	$-98.4°$	$57.3°$	50	80	44.7	82.3
8	$-90°$	$80°$	$-84.1°$	$72.0°$	50	50	46.8	46.2
9	$-90°$	$60°$	$-84.4°$	$56.5°$	50	40	45.5	44.2
10	$-75°$	$60°$	$-69.7°$	$52.2°$	75	120	69.8	122.7
11	$-60°$	$50°$	$-52.4°$	$44.1°$	75	80	72.3	86.3
12	$-45°$	$35°$	$-32.9°$	N/A	100	100	96.7	106.2

To validate the inverse solution obtained through Eqs. (3.11) and (3.12), we choose 12 sets of specific values for $\overline{\theta}_l$ and $\overline{\theta}_r$ in the closed region in Fig. 3.7 (values are shown in Table 3.1). Since the closed region is symmetric about the curve ⑥, we just choose points on one side of the curve. For each set (shown as red dots in Fig. 3.7), we plug in values of $\overline{\theta}_l$ and $\overline{\theta}_r$ to Eqs. (3.11) and (3.12) to solve for the corresponding tuning parameters θ_{in} and L_c and then implement them on the prototype using the two tuning strategies. Finally, we measure the resulting resting angles and compare them with the desired ones $\overline{\theta_l}$ and $\overline{\theta_r}$. The results are shown in Fig. 3.7 and Table 3.1,

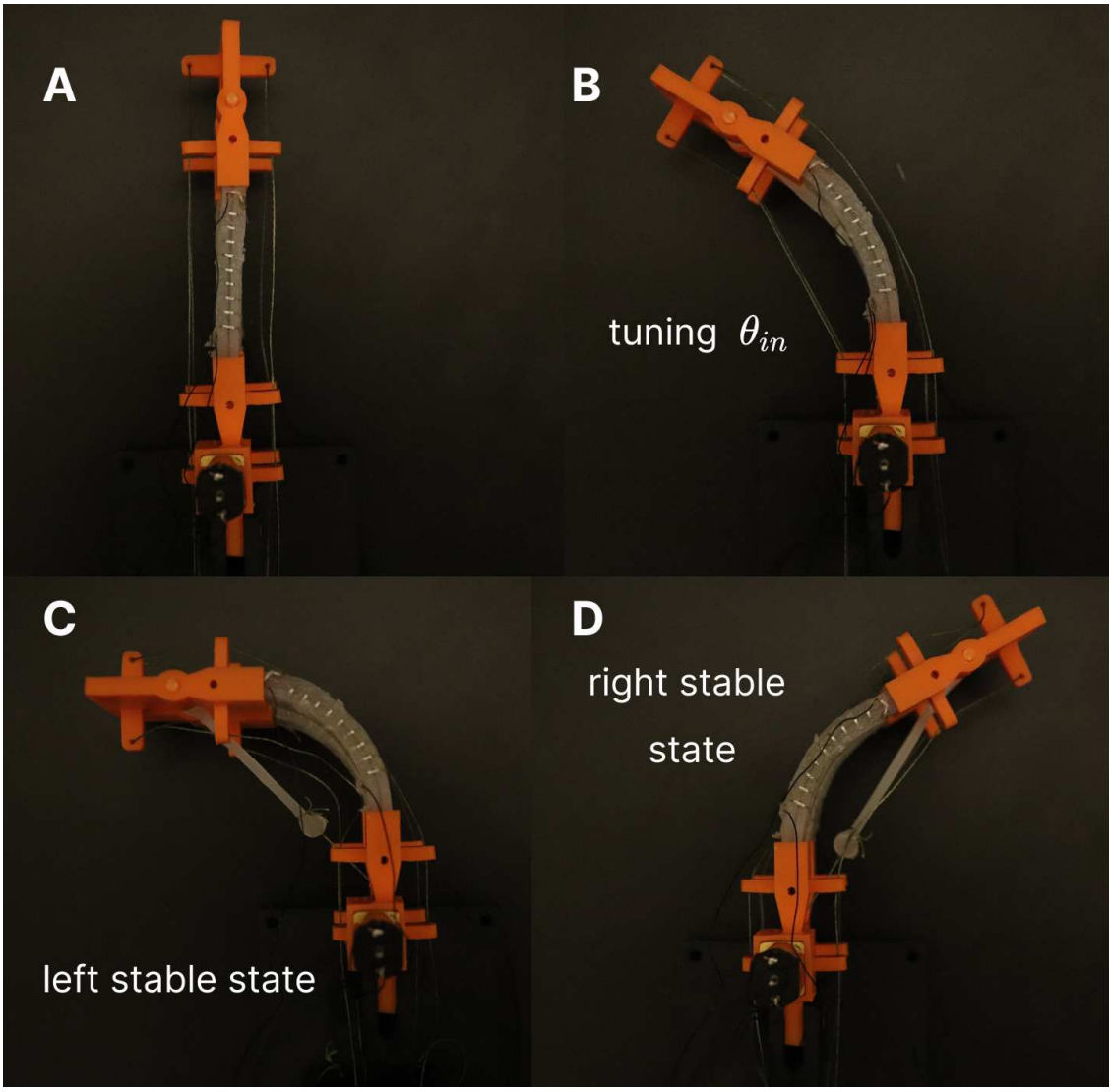


Figure 3.8: The inverse tuning process. (A) a tunable module in its initial straight shape. (B) Bend the module to the solved initial bending angle θ_{in}. (C) The tunable module in its left stable state with $\theta_l = -84.4°$. (D) The tunable module in its right stable state with $\theta_r = 56.5°$.

where the red and green dots represent the desired and the measured resting angles, respectively. For each set, a red line is used to connect the measured to the desired value. It can be seen that they match well with a mean error of 8.2 degrees and a standard deviation of 3.5 degrees. The error may be caused by the viscoelastic behavior and plastic deformation of the bending beams, as well as the deformation of the silicone tubes under high temperatures. For the case with $\overline{\theta}_l = -45^\circ$ and $\overline{\theta}_r = 35^\circ$, the resulted prototype becomes monostable after tuning, instead of bistable as expected. We believe this is because the imperfect behaviors of the bending beams and tubes overwhelm the small energy barrier of the prototype. We also show the pictures of the module with $\overline{\theta}_l = -90^\circ$ and $\overline{\theta}_r = 60^\circ$ at its straight, left stable, and right stable states in Fig. 3.8. For this module, we solve Eqs. (12) and (13) to obtain the two tuning parameters as $\theta_{in} = -57.39^\circ$ and $L_c = 27.11$ mm. After implementing the two tuning strategies, we measure the resting angles of the resulting bistable module to be $\theta_l = -84.4^\circ$ and $\theta_r = 56.5^\circ$.

For the monostable case, we have only one resting angle ($\overline{\theta}_l$ or $\overline{\theta}_r$) and one valid constraint equation (Eq. (3.11) or (3.12)) to solve for θ_{in} and L_c. With only one equation, we can still realize the given resting angle $\overline{\theta_l}$ (or $\overline{\theta_r}$) by first choosing a specific value for L_c and then solving for θ_{in} using Eq. (3.11) (or Eq. (3.12)) based on the chosen L_c.

For the inverse problem for tuning the energy barriers, there are no simple linear equations similar to Eqs. (3.11) and (3.12). Instead, the simplex search optimization method is used to solve for the values of the tuning parameters θ_{in} and L_c given the desired energy barriers $\overline{E}^l_{bar}$ and $\overline{E}^r_{bar}$. The objective function used in the optimization is the error between the desired energy barriers and that resulting from the tuning parameters θ_{in} and L_c. We also conduct experiments to validate the inverse solution for energy barriers obtained through the optimization method. We also choose 12 sets of specific values for $\overline{E}^l_{bar}$ and $\overline{E}^r_{bar}$ in the closed region in Fig. 3.7B. Following a procedure similar to that for the resting angles, we implement the two tuning strategies on a prototype and compare the desired and measured energy barriers. Red and green dots represent the desired and measured energy barriers, respectively. As shown in Fig. 3.7B, they match well with a mean error of 9.7 mJ and a standard deviation of 4.5 mJ. It can be seen that for each set, the measured values

are predominantly situated to the upper left of the desired values. This may be attributed to the permanent deformation of the silicone tubes caused by high temperatures when implementing the tuning strategies.

3.4 Applications

In this section, we employ the tunable modules for three different applications to demonstrate the utility of tuning EL on the fly.

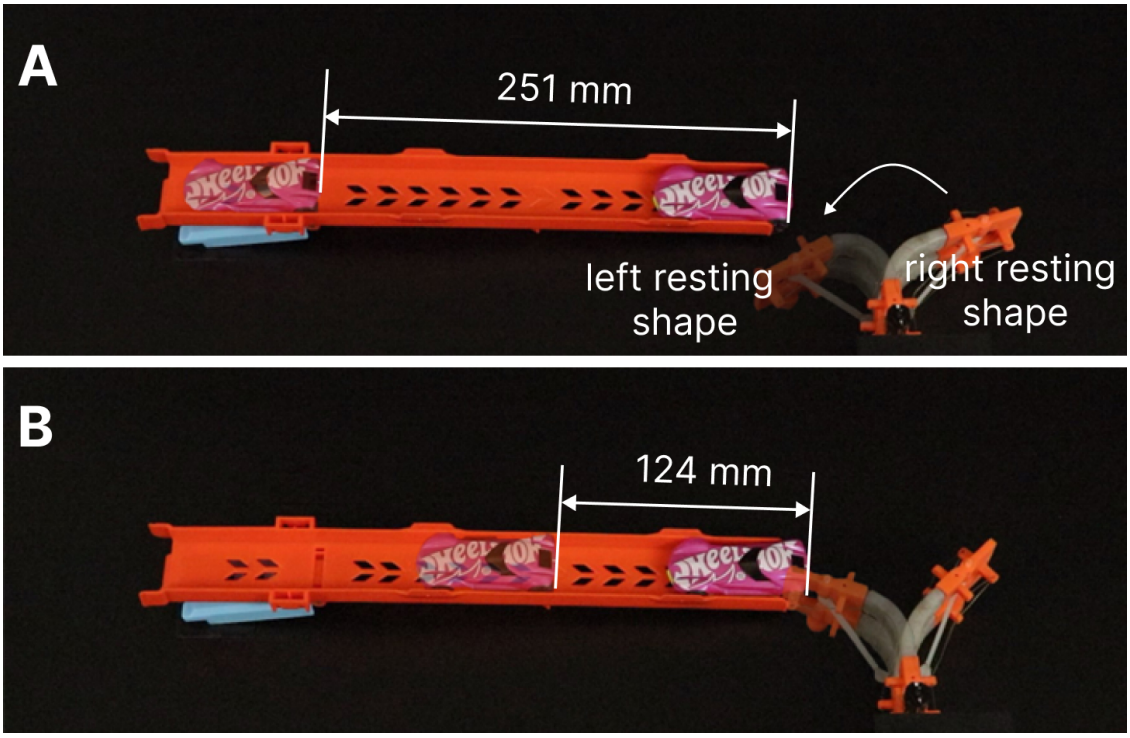


Figure 3.9: A kicker based on a single tunable module has adjustable kicking power. (A) When $\theta_{in} = -\pi/4$ and $L_c = 10$ mm, the toy car travels by 251 mm. (B) When $\theta_{in} = -\pi/8$ and $L_c = 18$ mm, the car travels by 124 mm.

3.4.1 An adjustable kicker with a single tunable module

We first demonstrate a single tunable module as a kicker that can change its kicking power by adjusting its energy barrier. The tunable module's bottom frame is fixed, while its top frame kicks a toy car on its trajectory during its transition motion. It is initially in its right resting shape. The toy

car is placed inside a sloped track to constrain its motion. As shown in the supplementary video and Fig. 3.9A, we first let the module have $\theta_{in} = -\pi/4$ and $L_c = 10$ mm, which results in a left energy barrier of $E^l_{bar} = 77.2$ mJ. If we actuate the left cable, the module would first gradually move to the left. Subsequently, upon surpassing the energy peak, it would swiftly release part of its stored strain energy and collide with the toy car during its transition motion. The collision would make the toy car proceed along the sloped track. In this case, the toy car travels a distance of 254 mm. Then, if we want to decrease the traveling distance to approximately half of its current value, we can tune the initial bending angle and the cable length of the tunable module to $\theta_{in} = -\pi/8$, $L_c = 18$ mm, respectively. This tuning will generate an energy barrier of $E^l_{bar} = 35.0$ mJ, approximately half of the previous energy barrier. After tuning, we repeat the experiment and find the toy car travels a distance of 124 mm (Fig. 3.9B), also about half of the previous travel distance.

3.4.2 A reconfigurable arm with two tunable modules

We then connect two modules in series to realize a reconfigurable arm, which can change its resting shapes by employing the two tuning strategies individually. As shown in Fig. 3.10A, each module of the arm has its own motor controlling its cable length, L_c, respectively, but they share the same side motors at the base for actuation. To do this, we route a single cable in a closed loop on each side of the arm. The arm can generate different final shapes by adjusting the resting shapes of both modules independently. Without loss of generality, we show six typical shapes in the arm's workspace.

The first shape is a straight one (Fig. 3.10A), which can be realized by making actuation cables on both sides and adjustable cables on both modules slack. For the second shape, the first module bends left, while the second is straight (Fig. 3.10B). To generate this shape, we tune the first module to be monostable, with its left resting angle being $\theta^1_l = -40^\circ$. We achieve this resting angle by tuning the first module's initial bending angle to $\theta^1_{in} = -62.2^\circ$. Its adjustable cable is slack.

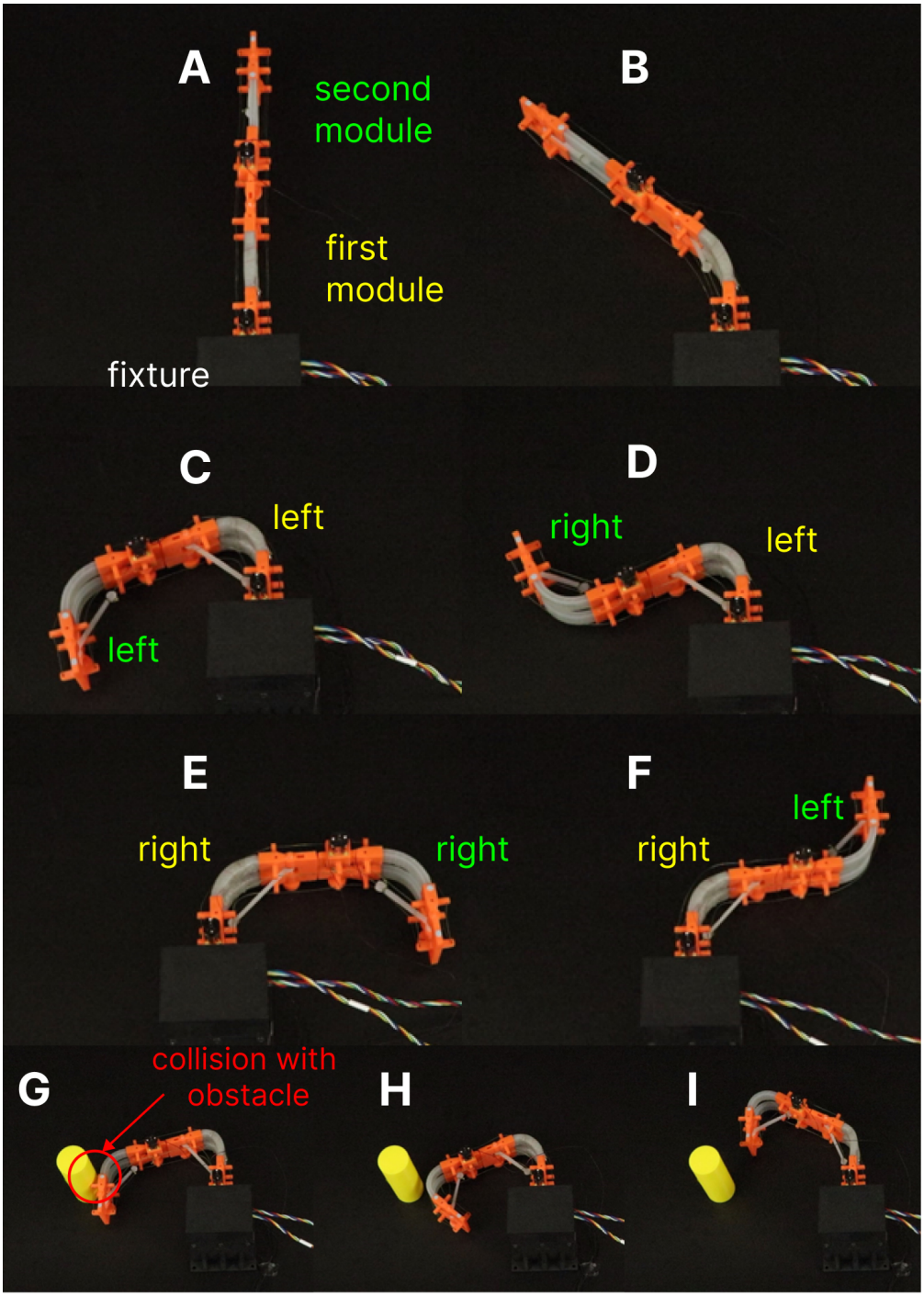


Figure 3.10: A reconfigurable arm consisting of two modules connected in series can generate different final shapes under the same actuation. Yellow and green texts are for the first and second modules, respectively. The arm is connected to a fixture via the bottom frame of the first module. (A)-(F) are six typical shapes of the arm. (G) Following one route would make the arm collide with an obstacle. (H) and (I) are two sequential images of the arm following another route to avoid colliding with the obstacle.

Shapes C, D, and E share the same initial bending angles and cable lengths for each module. It means the tuning strategies are not required to reconfigure between these three shapes. The first module is AB with $\theta_l^1 = -120^\circ$ and $\theta_r^1 = 60^\circ$, while the second module is SB with $\theta_l^2 = -90^\circ$ and $\theta_r^2 = 90^\circ$. As discussed in 3.3, we can solve for the corresponding values of tuning parameters θ_{in} and L_c for each module via the inverse process of tuning: $\theta_{in}^1 = -88.4^\circ$, $L_c^1 = 13.4$ mm, $\theta_{in}^2 = 0^\circ$, $L_c^2 = 21.2$ mm. In shape C (Fig. 3.10C), the second module has a smaller left energy barrier than the first module. When we actuate the right actuation cable, the second module will transition first (from left to right resting shape) while the first module stays around its left resting shape. In this way, we obtain shape D (Fig. 3.10D). If we pull the right actuation cable further, the first module will also transition to its right resting shape to generate shape E (Fig. 3.10E). Under the actuation of the right cable, the reconfigurable arm changes from shape C to E through shape D.

By implementing the tuning strategies, we can also make the first module transition first, thus the arm changes from shape C to shape E through another intermediate shape F (Fig. 3.10F). To do this, we reduce the value of L_c^2 from 21.2 mm to 12 mm to increase the left energy barrier of the second module and increase the value of L_c^1 from 13.4mm to 21.4 mm to decrease the left energy barrier of the first module. In this way, the first module would transition first when the right actuation cable is pulled, followed by the second module.

Changing the transition sequence (i.e., which module transitions first) can be useful in practical applications. For instance, it can be used to adjust the arm's trajectory to avoid collision with an obstacle (see the supplementary video). As shown in Fig. 3.10G, both modules are initially in their left resting shapes. We want to transition the arm to another shape where both modules are in their right resting shapes without colliding with the yellow cylinder obstacle. Following the first route mentioned above (from shape C through D to E) would make the second module collide with the obstacle. Following the latter route (from shape C through F to E) would successfully make the arm finish the transition without colliding with the obstacle. Two intermediate pictures of the arm during this transition are shown in Fig. 3.10H and I.

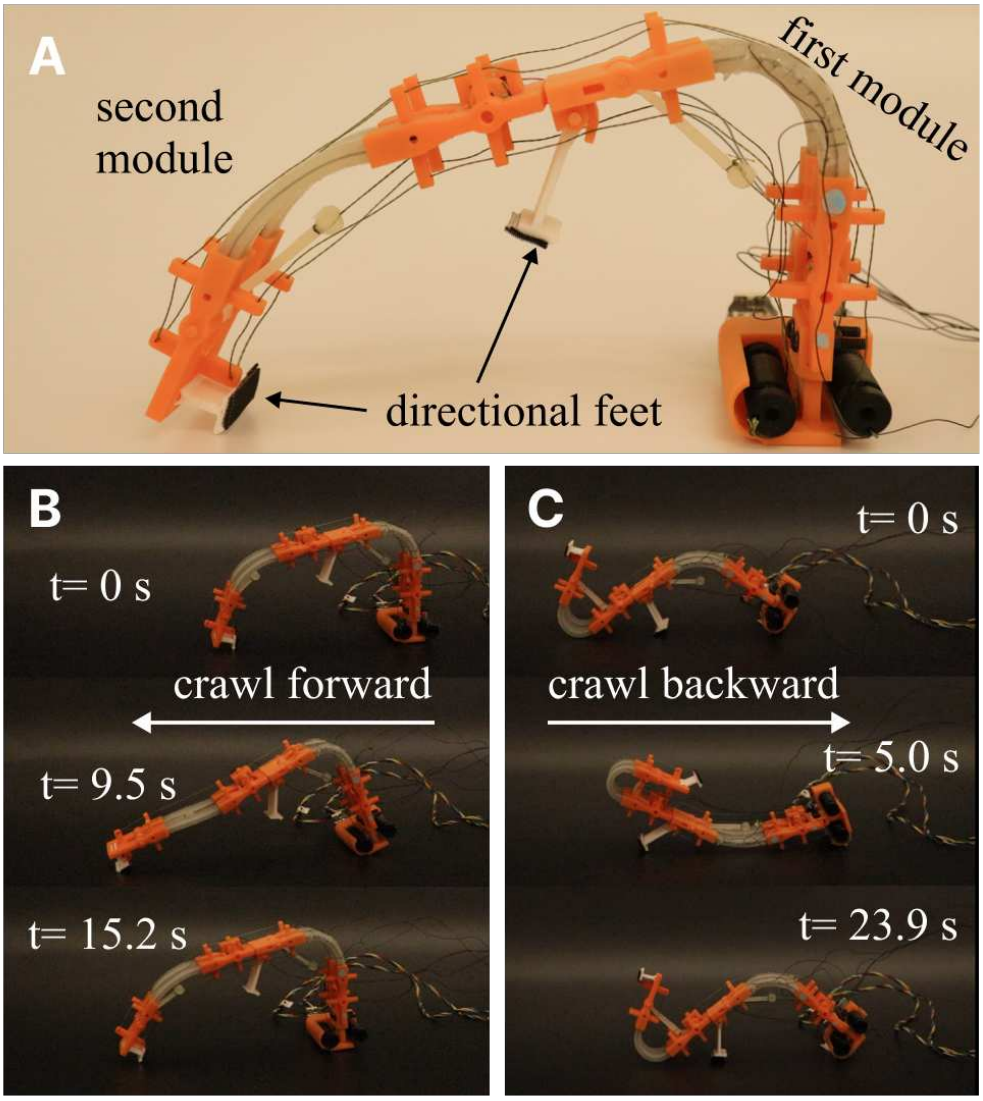


Figure 3.11: Using the tuning strategy to change the locomotion direction of the crawling robot. (A) To realize the crawling locomotion, two feet capable of directional friction force are attached to the top frame of each module. (B) Some sequential images of the robot crawling forward. (C) Some Sequential images of the robot crawling backward.

3.4.3 A reconfigurable crawling robot

We further develop a crawling robot to demonstrate that the tuning strategies can be used to change its locomotion direction (see the supplementary video). The robot is made from two modules connected in series, similar to the reconfigurable arm. It has a 3D-printed directional foot on each module (Fig. 3.11A). The foot on the first and second module is used for backward and forward locomotion, respectively. To crawl forward, we let the first module maintain its left resting shape while the second module is bent upward by the right actuation cable and downward by the left actuation cable repeatedly (Fig. 3.11B). We tune the second module with $\theta_{in}^2 = -30^\circ$ and $L_c^2 = 25$ mm, which results in energy barriers of $E_{bar}^l = 22.5$ mJ and $E_{bar}^r = 0.7$ mJ. We tune the first module with $\theta_{in}^1 = -90^\circ$ and $L_c^1 = 15$ mm, which results in a left energy barrier of $E_{bar}^l = 91.5$ mJ. Both energy barriers of the second module are much smaller than the left energy barrier of the first module. This ensures that when we alternatively actuate the two actuation cables, the first module stays around its left resting shape, while the second module can switch between its two stable states. In this way, the directional foot on the second module can make the robot crawl forward, while the foot on the first module does not touch the ground.

We can change the robot's locomotion direction by tuning the modules' parameters on the fly. To realize the backward crawling (Fig. 3.11C), we first move the foot on the second module away from the ground by switching the second module to its right resting shape. Now, the foot on the first module will touch the ground. We then tune the second module with $\theta_{in}^2 = 90^\circ$ and $L_c^2 = 8$ mm, resulting in a right energy barrier of 161.2 mJ. We also tune the first module with $\theta_{in}^1 = -30^\circ$ and $L_c^1 = 25$, which results in energy barriers of $E_{bar}^l = 22.5$ mJ and $E_{bar}^r = 0.7$ mJ. Both energy barriers of the first module are much smaller than the right energy barrier of the second module. This ensures that when we alternatively actuate the two actuation cables, the second module stays around its right resting shape, while the first module will switch between its two stable states. In this way, the directional foot on the first module would make the robot crawl backward.

3.5 Chapter Summary

In this chapter, we present methods to tune the energy landscape (EL) for a beam-based bistable module with elastic instabilities on the fly without manual alterations. The two tuning strategies we developed – adjusting the beam's initial bending angle and varying the offset of the spring's extension – have proven effective in manipulating the EL, as evidenced by our forward model and its subsequent experimental validation. By connecting multiple bistable modules in series, we can realize a reconfigurable robot with different resting shapes, energy barriers, or transition sequences. We have also successfully solved the inverse problem to obtain desired resting angles or energy barriers through the two tuning strategies. We also illustrate the practical applications of tuning through the successful deployment of the tunable module in three different scenarios: as a kicker to impart different energies to an object, as a reconfigurable arm that can reconfigure its resting shape, and as a crawling robot that can crawl in both directions using the same actuation.

Chapter 4

Co-Optimization of Design and Control

For the reconfigurable robots presented in Chapters 2 and 3, we need to determine the reconfiguring parameters and corresponding control policies to achieve desired functionalities. We can treat the reconfiguring parameters (stiffnesses of the SVJs in the case of origami module based robots, initial bending angle and offset of the spring in the case of bistable module based robots) as design parameters since they are normally tuned and fixed before carrying out a specific function. These parameters influence the robot's kinematics and dynamics, thereby affecting its interactions with the environment and the optimal control strategy to accomplish specific tasks. Consequently, optimizing both reconfiguring parameters and control policies simultaneously is essential for reconfigurable robots, a challenge known as the co-design or co-optimization problem.

To state the co-optimization problem for robotic systems more generally, a robot relies on both its physical design and control algorithm for a specific function. These two aspects are inherently coupled, as the robot's dynamics—shaped by its physical design parameters—directly influence the development of its control strategies. Robots with different designs may have entirely different control strategies. Designing a robot requires simultaneous optimization of both its physical design and control algorithm. To achieve this, a co-optimization process should be employed, aiming to obtain an optimal combination of design and control tailored to the specific environment and task requirements.

Many researchers have explored co-optimization approaches that jointly reason over physical design and control for legged robots [47, 49, 50], soft robots [48], robotic hands [64, 71]. For instance, Spielberg et al. [47] demonstrates that robot design parameters can be incorporated seamlessly into the trajectory optimization process, enabling the concurrent optimization of robot trajectories and physical designs. Deimel et al. [48] use a simplified model for the dynamics of the soft robotic hand in the co-design process, which updates the design parameters with particle filter optimization method. Also, co-optimization can be implemented as an iterative process, alternat-

ing between optimizing the physical design and refining the control algorithm. Liao et al. [51] proposed hierarchical process constrained batch Bayesian optimization (HPC-BBO), to automatically optimize robot morphologies and the controllers in a data efficient manner. HPC-BBO is a hierarchical Bayesian optimization process which iteratively optimizes morphology configurations and subsequently learns the corresponding controllers. It is worth noting that the aforementioned co-optimization methods rely on detailed dynamic models of the robots, which can often be highly complex or, in some cases, entirely unavailable.

Data-driven approaches, such as deep reinforcement learning (RL), have proven highly effective in addressing the complex dynamics of robotics and their interactions with the environment. Control policies modeled as neural networks and trained through deep reinforcement learning have become highly successful, achieving cutting-edge performance across a wide range of robotic tasks [60, 72, 73]. Recently, researchers have also explored implementing co-optimization using reinforcement learning methods [61–66]. For instance, He et al. [63] proposed the MORPH framework to co-optimize robot morphology and control policy to achieve a reaching task and a button-pressing task. It employs a neural network based proxy model to approximate the real physical model of the robot. The divergence between the proxy model and real physical model (measured using random samples) is used as a regularization term in the objective function of the reinforcement learning process. Wang et al. [74] propose Neural Graph Evolution to co-evolve both the robot design and the control policy, representing the morphology of the robot with a graph neural network.

Unlike traditional model-based co-optimization methods, which rely heavily on accurate physical models and often struggle with the complexity of real-world dynamics, RL-based co-optimization methods excel in learning directly from interactions with the environment. This makes them particularly effective for handling the intricate and dynamic behaviors of reconfigurable robots. In this chapter, we explore the co-optimization of reconfigurable robots, integrating the design of reconfiguring parameters with the development of control policies for a given task, using reinforcement learning methods.

4.1 Using RL to Co-optimize Reconfigurable Robots

The reinforcement learning problem is generally formulated as a Markov Decision Process (MDP). An MDP can be represented by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{F}, r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{F}$ is the state transition model, r is the reward function. An agent in state $s_t \in \mathcal{S}$ at time t takes action $a_t \in \mathcal{A}$ according to some policy π_θ, and the environment returns the agent's new state $s_{t+1} \in \mathcal{S}$ according to the state transition model $\mathcal{F}(s_{t+1}|s, a)$, along with the associated reward $r_t = r(s_t, a_t)$. The goal is to learn the optimal control policy $\pi^*_\theta : \mathcal{S} \to \mathcal{A}$ mapping states to actions that maximizes the expected return

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} [R(\tau)]$$

where τ is a trajectory obtained by letting the agent act in the environment using the policy π_θ, $R(\tau) = \sum_{i=0}^{T} \gamma^i r_{t+i}$ is the return for the trajectory τ, T is the length of the trajectory, $\gamma \in [0, 1)$ is the discount factor of the future rewards.

A stochastic policy, denoted as $\pi_\theta(a_t \mid s_t)$, is commonly used to predict the action a_t given the current state s_t. The stochastic nature of the policy encourages exploration during the training process, enhancing the model's ability to discover optimal actions. Typically, $\pi_\theta(a_t \mid s_t)$ is modeled as a neural network, which takes the current state s_t as input, and outputs a probability distribution for sampling the action a_t. In most cases, a Gaussian distribution is used for the probability distribution, where the neural network outputs the mean and standard deviation for each dimension of the action a_t. In the case of complex continuous action spaces, policy gradient methods are commonly used to learn the parameters θ of the policy through stochastic gradient ascent on the expected return.

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) R(\tau) \right] \tag{4.1}$$

"Vanilla" policy gradient methods using the formula shown above struggles with high variance and slow convergence.

In this work, we use the Proximal Policy Optimization (PPO, [75]) algorithm to learn an optimal control policy for our reconfigurable robots (e.g., origami manipulator) to achieve specific tasks (e.g., reaching task). PPO offers significant advantages over traditional policy gradient methods by providing a stable and efficient training process. The actor-critic framework, which uses an actor network to determine the policy and a critic network to estimate the value function, enables PPO to reduce variance in policy gradient updates and improve learning efficiency. Unlike vanilla policy gradients, PPO employs a clipped surrogate objective that prevents large policy updates, ensuring stable learning. Additionally, PPO does not require the complex computation of trust regions used in Trust Region Policy Optimization (TRPO, [76]), making it simpler to implement while retaining comparable performance. Its balance between exploration and exploitation, combined with scalability and robustness, makes PPO a widely adopted algorithm in reinforcement learning.

PPO is an on-policy algorithm that alternates between sampling data from the environment and optimizing the objective:

$$\hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{4.2}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the ratio function, $\hat{A}$ is the advantage function, ϵ is a small hyperparameter which roughly says how far away the new policy is allowed to go from the old one. The clipped objective has the effect of maximizing expected return by making only small steps in policy space at a time.

We note that, for reconfigurable robots, the reconfiguring parameters and control sequence are inherently coupled since the reconfiguring parameters will influence the state transition model (e.g., kinematic model of the origami manipulator). Robots with different configuration will naturally have different kinematics (or dynamics) and optimal control policies π_θ^*. This means that we should learn the optimal reconfiguring parameters and control policy simultaneously. For example, in an origami manipulator composed of four origami modules connected in series, different stiffness configurations can lead to varying feasible workspaces. This variation occurs because

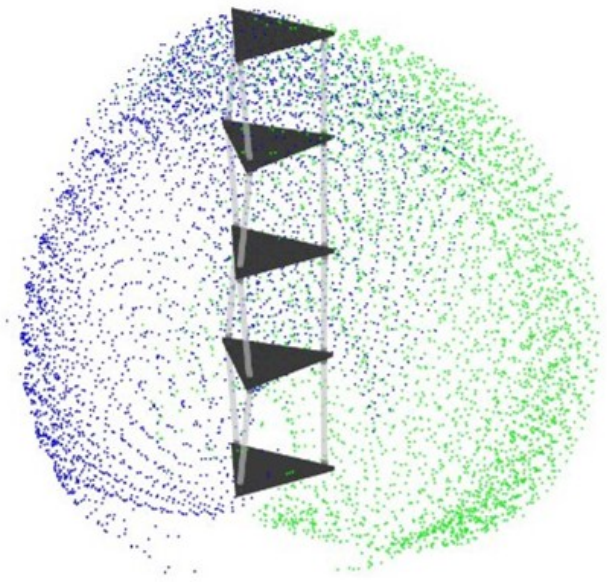


Figure 4.1: Different stiffnesses result in different workspace

higher stiffnesses demand greater actuation forces on the cable, which may exceed the maximum capacity of the actuation motor. Two typical workspaces of a manipulator of different stiffness configurations are shown in Fig. 4.1. To enable the manipulator to reach a specific goal point in the space, it is not sufficient to learn the control policy (displacement of the actuation cable) alone. It is also necessary to determine a specific stiffness configuration whose workspace encompasses the desired goal point.

In this work, we co-optimize the reconfiguring parameters and control policy of the robots using the algorithm presented in [77], which is inspired by the parameter exploring strategy proposed in [78]. The overview of this co-optimization approach is depicted in Fig. 4.2. The goal of our co-optimization is to enable the discovery of optimal reconfiguring parameters $\omega^* \in \Omega$ from the space of feasible reconfiguring parameters Ω that maximizes the agent's success when used in conjunction with a corresponding optimal control policy π_θ^*. Here we note that ω denotes the stiffnesses of the SVJs in the case of origami module based robots, while in bistable module-based robots, it represents the initial bending angles and spring offsets. Instead of treating policy training as a black-box optimization that we perform independently for each possible configuration,

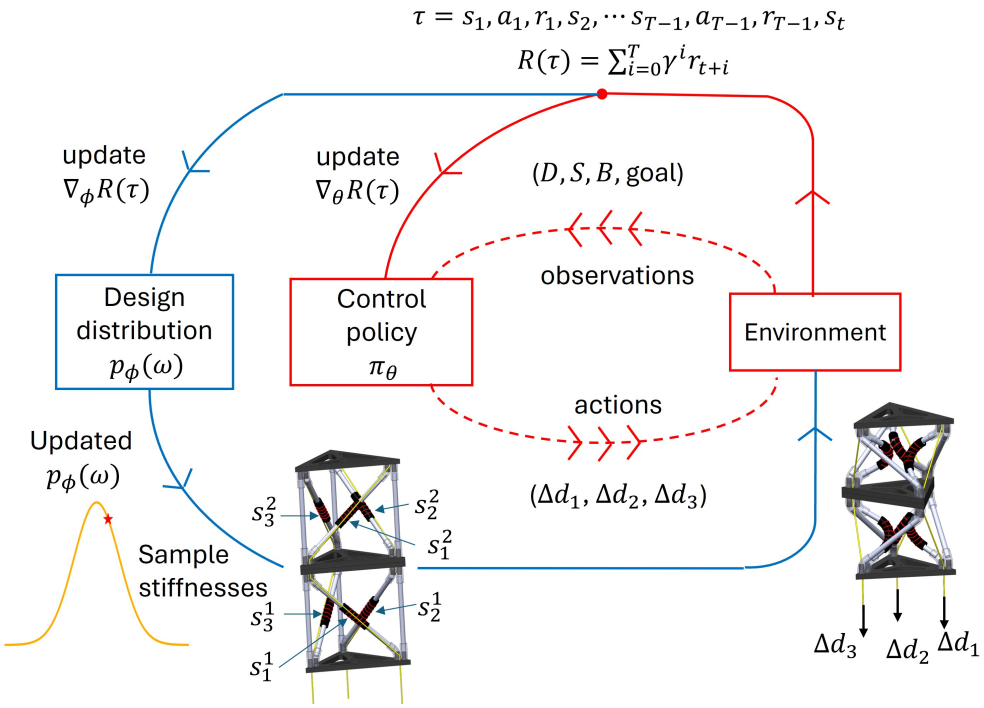


Figure 4.2: overview of the co-optimization approach

we search over configuration simultaneously with a search over policies. Basically, we extend the standard reinforcement learning formulation to also include ω as a learnable parameter. We describe the co-optimization process as follows.

Let $p_\phi(\omega)$ denote the Gaussian distribution of the possible reconfiguring parameters. The learnable parameters ϕ, comprising the means and standard deviations of the distribution, encapsulate the framework's belief about which designs are more likely to result in greater returns. The policy function $\pi_\theta(a_t|s_t, \omega)$ is now provided with the parameters ω of the configuration it is controlling as input.

Formally, we seek to find configuration and policy parameters ϕ^* and θ^* such that:

$$\phi^*, \theta^* = \arg\max_{\phi,\theta} \mathbb{E}_{\omega \sim p_\phi} \left[\mathbb{E}_{\pi_\theta} [R_\tau] \right]. \tag{4.3}$$

At each iteration of training, the policy is trained (using PPO) to maximize the expected return over configurations sampled from the current configuration distribution p_ϕ. At the same time,

the configuration distribution is updated every iteration to increase the probability density around designs that perform well when using the current learned policy π_θ [77]:

$$\nabla \mathbb{E}_{\omega \sim p_\phi}\left[\mathbb{E}_{\pi_\theta}[R_t]\right] = \mathbb{E}_{\omega \sim p_\phi}\left[\nabla \log p_\phi(\omega) \mathbb{E}_{\pi_\theta}[R_t]\right]. \tag{4.4}$$

This shifts the parameters of the configuration distribution ϕ to maximize the expected reward under the current policy π_θ, and is analogous to gradient-based updates to the policy parameters. We parameterize the configuration distribution p_ϕ as a Gaussian distribution, which is initialized with random mean μ and large standard deviation σ to cover the whole parameter ranges.

$$p(\omega_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(\omega_i - \mu_i)^2}{2\sigma_i^2}\right), \tag{4.5}$$

The term $\nabla \log p_\phi(\omega)$ in Eq. (4.4) is calculated as follows [78].

$$\nabla_{\mu_i} \log p(\omega_i) = \frac{\omega_i - \mu_i}{\sigma_i^2}, \tag{4.6}$$

$$\nabla_{\sigma_i} \log p(\omega_i) = \frac{(\omega_i - \mu_i)^2 - \sigma_i^2}{\sigma_i^3}. \tag{4.7}$$

4.2 Co-optimization for Origami Robots based on Forward Kinematics

In this section, we try to apply the co-optimization learning algorithm on an origami manipulator consisting of multiple modules connected in series to achieve some tasks. First, we introduce the improvement on the design of the origami modules to enhance controllability. Then we try to make the end effector point of the origami manipulator reach a specific goal point while avoiding an rectangular obstacle.

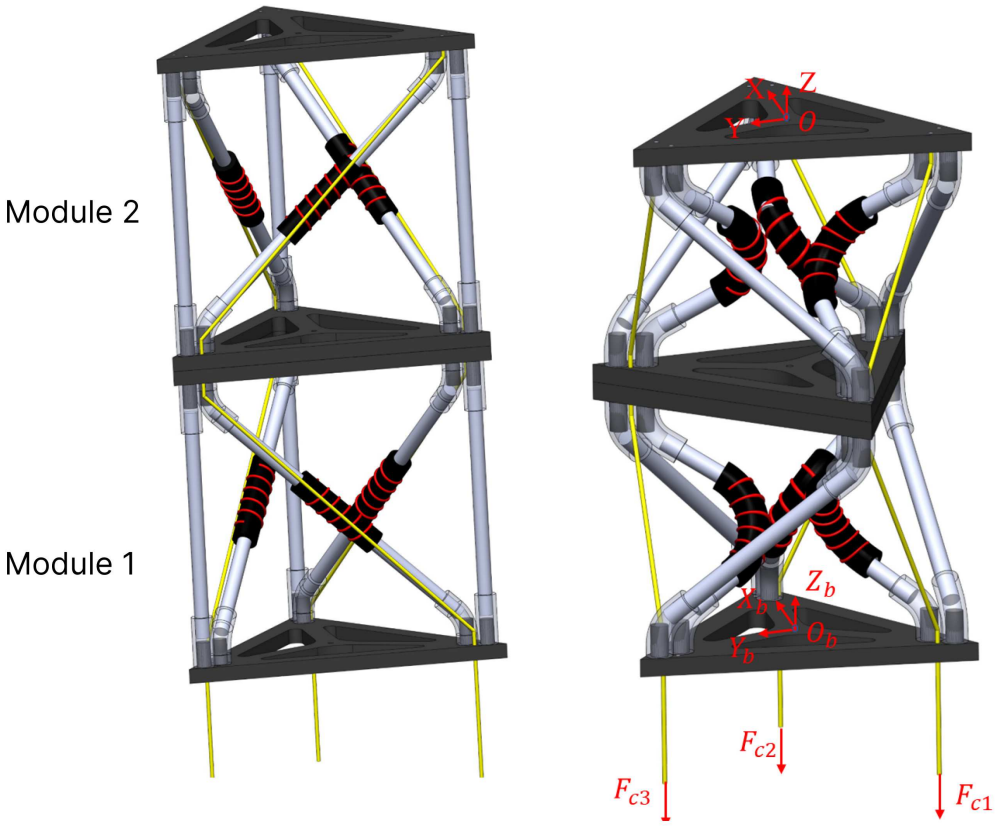


Figure 4.3: Improvement on the design of the origami manipulator. The bottom plate of module 1 is fixed.

4.2.1 Improved Design with Three Tendons and the Associated Modeling

In our previous design, we used a single cable (shown in yellow in Fig. 2.3) to apply the force on the top plate to actuate the reconfigurable origami module. To achieve more possible shapes and motions upon actuation, we improve the design to use three independent tendons on three different sides of the module to actuate it. The tendons (shown in yellow in Fig. 4.3) are routed along the diagonal links in a zigzag pattern on each side. Although an origami manipulator can comprise more modules, only two modules are shown in Fig. 4.3 for clarity. Also, the stiffnesses of the SVJs in the previous design were binary (either soft or rigid). We plan to employ a new type of SVJ (made from layers of shape memory polymers and thermoplastics) capable of continuously adjusting its stiffness [79]. By assuming continuously adjustable stiffness, the forward model becomes more general and extends its applicability to a wider range of scenarios.

We note that the shape of a module is uniquely represented by the lengths of its three virtual straight diagonal links $B = [b_1^i, b_2^i, b_3^i]^T$ (see Eqs. 3-8 in Chapter 2). The superscript denotes the numbering of the module in the manipulator (see Fig. 2.7 and Fig. 4.3). The subscript denotes the numbering of the facets in each origami module. If N modules are connected in series to form an origami manipulator, then the column vector B has $3N$ entries.

$$B = [b_1^1, b_2^1, b_3^1, b_1^2, b_2^2, b_3^2, ..., b_1^N, b_2^N, b_3^N]^T \tag{4.8}$$

The stiffnesses of all joints are similarly represented as a vector of $3N$ entries.

$$S = [s_1^1, s_2^1, s_3^1, s_1^2, s_2^2, s_3^2, ..., s_1^N, s_2^N, s_3^N]^T \tag{4.9}$$

For the original module design with only one actuation tendon, we have a constraint equation (Eq. (2.9)) relating the shape B to the given cable displacement Δ.

$$\Delta = \sqrt{(d_x^{ini})^2 + (d_y^{ini})^2 + (d_z^{ini})^2} - \sqrt{d_x^2 + d_y^2 + d_z^2} \tag{4.10}$$

where d_x^{ini}, d_y^{ini}, d_z^{ini} represent the position of the center of the top plate before the module is deformed by the cable. d_x, d_y, d_z represent the position of the center when the module is actuated by the cable. The aforementioned positions are determined by the shape B of the modules (see Eqs. 3-8 in Chapter 2).

Now, with the improved modules actuated by three independent tendons, the constraint equation should be modified to reflect the change on the routing of the cables: $\Delta = [\Delta_1, \Delta_2, \Delta_3]^T$ where Δ_i denotes the displacement of the tendon on the ith side of the modules.

$$\Delta_i \leq N \cdot b_{ini} - \sum_{j=1}^{N} b_i^j, \quad i = 1, 2, 3 \tag{4.11}$$

where b_{ini} is the initial length of the diagonal links and i corresponds to the i-th side of the manipulator.

Similarly, it is possible that infinite many sets of B may satisfy the same tendon constraint. We use the minimum potential energy method to choose the set of B from the infinite many possible Bs that minimizes the potential energy of the manipulator, E. The potential energy for each module is calculated using Eq. 2.10.

For origami manipulators comprising multiple modules connected in series, we also formulate the forward model as an optimization problem as follows.

$$\begin{aligned} \min_{B} \quad & E = \sum_{j=1}^{N} E^j \\ \text{s.t.} \quad & \text{Constraints (Eq. 4.11)} \end{aligned} \tag{4.12}$$

where E^j is the potential energy of each module in the manipulator.

The solution B of the optimization problem depends on both the stiffnesses of the VSJs S and the tendon displacement D, since S and D are included in the objective function and constraint equations in Eq. (4.12). With an optimal B, we can solve the forward problem to predict the final shape of the manipulator consisting of multiple modules connected in series, given D and S. The position of the end effector point of the manipulator can be readily calculated based on the shape of the manipulator.

In the context of reinforcement learning, the forward model serves as the state transition model $\mathcal{F}_\phi(s' \mid s, a)$. This model predicts the next state of the origami manipulator based on its current state and the action taken at this step (e.g., tendon displacement or changes in tendon displacement).

In this work, we develop a custom environment for the origami manipulator following the Gymnasium API. It implements two key methods: reset and step. The reset method initializes the manipulator to its initial state at the start of each episode. The step method takes an action sampled from the current policy, computes the next state using the forward model (state transition model), and returns the next state along with the corresponding reward and additional information.

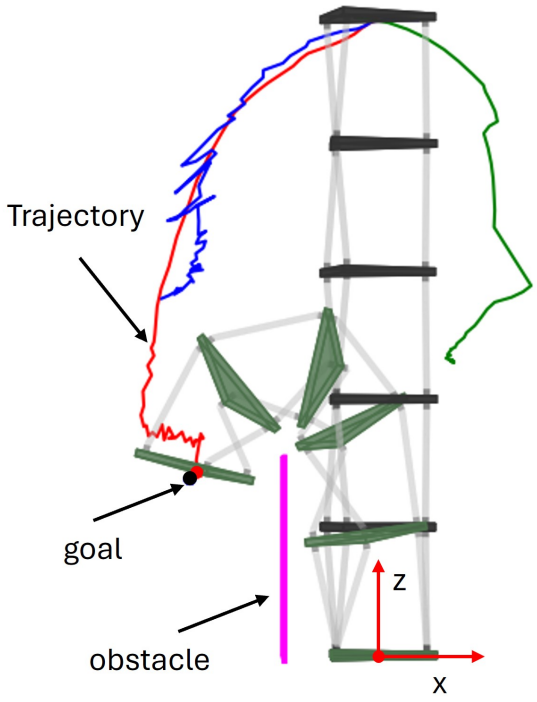


Figure 4.4: Evaluation of the trained policy for the reaching task with obstacle avoidance. The red curve shows the trajectory of the end-effector point. The red dot shows the position of the end-effector point at the last step.

4.2.2 Reaching a Goal Point for the Origami Manipulator

In this section, we describe the simulation setup and the corresponding results. The forward model in Section 4.2.1 is implemented as the state transition model in the learning process. As shown in Fig. 4.4, we increase the number of modules in the manipulator from 4 to 5 to achieve a more challenging reaching task with obstacle avoidance. The bottom plate of its bottommost module is fixed. We aim to optimize both the stiffnesses of all VSJs and the control strategy on the tendon displacements to make the end-effector point of the manipulator reach a goal point while avoiding collision with a surrounding obstacle.

Reaching Task with One Obstacle

The objective of this task is to train the RL agent to reach a predefined goal position at $[-50, 0, 50]^T$ while avoiding a planar obstacle. The obstacle is positioned parallel to the YZ plane at a fixed x-coordinate of -25 mm. The agent must navigate through the environment while adher-

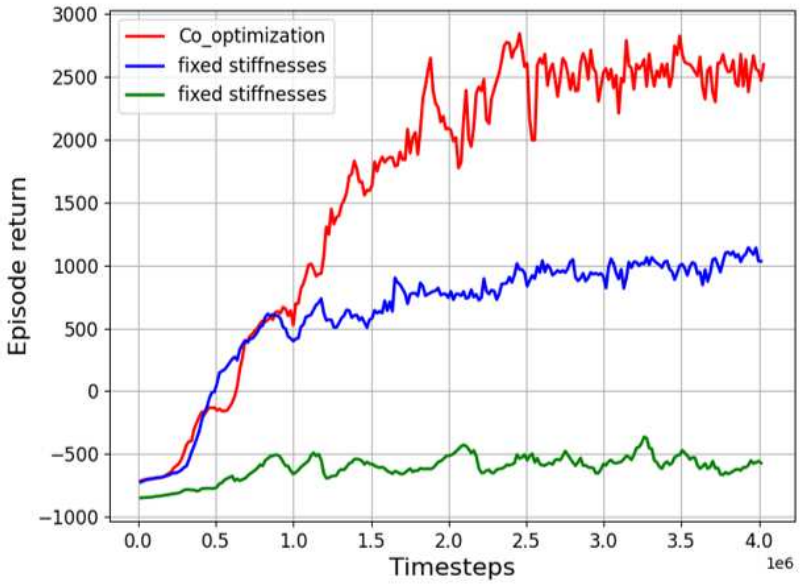


Figure 4.5: Average return of the training process with one obstacle for a total of 4 million time steps.

ing to kinematic constraints and ensuring collision-free movement. Since the obstacle lies between the manipulator's initial position and the goal, the manipulator must maneuver around the obstacle to successfully reach the target. To guide the agent toward the goal position, we employ a dense reward function defined as follows.

$$r = -r_c \mathbf{1}(\text{collision}) + r_s \mathbf{1}(d < d_s) + \frac{r_d}{0.2 + d} \tag{4.13}$$

where r_c is the collision penalty applied if the agent collides with the obstacle, r_s is the success bonus reward applied if the agent reaches the goal within threshold d_s of 3.0 mm. The indicator function $\mathbf{1}(\text{collision})$ returns 1 if a collision is detected and 0 otherwise. Similarly, the indicator function $\mathbf{1}(d < d_s)$ returns 1 if d is less than d_s, and 0 otherwise. $d = \|p_t - p_g\|$ is the distance between the effective tip p_t and the goal position p_g. The last term represents the distance reward scaled by a constant r_d. We extended the forward model to calculate the distance d and to detect the collision between the manipulator and the obstacle.

We employ the PPO algorithm with a multi-layer perceptron (MLP) policy, consisting of two hidden layers with 64 neurons each and ReLU activation. The observation space of the control policy consists of tendon displacements, stiffnesses of the joints, shape of the manipulator, and the

goal position. The actions are the change of tendon displacements. The forward model is used to develop a custom environment compatible using the Gymnasium API [80] for the RL process. Our co-optimization framework is implemented with the reinforcement learning library Stable-Baselines3 [81].

To balance exploration and exploitation, we employ a linear decay strategy for both the entropy coefficient and the learning rate. The entropy coefficient decreases linearly from 0.02 to 0.001, while the learning rate gradually declines from 0.00025 to 0. We apply normalization to standardize both observations and rewards to help mitigate large fluctuations and improves learning stability. We present the average episode return as a function of time steps in Fig. 4.5.

The stiffnesses of the VSJs are optimized through the training process to be S = [2.50, 2.41, 1.87, 0.58, 2.12, 1.52, 2.50, 0.40, 0.40, 1.26, 0.40, 1.09, 0.82, 1.55, 1.25]^T. To evaluate the trained control policy and stiffness values, we deploy them on a simulated manipulator and visualize the results in Fig. 4.4. The initial shape of the manipulator is depicted in black and white. The trajectory of the effective tip (shown as the red curve) and the final shape of the manipulator, shown in green, show the agent's ability to successfully reach the goal (the black dot) while effectively avoiding the surrounding obstacle. We also train control policies for the same reaching task with obstacle avoidance without the co-optimization procedure. In this case, the stiffness values are predetermined as stiffness sets S_1 and S_4 and remain fixed throughout the learning process. The trained control policies are then deployed on the manipulator with these specified stiffnesses, and the resulting trajectories are shown as the blue and green curves, respectively. For stiffness set S_1, the agent navigates toward the goal but fails to reach it completely. In contrast, for stiffness S_4, the agent becomes trapped in a local minimum, avoiding both the goal position and the obstacle.

Reaching Task with Two Obstacles

We increase the task difficulty by introducing a second obstacle (obstacle 2 in Fig. 4.6), which is parallel to the YZ plane and positioned at a fixed x-coordinate of -60 mm. This obstacle extends from a z-coordinate of 135 mm (bottom edge) to 200 mm (top edge).

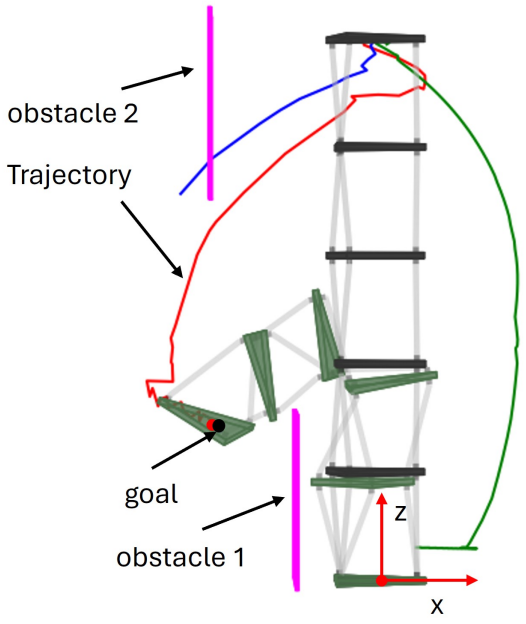


Figure 4.6: Execution of the trained policy for the reaching task while avoiding two obstacles. The red curve shows the trajectory of the effective tip. The red dot shows the position of the effective tip at the last step.

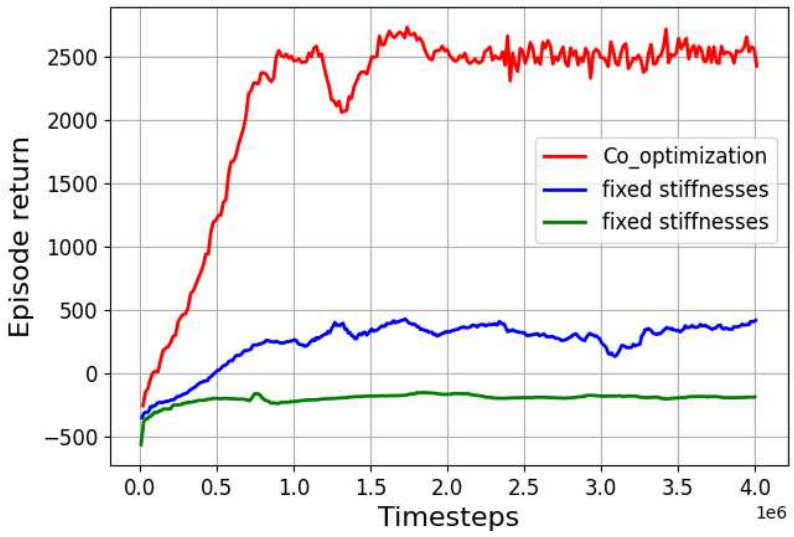


Figure 4.7: Average return of the training process with two obstacles for a total of 2 million timesteps

As shown in Fig. 4.4, the previously trained policy leads the agent to collide with this second obstacle during its motion. Unlike the first obstacle, which primarily constrains the final stage of the motion, the second obstacle introduces constraints early in the trajectory, requiring the agent to plan its movement from the very beginning.

Now we include the second obstacle in the custom environment and train a new policy to make the agent reach the same goal point while avoiding both obstacles. The reward function and hyperparameters are the same as before, except that we increase the initial value of the entropy coefficient to 0.03 to encourage greater exploration.

We present the average episode return as a function of time steps in Fig. 4.7. The stiffnesses of the VSJs are optimized through the training process to be S = [2.37, 1.70, 2.50, 1.86, 2.04, 2.11, 0.49, 0.82, 1.98, 0.50, 2.50, 2.31, 0.69, 2.50, 0.90]^T. We also deploy the learned control policy and stiffness values on a simulated manipulator and visualize the results in Fig. 4.6. The resulting trajectory of the effective tip (shown as the red curve) demonstrates that the agent successfully reaches the goal (depicted as the black dot) while effectively avoiding both obstacles. Notably, we observe that the manipulator initially bends slightly to the right before redirecting its motion toward the left to reach the goal. This initial rightward movement allows the manipulator to navigate around obstacle 2 before proceeding toward the target, illustrating a strategic adjustment that ensures collision-free motion. For comparison, we also train control policies for the same reaching task with both obstacle avoidance without the co-optimization procedure. As in the previous case, the stiffness values are predetermined to be S_1 and S_4 and remain fixed throughout the learning process. The resulting trajectories are shown as the blue and green curves, respectively in Fig. 4.7. For stiffness set S_1, the agent navigates toward the goal but fails to avoid obstacle 2. For stiffness set S_4, the agent again get trapped in a local minimum, avoiding both the goal position and the planar obstacle.

4.3 Co-optimization for Origami Robots based on physics engine

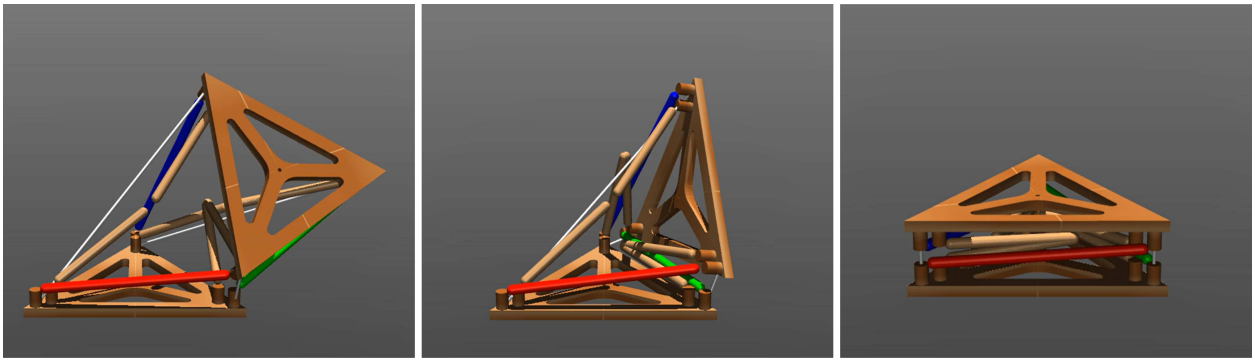
In the previous section, the robot learning framework was implemented based on the forward kinematics of the origami manipulator, which was formulated using the minimum potential energy method. However, this approach has certain limitations. It neglects interactions with the environment, such as reaction forces from the ground or external objects, collisions with obstacles, and the effects of gravity acting on the manipulator itself or its payload. Furthermore, because the forward kinematics is solved through numerical optimization, it is computationally expensive and thus inefficient for reinforcement learning.

To incorporate interactions with the environment, we employ the physics engine MuJoCo [82] to simulate the dynamics of the reconfigurable robots. MuJoCo is a real-time physics engine capable of fast and accurate simulation of articulated structures and their interactions with the environment.

In the MuJoCo physics engine, the actuation tendons of the robot are modeled as spatial tendons, while the compliant joints connecting the three vertical and three diagonal links to the bottom plate are simplified as ball joints. The top plate of each origami module is connected to the six links through six spherical joints. Because MuJoCo supports only chain-like kinematic structures, the joint connecting the first vertical link to the top plate is explicitly defined as a ball joint in the model, whereas the remaining five joints are enforced through equality constraints. The simulation timestep is set to 0.01 s, and actuation is realized by position actuators attached to the three tendons.

Similarly, we develop a custom environment file following the Gymnasium API. Now the custom environment is based on the physics engine Mujoco, instead of forward kinematics. For each step forward, the environment file calls Mujoco to calculate the next state of the origami manipulator, based on the current state and action. We use a control frequency of 10 Hz by executing 10 MuJoCo time steps per action, ensuring smooth physical motion.

We note that the position actuators attached to spatial tendons in MuJoCo are implemented as bidirectional elastic springs that generate forces in both tension and compression. This differs from the physical hardware, where actuation tendons (e.g., fishing lines) can only be pulled. To bridge this reality gap, we enforce a tension-only constraint in the step method of the custom environment file by tracking desired tendon lengths separately from actual tendon lengths. The desired tendon length, representing the motor's commanded position, can change bidirectionally. At each timestep, we compare the desired length with the current actual length for each tendon. When the desired length is less than the current length, the tendon would be taut and the position actuator applies tension force as normal. When the desired length is greater than or equal to the current length, indicating the motor is attempting to push the tendon, we disable that actuator using MuJoCo's actuator disabling mechanism, causing it to generate zero force and simulating a slack tendon with zero stiffness.



(a) slanting (b) bending (c) collapsing

Figure 4.8: An origami module modeled with Mujoco. a) Slanting motion with one tendon actuated. b) Bending motion with two tendons actuated. c) Collapsing motion with three tendons actuated.

A single module simulated in MuJoCo is shown in Fig. 4.8, where the tendons are rendered in white. The bottom plate of origami module is fixed. All three joints are assigned the same stiffness value of 1. When the position actuator of the first tendon is activated, the module slants. When the actuators of both the first and second tendons are activated, the module bends. Activating all three

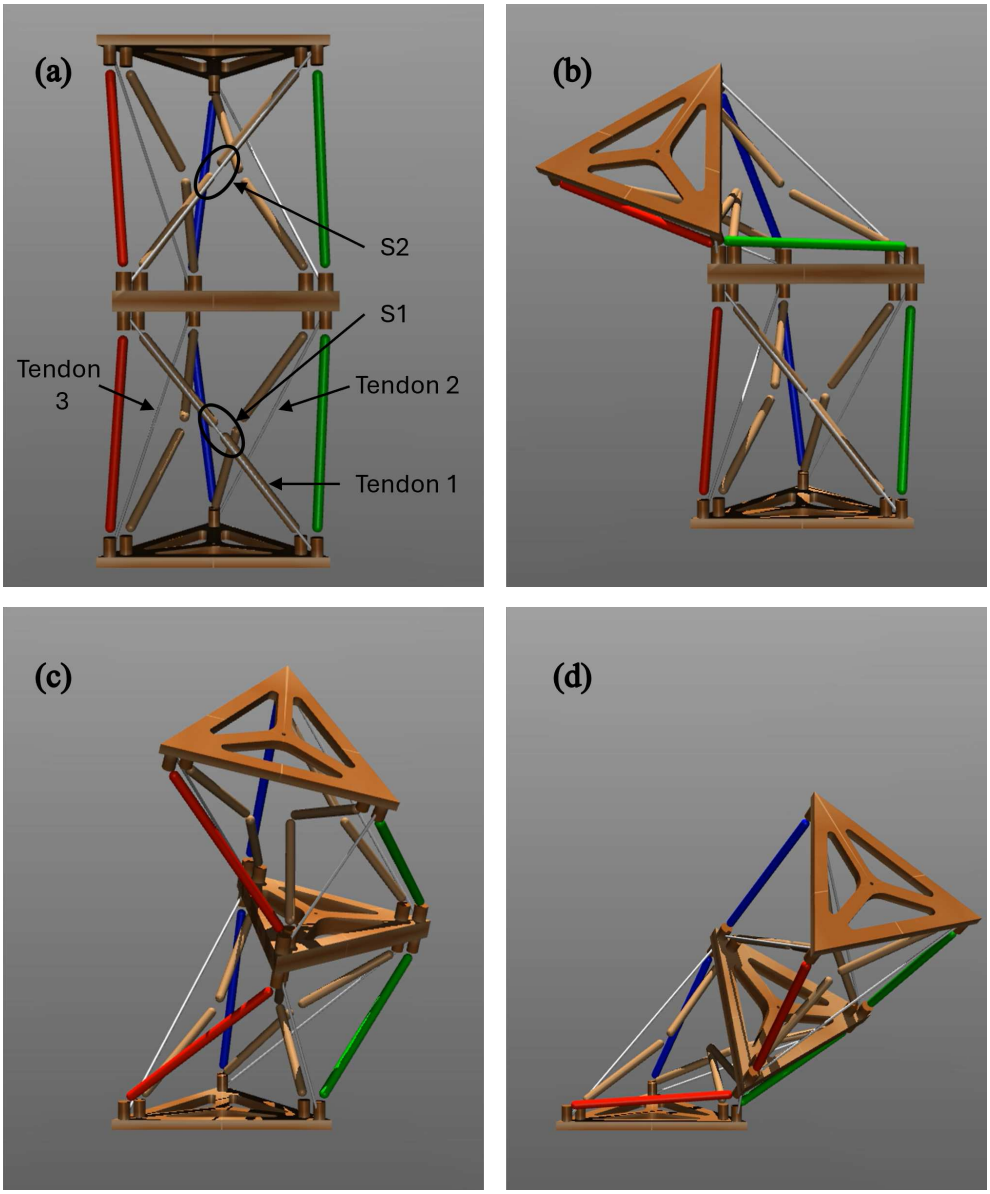


Figure 4.9: An origami manipulator consisting of two modules connected in series. By adjusting the stiffnesses of the two joints along the same tendon, we can achieve different motions upon tendon actuation. a) Initial straight shape. b) Only top module slanted. c) Both modules equally slanted. d) Only bottom plate slanted.

actuators causes the module to collapse. Unlike kinematic models, MuJoCo naturally accounts for contact forces and collisions among all module components.

We can easily connect multiple modules in series in Mujoco, by "welding" the bottom plate of the next module to the top plate of the previous module. As discussed in Chapter 2, for an origami manipulator composed of multiple modules connected in series, the stiffness distribution of the joints significantly influences the motion and overall shape of the manipulator under tendon actuation. This behavior can also be simulated in MuJoCo. Figure 4.9a shows the initial straight configuration of a two-module manipulator. The stiffnesses of the first joints of the bottom and top modules are denoted as S_1 and S_2, respectively. When $S_1 = 2.5$ N·m/rad and $S_2 = 0.5$ N·m/rad, as shown in Fig. 4.9b, actuating the first tendon causes the slanting motion to concentrate on the top module. Conversely, when $S_1 = 0.5$ N·m/rad and $S_2 = 2.5$ N·m/rad, as shown in Fig. 4.9d, the slanting motion concentrate on the bottom module. Additional, if $S1 = S2 = 1$ N·m/rad, the slanting motions of the two modules become approximately equal.

We apply the co-optimization procedure to achieve a reaching task, wherein the objective is to move the end-effector point of the two-module origami manipulator to a specified goal point. The co-optimization procedure follows the framework described in Section 4.2. The observation space comprises the desired tendon lengths, current end-effector position, goal position, and orientations of each module's top plate. The action space consists of changes on desired tendon lengths, constrained to the range of -1 mm to 1 mm, represented as a 3-by-1 column vector. The stiffness values of the diagonal joints are optimized concurrently through the reinforcement learning process according to Eq. 4.4.

We design a dense reward function that combines multiple components to guide the policy toward precise reaching while ensuring stable convergence. The reward at timDestep t is computed as:

$$r_t = r_t^{\text{dist}} + r_t^{\text{prox}} + r_t^{\text{vel}} + r_t^{\text{success}} - c_{\text{time}} \tag{4.14}$$

where each component serves a distinct purpose in shaping the learning behavior. The distance-based reward r_t^{dist} encourages progress toward the goal:

$$r_t^{\text{dist}} = \lambda_{\text{dist}}(d_{t-1} - d_t) \tag{4.15}$$

with $d_t = \|p_t^{\text{ee}} - p^{\text{goal}}\|_2$ representing the Euclidean distance between the end-effector position p_t^{ee} and the goal position p^{goal}, and $\lambda_{\text{dist}} = 15.0$ as the scaling factor. This term provides dense feedback by rewarding any reduction in distance.

To further guide the policy in close proximity to the goal, we incorporate a proximity reward:

$$r_t^{\text{prox}} = \frac{\lambda_{\text{prox}}}{1 + 1000 \cdot d_t} \tag{4.16}$$

where $\lambda_{\text{prox}} = 2.0$. This inversely proportional term provides increasingly strong positive feedback as the end-effector approaches the goal, helping to overcome the diminishing returns of r_t^{dist} in the final approach phase.

To promote stability upon reaching the goal, we introduce a velocity penalty within a braking zone:

$$r_t^{\text{vel}} = \begin{cases} -\lambda_{\text{vel}}\|v_t^{\text{ee}}\|_2 & \text{if } d_t < d_{\text{brake}} \\ 0 & \text{otherwise} \end{cases} \tag{4.17}$$

where v_t^{ee} denotes the end-effector linear velocity, $\lambda_{\text{vel}} = 3.0$, and $d_{\text{brake}} = 0.01$ m defines the braking zone threshold. This component discourages high-speed approaches near the target, ensuring smooth and controlled final positioning.

Success is defined when the end-effector point satisfies both positional and velocity criteria simultaneously: $d_t < d_{\text{success}} = 0.002$ m and $\|v_t^{\text{ee}}\|_2 < 0.001$ m/s. Upon meeting these conditions, the agent receives a substantial success bonus $r_t^{\text{success}} = 400$ and the episode terminates. Finally, we impose a constant time penalty $c_{\text{time}} = 0.1$ per timestep to encourage efficient task completion. This

relatively small penalty ensures that episodes do not terminate prematurely while still incentivizing faster convergence to the goal.

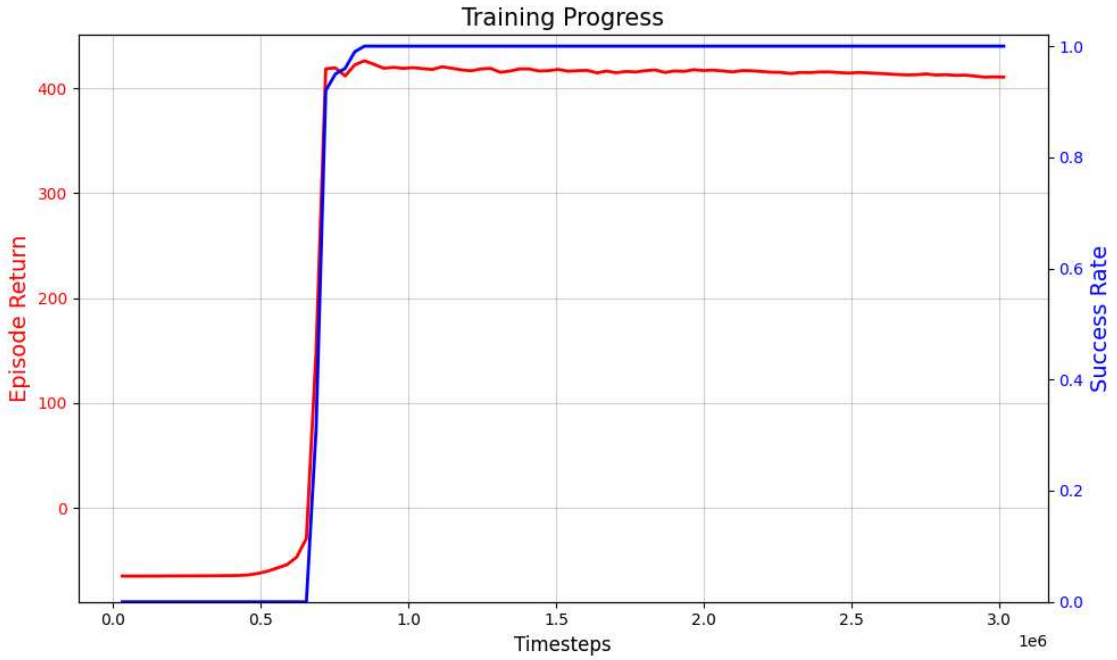


Figure 4.10: Training curves for the reaching task for origami manipulator made of two modules. The episode return (blue, left axis) and success rate (red, right axis) are shown as functions of training timesteps. The agent achieves near-perfect success rate after 1 million timesteps.

Figure 4.11 presents the evolution of episode return and success rate over 3 million training timesteps. The episode return shows a rapid increase around 0.7 million timesteps, followed by the success rate rising sharply to 100% at approximately 0.8 million timesteps. Notably, after 1 million timesteps, the episode return exhibits a gradual decline, which corresponds to a reduction in average episode length from approximately 127 to 118 steps. This trend indicates that the agent is learning to reach the goal more efficiently, completing tasks in fewer steps while maintaining perfect success.

After assigning the learned stiffness values to the origami manipulator modeled in Mujoco, we evaluate the learning result through an evaluation episode. Fig 4.11 presents a sequence of snapshots illustrating the manipulator's motion during this evaluation episode. Fig 4.11a depicts the initial configuration with the goal position rendered as a white sphere. At each timestep, the

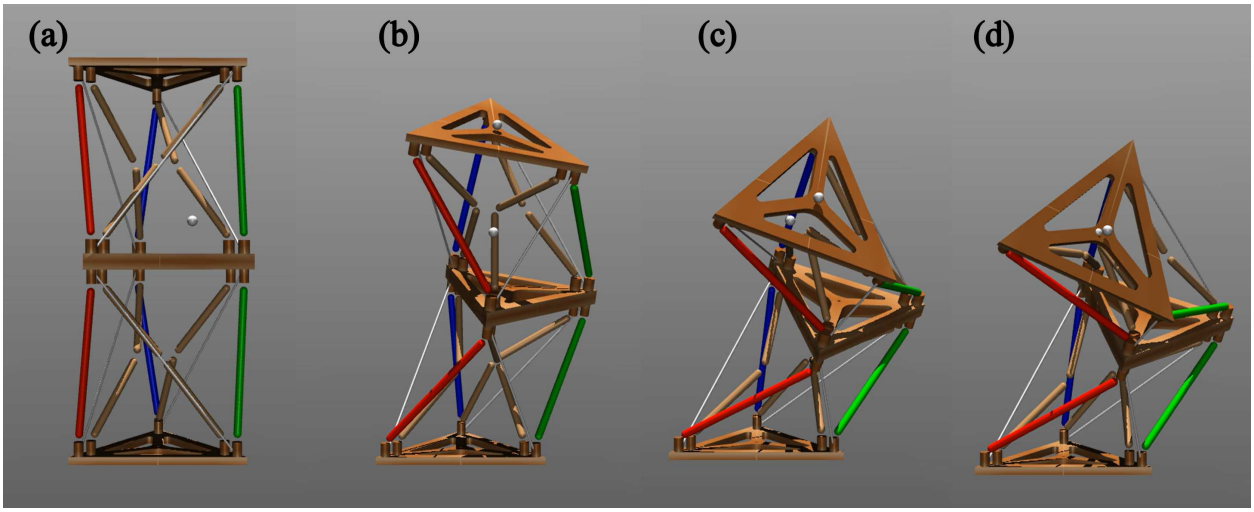


Figure 4.11: Evaluation in the physics engine. After the co-optimization process, the learned joint stiffness values are incorporated into the physics engine, and tendon actuations are applied according to the learned control policy.

current state is extracted from MuJoCo and passed to the learned policy, which outputs the corresponding action. This action is then applied to the manipulator in the simulation. This closed-loop process continues until task completion. As shown in Fig 4.11d, the end-effector point successfully reaches the goal within a threshold distance of 2 mm. Figs 4.11b and c illustrate intermediate configurations during the reaching motion.

We also train a goal-conditioned policy by randomizing the goal position at the start of each episode. The goal point is sampled uniformly within a sphere of radius 8 mm centered at the nominal target position. Under this configuration, the learned policy can ideally reach any point within this spherical region. Training proceeds for 15 million timesteps, with the success rate shown in Fig. 4.12. The success rate plateaus at approximately 40% after 4 million timesteps, indicating that different stiffness configurations are required to effectively reach targets in different locations.

Figure 4.12: Success rate as a function of training timesteps for the goal-conditioned policy. Goal positions are randomly sampled within a sphere of radius 8 mm centered at the nominal target position.

4.3.1 Deployment on a Prototype

To validate the effectiveness of the learning result, we develop a prototype (as shown in Fig. 4.13) to deploy the learned stiffness values and control policy. The task is the same as in the simulation, making the end-effector point of the manipulator reach the specified goal point.

The bottom and top plates of each module are co-printed with the spherical joints—implemented as thermoplastic polyurethane (TPU) tubes—as an integrated structure through multi-material 3D printing (Original Prusa XL printer). The plates and spherical joints are fabricated using PETG and TPU filaments, respectively. Each vertical link is implemented a carbon fiber rod measuring 55 mm in length and 2 mm in diameter. Each diagonal link consists of two rigid half-links and a flexible bending joint, which are also co-printed as a single integrated structure. The rigid sections are printed with PETG filament, while the bending joint is made of TPU filament. At each corner of the triangular plate, a guiding hole is provided for routing the actuation tendons. A polytetrafluoroethylene (PTFE) tube is inserted into each guiding hole to significantly reduce friction between

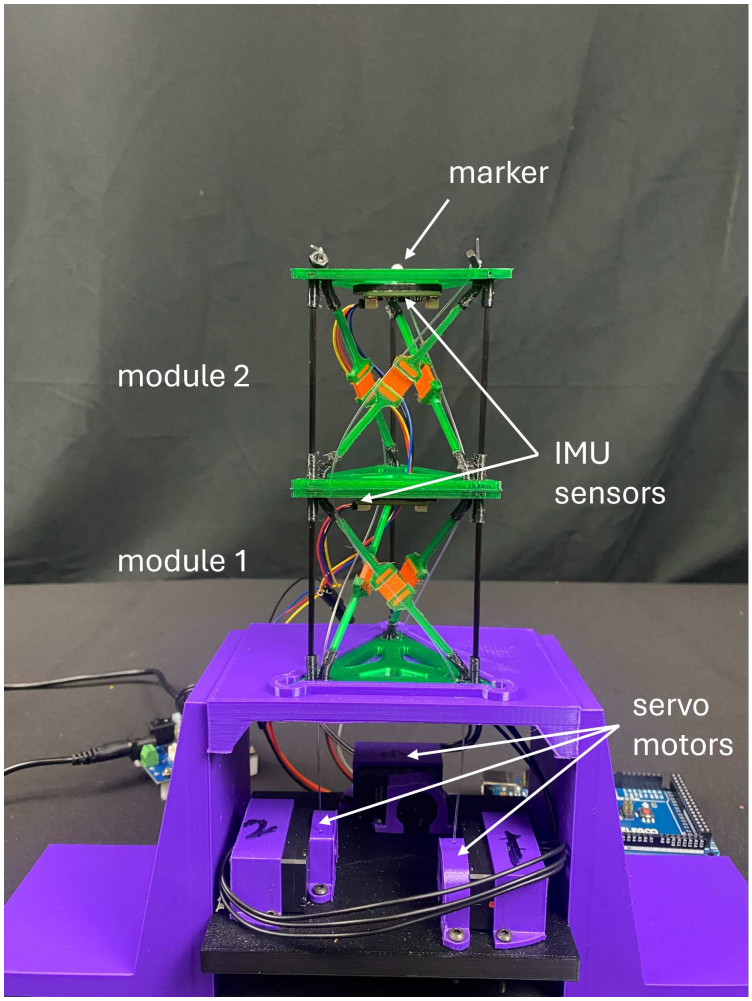


Figure 4.13: A prototype manipulator comprising two modules.

the actuation tendon and the hole surface. For ease of assembly, double-sided adhesive tape is used to attach the top plate of one module to the bottom plate of the next, thereby allowing multiple modules to be stacked in series. The bottom plate of the lowest module is fixed to a fixture surface. On each side of the origami manipulator, a tendon (Monofilament Fishing Line, 0.66 mm; Reaction Tackle) is routed along the diagonal links through the corresponding guiding holes in a zigzag pattern. The lower ends of the tendons pass through the guiding holes in the fixture surface and are attached to the winches of three servo motors (DYNAMIXEL XL330-M288-T; Robotis).

We use inertial measurement unit (IMU) sensors (BNO055; Adafruit) to detect the absolute orientation of the top plate of each module. The readings from the IMU sensors are internally fused using the sensor's built-in nine-degree-of-freedom (9-DoF) sensor fusion algorithm and converted into Euler rotation angles (roll, pitch, and yaw) through the onboard processor. The resulting orientation data are incorporated into the observation space and provided to the control policy for decision making. In addition, a tracking marker is placed at the end-effector point on the top plate of the uppermost module, and a motion-tracking camera (V120:Trio; OptiTrack) is used to record its real-time position. The measured position is likewise incorporated into the observation space and supplied to the control policy.

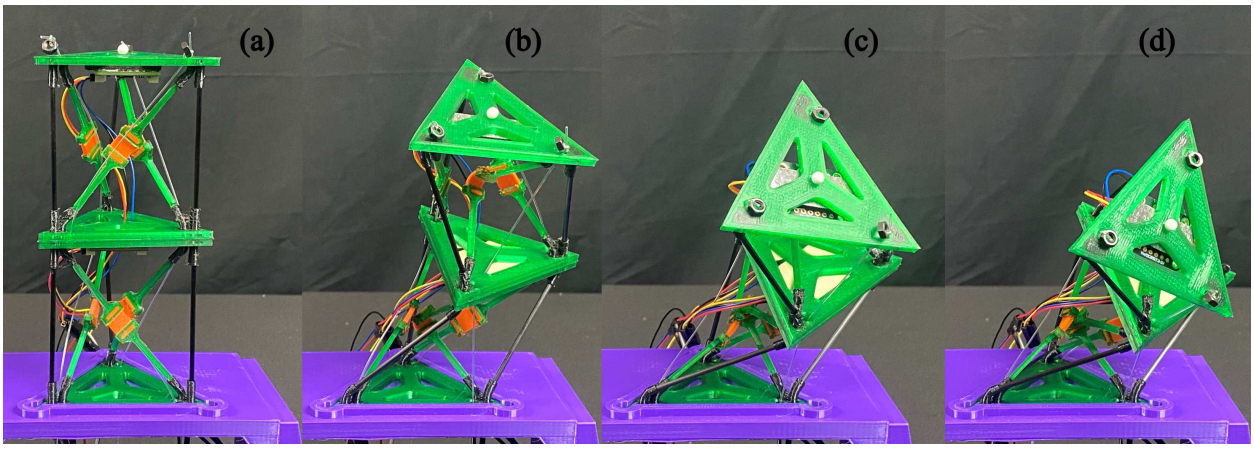


Figure 4.14: Deployment of the learned stiffness values and control policy on a prototype manipulator made of two modules.

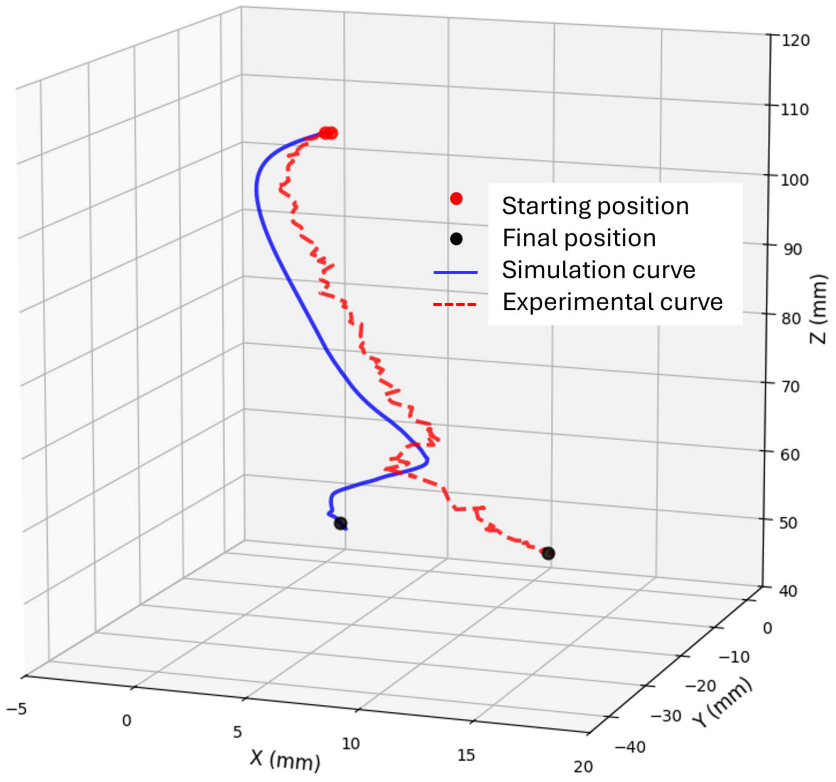


Figure 4.15: End-effector trajectories during a reaching task in MuJoCo simulation (blue solid line) and experimental deployment (red dashed line).

First, we apply the learned stiffness values S = [0.81, 0.80, 2.39, 0.94, 1.27, 2.12]^T N·m/rad to the six diagonal joints by fabricating TPU bending joints in the diagonal links with specific thicknesses and widths to achieve the target stiffness. We then execute the learned control policy on the two-module prototype at 10 Hz for the reaching task. Fig. 4.14 shows an image sequence of the prototype during task execution. The observed motion trends resemble those in the MuJoCo simulation. Fig. 4.15 compares the end-effector trajectories between Mujoco simulation (blue solid line) and prototype experiment (red dashed line), showing similar overall motion patterns despite some deviations. For deployment on the prototype, the final distance between the end effector and the goal point stabilizes at 14.51 mm. Despite the sim-to-real gap, the prototype effectively reproduces the qualitative motion patterns observed in simulation, demonstrating that the learned control strategy successfully adapts to the physical system. We attribute the positioning error to two primary sources of model discrepancy. First, the compliant TPU joints on the plates are approximated as ideal spherical joints in MuJoCo, which neglects their complex geometries and nonlinear stiffness characteristics. Second, the simulation does not account for friction forces between the tendons and the guide holes during actuation, which affects the motion of the manipulator. These unmodeled dynamics contribute to the deviation between simulated and experimental performance, though the overall motion trajectory remains qualitatively consistent.

4.3.2 Payload and More Modules

MuJoCo provides the capability to model interactions between the manipulator and the external environment, including the effect of external loads. To demonstrate this, we apply the co-optimization process to a reaching task for an origami manipulator with a payload mounted on its top plate.

As shown in Fig. 4.17a, a 0.2 kg cylindrical payload (rendered in black) with a radius of 15 mm and a height of 3 mm is mounted on the top plate of the uppermost module. The end-effector and goal point are visualized as a red and a white sphere, respectively. The co-optimization proce-

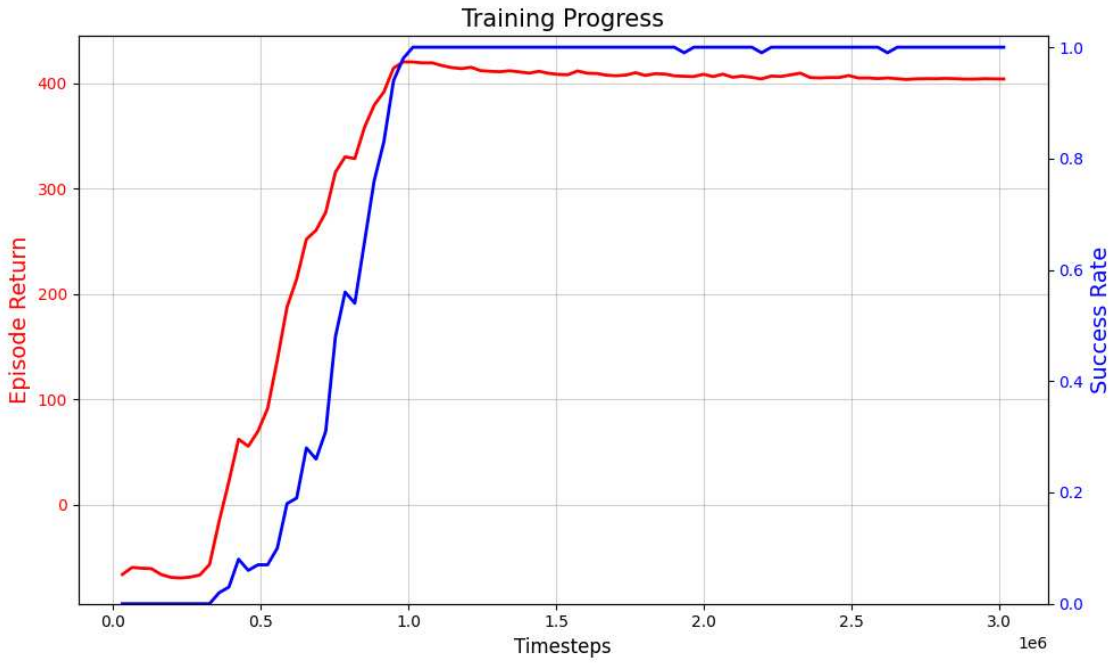


Figure 4.16: Training curves for the reaching task of the origami manipulator with a payload mounted on the top plate. The episode return (blue, left axis) and success rate (red, right axis) are shown as functions of training timesteps. The agent achieves near-perfect success rate after 1 million timesteps.

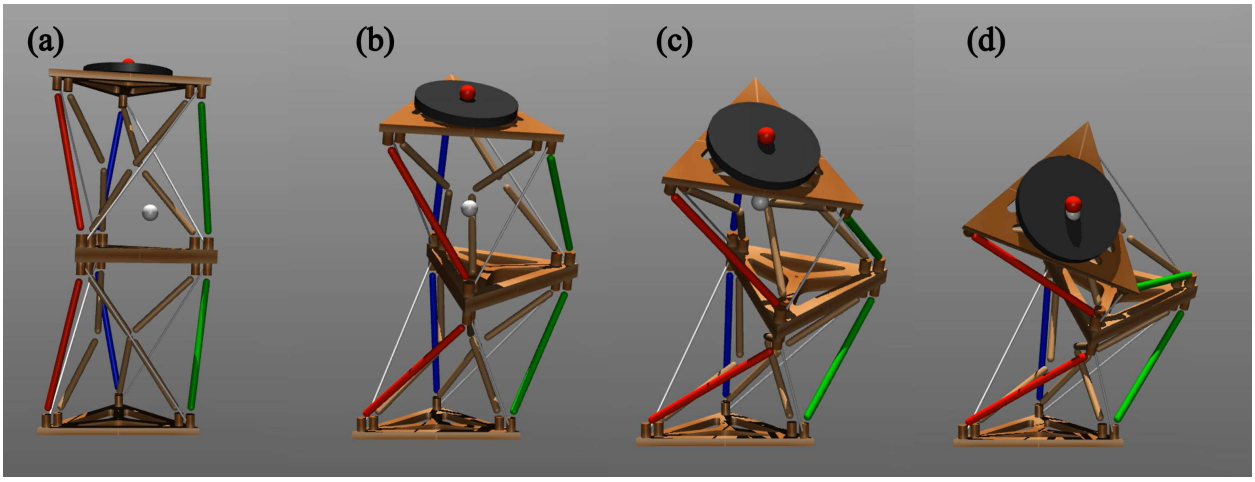


Figure 4.17: Evaluation in the physics engine for manipulator with payload of 200 grams mounted on top plate. After the co-optimization process, the learned joint stiffness values are incorporated into the physics engine, and tendon actuations are applied according to the learned control policy.

dure and hyperparameters are identical to those described in Section 4.3, except that the origami manipulator is initially slightly deformed by the payload.

The learning process is carried out for 3 million timesteps. The corresponding training curves for the reaching task of the origami manipulator with the mounted payload are shown in Fig. 4.16. The episode return (red, left axis) and success rate (blue, right axis) are plotted as functions of training timesteps. The agent achieves a near-perfect success rate after approximately 1 million timesteps. Notably, after 2 million timesteps, the episode return exhibits a slow and gradual decline, which corresponds to a reduction in average episode length. This trend indicates that the agent is learning to reach the goal more efficiently, completing tasks in fewer steps while maintaining perfect success.

After assigning the learned stiffness values to the origami manipulator modeled in MuJoCo, we evaluate its performance through a representative reaching episode. Fig. 4.17 presents a sequence of snapshots illustrating the manipulator's motion. The initial configuration is shown in Fig. 4.17a. Figs. 4.17b and c depict intermediate configurations during the reaching motion, while Fig. 4.17d shows that the end effector point successfully reaches the target, achieving a final position error of 1.46 mm, which is within the 3 mm threshold.

More complex shapes and motions can be achieved by connecting multiple modules in series. As described in Section 2.4, we develop a quadruped robot whose legs each consist of four origami modules connected in series. In this implementation, the mechanical design parameters—specifically, the stiffness values of the diagonal joints within each module—are manually selected based on engineering intuition and preliminary testing. Similarly, the tendon control algorithms are developed through empirical trial-and-error. While this manual design approach successfully demonstrates the robot's locomotion capabilities across four different modes, it is inherently time-consuming and does not guarantee optimal performance, as the interactions between structural parameters and control policies are not systematically explored. The co-optimization framework based on the MuJoCo physics engine can potentially be extended to optimize both the mechanical design (stiffness values of diagonal joints) and control policy simultaneously for mul-

timodal locomotion. This approach could enable high-performance locomotion behaviors beyond the four locomotion modes demonstrated in Section 2.4.

Here, as a preliminary demonstration, we apply the co-optimization framework to a single robot leg composed of four serially connected modules, rather than optimizing the entire quadruped locomotion robot. We employ the same learning hyperparameters and reward functions as described in Section 4.3, with one modification: the success distance threshold is increased from 2 mm to 3 mm to reduce task difficulty and accelerate convergence.

Figure 4.18: Training curves for the reaching task of the origami manipulator made of four modules. The episode return (blue, left axis) and success rate (red, right axis) are shown as functions of training timesteps. The agent achieves near-perfect success rate after 2.5 million timesteps.

Figure 4.18 presents the evolution of episode return and success rate over 3 million training timesteps. The episode return shows a steady increase to approximately 420 at around 2 million timesteps, followed by stabilization near the optimal return. The success rate increases to approximately 100% in the final phase of training. Both the episode return and success rate exhibit slower growth compared to the two-module manipulator described in Section 4.3. This is expected, as the initial distance between the end-effector and goal point is substantially larger in the four-module

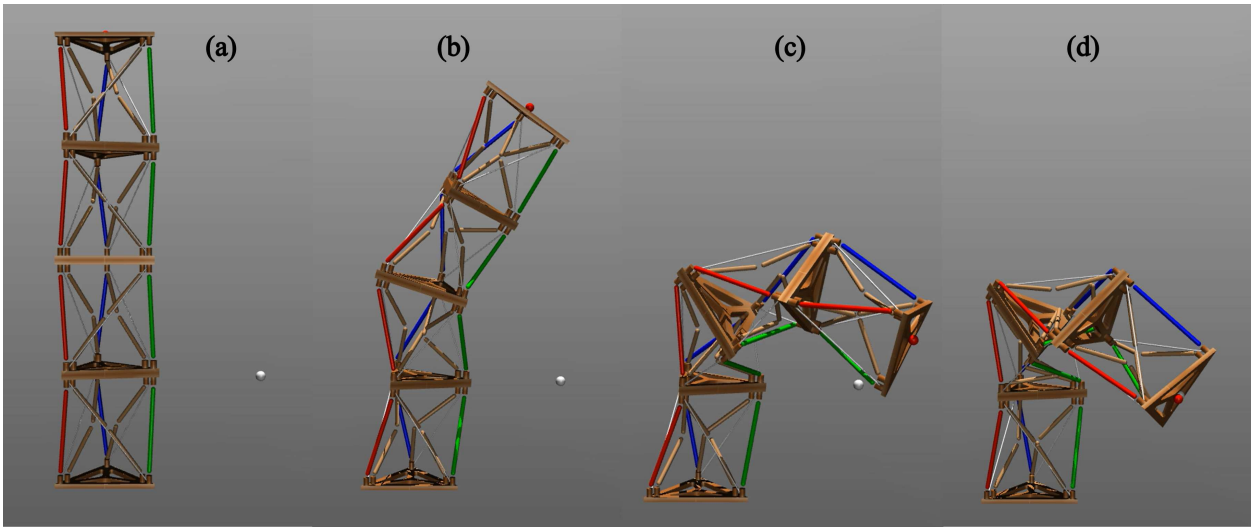


Figure 4.19: Evaluation in the physics engine for manipulator made of four modules. After the co-optimization process, the learned joint stiffness values are incorporated into the physics engine, and tendon actuations are applied according to the learned control policy.

manipulator. Consequently, the agent exhibits a slower learning rate and requires more timesteps to converge to an effective policy.

After assigning the learned stiffness values to the four-module origami manipulator modeled in Mujoco, we evaluate the learning result through an evaluation episode. Fig 4.19 presents a sequence of snapshots illustrating the manipulator's motion during this evaluation episode. Fig 4.19a depicts the initial configuration with the goal position rendered as a white sphere. As shown in Fig 4.19d, the end-effector point successfully reaches the goal within a threshold distance of 3 mm. Figs 4.19b and c illustrate intermediate configurations during the reaching motion.

4.4 Chapter Summary

This chapter presents a framework for co-optimizing mechanical design parameters and control policies of modular reconfigurable robots using reinforcement learning. We formulate the problem where reconfiguring parameters (joint stiffnesses) and control policies (tendon displacements) are simultaneously optimized to maximize task performance.

Initially, we develop a forward kinematics model based on the minimum potential energy method as the state transition model in a custom Gymnasium environment. Applied to reach-

ing tasks with obstacle avoidance using a five-module manipulator, the results demonstrate that fixed stiffness configurations often lead to failure, while co-optimized stiffness values and control policies successfully accomplish the tasks, proving the necessity of simultaneous optimization.

To incorporate environmental interactions such as contact forces and gravity, we employ the MuJoCo physics engine with physics-based custom environments. We enforce tension-only constraints on actuation tendons to accurately represent the physical hardware. Through Proximal Policy Optimization training, we achieve near-perfect success rates for reaching tasks with two- and four-module manipulators, including scenarios with payloads. The learned parameters are validated on a physical two-module prototype, demonstrating effective sim-to-real transfer.

This work establishes a comprehensive data-driven framework for co-optimizing mechanical design and control of modular reconfigurable robots, offering a systematic alternative to manual tuning and enabling extension to more complex tasks such as multi-modal locomotion.

Chapter 5

Conclusion and Future Work

Reconfigurable robots that exhibit multiple modes of locomotion, multifunctionality, and shape-morphing capabilities are increasingly essential to meet the diverse and evolving demands of modern life. Reconfigurability in existing robotic systems is generally realized by integrating multiple mechanisms into a single robot. These reconfigurable robots are often cumbersome or challenging to control and actuate, and their adaptability is also limited after fabrication.

We propose realizing reconfigurable robots by stacking multiple reconfigurable modules in series and using a few tendons for actuation. The shape and motion of these reconfigurable modules under tendon actuation can be adjusted. These modular reconfigurable robots offer advantages such as ease of fabrication, extensive workspace, predictable behavior, and simplified control.

First, we present the origami-module-based reconfigurable robots. The origami module, inspired by the Kresling origami pattern, is equipped with joints that can independently transition between soft and rigid states, enabling the module to adapt its behavior under tendon actuation. We demonstrate the different shapes and motions that can be achieved by the origami module. A forward model is developed and validated to predict the motion of the module when actuated. We develop a reconfigurable robot with four legs, each composed of four serially connected modules. The robot can walk, crawl, and inch using the same mechanical structure.

We also investigate reconfigurable robots based on bistable modules. We first present methods to tune the energy landscape (EL) for a beam-based bistable module with elastic instabilities on the fly, without manual alterations. The two tuning strategies we developed—adjusting the beam's initial bending angle and varying the offset of the spring's extension—have proven effective in manipulating the EL, as evidenced by our forward model and its subsequent experimental validation. By connecting multiple bistable modules in series, we can realize a reconfigurable robot with different resting shapes, energy barriers, or transition sequences. We have also successfully solved the inverse problem to obtain desired resting angles or energy barriers through the two tuning strate-

gies. We further illustrate the practical applications of tuning through the successful deployment of the tunable module in three different scenarios: as a kicker to impart different energies to an object, as a reconfigurable arm that can reconfigure its resting shape, and as a crawling robot that can crawl in both directions using the same actuation.

In the last chapter, we employ reinforcement learning to realize co-optimization of design and control for reconfigurable robots consisting of multiple origami modules connected in series, where both the reconfiguring parameters (stiffnesses of compliant joints) and control policies (tendon displacements) are simultaneously optimized. Initially, we develop a forward kinematics model based on the minimum potential energy method as the state transition model in a custom Gymnasium environment, successfully training control policies for reaching tasks with obstacle avoidance and demonstrating the necessity of co-optimizing stiffness values alongside control strategies. However, this kinematic model neglects environmental interactions such as contact forces and gravitational effects. We therefore employ the MuJoCo physics engine for more realistic simulation, developing physics-based custom environments with tension-only constraints on the actuation tendons to accurately represent the physical hardware, where fishing lines can only be pulled. Through extensive training using Proximal Policy Optimization, we achieve near-perfect success rates for reaching tasks with manipulators composed of two and four modules, including scenarios with payloads mounted on the top plate. The learned stiffness values and control policies are successfully validated through deployment on a physical two-module prototype, demonstrating effective sim-to-real transfer. This work establishes a comprehensive framework for co-optimizing both the mechanical design and control of modular reconfigurable robots, paving the way for automated design and control of more complex multifunctional robotic systems.

For future work, we will focus on the following two directions:

First, we will apply the co-optimization framework to reconfigurable robots based on bistable modules. In this case, the design parameters would be the two tuning parameters: the flexible beam's initial bending angle and the spring's offset. These tuning parameters influence the resting shapes and energy barriers of the bistable modules, leading to different static shapes and dynamic

behaviors when actuated by the pulling tendons. The control policy would determine the actions on the tendon displacements. By employing the co-optimization framework, we can simultaneously optimize both the design parameters and control policy to accomplish specific shape-morphing or motion tasks.

Second, we plan to extend the co-optimization framework to the complete quadruped locomotion robot, where each of the four legs is implemented as a four-module origami manipulator. The task would focus on achieving faster locomotion speeds and discovering more robust and energy-efficient gaits through simultaneous optimization of joint stiffness distributions across all modules and coordinated tendon-actuation patterns. This represents a significantly more complex optimization problem compared to single-leg reaching tasks. Eventually, we envision incorporating an onboard camera mounted on the robot's body, using visual observations (image pixels) as part of the observation space to enable the robot to perceive its environment and autonomously transition between different locomotion modes—such as walking, crawling, and inching—based on terrain conditions or task requirements. This vision-based approach would move toward fully autonomous multi-modal locomotion without relying on predefined mode-switching rules.

Bibliography

[1] Ratan Othayoth, George Thoms, and Chen Li. An energy landscape approach to locomotor transitions in complex 3d terrain. *Proceedings of the National Academy of Sciences*, 117(26):14987–14995, 2020.

[2] Germán Sumbre, Graziano Fiorito, Tamar Flash, and Binyamin Hochner. Octopuses use a human-like strategy to control precise point-to-point arm movements. *Current Biology*, 16(8):767–772, 2006.

[3] Stefano Mintchev and Dario Floreano. Adaptive morphology: A design principle for multimodal and multifunctional robots. *IEEE Robotics & Automation Magazine*, 23(3):42–54, 2016.

[4] Kenji Misu, Akira Yoshii, and Hiromi Mochiyama. A compact wheeled robot that can jump while rolling. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7507–7512. IEEE, 2018.

[5] Jianguo Zhao, Weihan Yan, Ning Xi, Matt W Mutka, and Li Xiao. A miniature 25 grams running and jumping robot. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5115–5120. IEEE, 2014.

[6] Christopher J Dudley, Alexander C Woods, and Kam K Leang. A micro spherical rolling and flying robot. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5863–5869. IEEE, 2015.

[7] Ludovic Daler, Julien Lecoeur, Patrizia Bernadette Hählen, and Dario Floreano. A flying robot with adaptive morphology for multi-modal locomotion. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1361–1366. IEEE, 2013.

[8] Zhenishbek Zhakypov, Kazuaki Mori, Koh Hosoda, and Jamie Paik. Designing minimal and scalable insect-inspired multi-locomotion millirobots. *Nature*, 571(7765):381–386, 2019.

[9] Sa-Reum Kim, Dae-Young Lee, Je-Sung Koh, and Kyu-Jin Cho. Fast, compact, and lightweight shape-shifting system composed of distributed self-folding origami modules. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4969–4974. IEEE, 2016.

[10] Lu Lu, Xiangxin Dang, Fan Feng, Pengyu Lv, and Huiling Duan. Conical kresling origami and its applications to curvature and energy programming. *Proceedings of the Royal Society A*, 478(2257):20210712, 2022.

[11] Zhenishbek Zhakypov, Mohsen Falahi, Manan Shah, and Jamie Paik. The design and control of the multi-modal locomotion origami robot, tribot. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4349–4355. IEEE, 2015.

[12] Zhenishbek Zhakypov, Christoph H Belke, and Jamie Paik. Tribot: A deployable, self-righting and multi-locomotive origami robot. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5580–5586. IEEE, 2017.

[13] Shuhei Miyashita, Steven Guitron, Shuguang Li, and Daniela Rus. Robotic metamorphosis by origami exoskeletons. *Science Robotics*, 2(10):eaao4369, 2017.

[14] Sang-Min Baek, Sojung Yim, Soo-Hwan Chae, Dae-Young Lee, and Kyu-Jin Cho. Ladybird beetle–inspired compliant origami. *Science Robotics*, 5(41):eaaz6262, 2020.

[15] Joshua Kaufmann, Priyanka Bhovad, and Suyi Li. Harnessing the multistability of kresling origami for reconfigurable articulation in soft robotic arms. *Soft Robotics*, 2021.

[16] Priyanka Bhovad, Joshua Kaufmann, and Suyi Li. Peristaltic locomotion without digital controllers: Exploiting multi-stability in origami to coordinate robotic motion. *Extreme Mechanics Letters*, 32:100552, 2019.

[17] Tao Jin, Long Li, Tianhong Wang, Guopeng Wang, Jianguo Cai, Yingzhong Tian, and Quan Zhang. Origami-inspired soft actuators for stimulus perception and crawling robot applications. *IEEE Transactions on Robotics*, 2021.

[18] Alexander Pagano, Tongxi Yan, Brian Chien, A Wissa, and S Tawfick. A crawling robot driven by multi-stable origami. *Smart Materials and Structures*, 26(9):094007, 2017.

[19] Kiju Lee, Yanzhou Wang, and Chuanqi Zheng. Twister hand: Underactuated robotic gripper inspired by origami twisted tower. *IEEE Transactions on Robotics*, 36(2):488–500, 2020.

[20] Shuai Wu, Qiji Ze, Jize Dai, Nupur Udipi, Glaucio H Paulino, and Ruike Zhao. Stretchable origami robotic arm with omnidirectional bending and twisting. *Proceedings of the National Academy of Sciences*, 118(36):e2110023118, 2021.

[21] Amir Firouzeh, Marco Salerno, and Jamie Paik. Stiffness control with shape memory polymer in underactuated robotic origamis. *IEEE Transactions on Robotics*, 33(4):765–777, 2017.

[22] Davide Zappetti, Seung Hee Jeong, Jun Shintake, and Dario Floreano. Phase changing materials-based variable-stiffness tensegrity structures. *Soft robotics*, 7(3):362–369, 2020.

[23] Yang Yang, Yonghua Chen, Yingtian Li, Zheng Wang, and Yunquan Li. Novel variable-stiffness robotic fingers with built-in position feedback. *Soft robotics*, 4(4):338–352, 2017.

[24] Wei Wang, Chak Yuk Yu, Pablo Antonio Abrego Serrano, and Sung-Hoon Ahn. Shape memory alloy-based soft finger with changeable bending length using targeted variable stiffness. *Soft robotics*, 7(3):283–291, 2020.

[25] Jiefeng Sun and Jianguo Zhao. An adaptive walking robot with reconfigurable mechanisms using shape morphing joints. *IEEE Robotics and Automation Letters*, 4(2):724–731, 2019.

[26] Yoël Forterre, Jan M Skotheim, Jacques Dumais, and Lakshminarayanan Mahadevan. How the venus flytrap snaps. *Nature*, 433(7024):421–425, 2005.

[27] ML Smith, GM Yanega, and A Ruina. Elastic instability model of rapid beak closure in hummingbirds. *Journal of theoretical biology*, 282(1):41–51, 2011.

[28] Joshua Kaufmann, Priyanka Bhovad, and Suyi Li. Harnessing the multistability of kresling origami for reconfigurable articulation in soft robotic arms. *Soft Robotics*, 9(2):212–223, 2022.

[29] Dinesh K Patel, Xiaonan Huang, Yichi Luo, Mrunmayi Mungekar, M Khalid Jawed, Lining Yao, and Carmel Majidi. Highly dynamic bistable soft actuator for reconfigurable multimodal soft robots. *Advanced Materials Technologies*, 8(2):2201259, 2023.

[30] Shuang Wu, Gregory Langston Baker, Jie Yin, and Yong Zhu. Fast thermal actuators for soft robotics. *Soft Robotics*, 9(6):1031–1039, 2022.

[31] Johannes TB Overvelde, Tamara Kloek, Jonas JA D'haen, and Katia Bertoldi. Amplifying the response of soft actuators by harnessing snap-through instabilities. *Proceedings of the National Academy of Sciences*, 112(35):10863–10868, 2015.

[32] Yichao Tang, Yinding Chi, Jiefeng Sun, Tzu-Hao Huang, Omid H Maghsoudi, Andrew Spence, Jianguo Zhao, Hao Su, and Jie Yin. Leveraging elastic instabilities for amplified performance: Spine-inspired high-speed and high-force soft robots. *Science advances*, 6(19):eaaz6912, 2020.

[33] Jakob A Faber, Andres F Arrieta, and André R Studart. Bioinspired spring origami. *Science*, 359(6382):1386–1391, 2018.

[34] Haijie Zhang, Elisha Lerner, Bo Cheng, and Jianguo Zhao. Compliant bistable grippers enable passive perching for micro aerial vehicles. *IEEE/ASME Transactions on Mechatronics*, 26(5):2316–2326, 2020.

[35] Yongkang Jiang, Xin Tong, Jian Li, Hui Li, Chongjing Cao, Xing Gao, and Yingtian Li. Reprogrammable bistable actuators for multimodal, fast, and ultrasensitive grasping. *IEEE/ASME Transactions on Mechatronics*, 2023.

[36] Jiefeng Sun, Brandon Tighe, and Jianguo Zhao. Tuning the energy landscape of soft robots for fast and strong motion. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10082–10088. IEEE, 2020.

[37] Lei Jiang, Bo Li, Wentao Ma, Yehui Wu, Ruiyu Bai, Wenjie Sun, Yanjie Wang, and Guimin Chen. Electroactive soft bistable actuator with adjustable energy barrier and stiffness. *IEEE Transactions on Robotics*, 2023.

[38] Aniket Pal and Metin Sitti. Programmable mechanical devices through magnetically tunable bistable elements. *Proceedings of the National Academy of Sciences*, 120(15):e2212489120, 2023.

[39] Yuhang Liu, Kai Luo, Shuai Wang, Xiaodong Song, Zhijuan Zhang, Qiang Tian, and Haiyan Hu. A soft and bistable gripper with adjustable energy barrier for fast capture in space. *Soft Robotics*, 10(1):77–87, 2023.

[40] Yinding Chi, Yaoye Hong, Yao Zhao, Yanbin Li, and Jie Yin. Snapping for high-speed and high-efficient butterfly stroke–like soft swimmer. *Science Advances*, 8(46):eadd3788, 2022.

[41] Zechen Xiong, Yufeng Su, and Hod Lipson. Fast untethered soft robotic crawler with elastic instability. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2606–2612. IEEE, 2023.

[42] Zechen Xiong, Zihan Guo, Li Yuan, Yufeng Su, Yitong Liu, and Hod Lipson. Rapid grasping of fabric using bionic soft grippers with elastic instability. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6449–6455. IEEE, 2023.

[43] Yinding Chi, Yichao Tang, Haijun Liu, and Jie Yin. Leveraging monostable and bistable pre-curved bilayer actuators for high-performance multitask soft robots. *Advanced Materials Technologies*, 5(9):2000370, 2020.

[44] Lifeng Zhou, Alexander E Marras, Hai-Jun Su, and Carlos E Castro. Direct design of an energy landscape with bistable dna origami mechanisms. *Nano letters*, 15(3):1815–1821, 2015.

[45] Andres F Arrieta, Valentin Van Gemmeren, Aaron J Anderson, and Paul M Weaver. Dynamics and control of twisting bi-stable structures. *Smart Materials and Structures*, 27(2):025006, 2018.

[46] Tian Chen, Osama R Bilal, Kristina Shea, and Chiara Daraio. Harnessing bistability for directional propulsion of soft, untethered robots. *Proceedings of the National Academy of Sciences*, 115(22):5698–5702, 2018.

[47] Andrew Spielberg, Brandon Araki, Cynthia Sung, Russ Tedrake, and Daniela Rus. Functional co-optimization of articulated robots. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5035–5042. IEEE, 2017.

[48] Raphael Deimel, Patrick Irmisch, Vincent Wall, and Oliver Brock. Automated co-design of soft hand morphology and control strategy for grasping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1213–1218. IEEE, 2017.

[49] Krishnamanaswi M Digumarti, Christian Gehring, Stelian Coros, J Hwangbo, and Roland Siegwart. Concurrent optimization of mechanical design and locomotion control of a legged robot. In *Mobile Service Robotics*, pages 315–323. World Scientific, 2014.

[50] Sehoon Ha, Stelian Coros, Alexander Alspach, Joohyung Kim, and Katsu Yamane. Joint optimization of robot design and motion parameters using the implicit function theorem. In *Robotics: Science and systems*, volume 13, pages 10–15607, 2017.

[51] Thomas Liao, Grant Wang, Brian Yang, Rene Lee, Kristofer Pister, Sergey Levine, and Roberto Calandra. Data-efficient learning of morphology and controller for a microrobot. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2488–2494. IEEE, 2019.

[52] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science robotics*, 7(62):eabk2822, 2022.

[53] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[54] Keene Chin, Tess Hellebrekers, and Carmel Majidi. Machine learning for soft robotic sensing and control. *Advanced Intelligent Systems*, 2(6):1900171, 2020.

[55] James M Bern, Yannick Schnider, Pol Banzet, Nitish Kumar, and Stelian Coros. Soft robot control with a learned differentiable model. In *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 417–423. IEEE, 2020.

[56] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[57] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[58] Mina Alibeigi, Majid Nili Ahmadabadi, and Babak Nadjar Araabi. A fast, robust, and incremental model for learning high-level concepts from human motions by imitation. *IEEE Transactions on Robotics*, 33(1):153–168, 2016.

[59] David Hoeller, Nikita Rudin, Dhionis Sako, and Marco Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024.

[60] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[61] Tianjian Chen, Zhanpeng He, and Matei Ciocarlie. Hardware as policy: Mechanical and computational co-optimization using deep reinforcement learning. *arXiv preprint arXiv:2008.04460*, 2020.

[62] Ye Yuan, Yuda Song, Zhengyi Luo, Wen Sun, and Kris Kitani. Transform2act: Learning a transform-and-control policy for efficient agent design. *arXiv preprint arXiv:2110.03659*, 2021.

[63] Zhanpeng He and Matei Ciocarlie. Morph: Design co-optimization with reinforcement learning via a differentiable hardware model proxy. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7764–7771. IEEE, 2024.

[64] Sharfin Islam, Zhanpeng He, and Matei Ciocarlie. Task-based design and policy co-optimization for tendon-driven underactuated kinematic chains. *arXiv preprint arXiv:2405.14566*, 2024.

[65] Kevin Sebastian Luck, Heni Ben Amor, and Roberto Calandra. Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning. In *Conference on Robot Learning*, pages 854–869. PMLR, 2020.

[66] Andrew Spielberg, Allan Zhao, Yuanming Hu, Tao Du, Wojciech Matusik, and Daniela Rus. Learning-in-the-loop optimization: End-to-end control and co-design of soft robots through learned deep latent representations. *Advances in Neural Information Processing Systems*, 32, 2019.

[67] Hiromi Yasuda, Tomohiro Tachi, Mia Lee, and Jinkyu Yang. Origami-based tunable truss structures for non-volatile mechanical memory operation. *Nature communications*, 8(1):1–7, 2017.

[68] Zirui Zhai, Yong Wang, and Hanqing Jiang. Origami-inspired, on-demand deployable and collapsible mechanical metamaterials with tunable stiffness. *Proceedings of the National Academy of Sciences*, 115(9):2032–2037, 2018.

[69] Guan Liwen, Xu Huayang, and Liu Zhihua. Kinematic analysis of cable-driven parallel mechanisms based on minimum potential energy principle. *Advances in Mechanical Engineering*, 7(12):1687814015622339, 2015.

[70] Yingchao Zhang, Yang Jiao, Jian Wu, Yinji Ma, and Xue Feng. Configurations evolution of a buckled ribbon in response to out-of-plane loading. *Extreme Mechanics Letters*, 34:100604, 2020.

[71] Tianjian Chen, Zhanpeng He, and Matei Ciocarlie. Co-designing hardware and control for robot hands. *Science Robotics*, 6(54):eabg2133, 2021.

[72] Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[73] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter 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)*, pages 23–30. IEEE, 2017.

[74] Tingwu Wang, Yuhao Zhou, Sanja Fidler, and Jimmy Ba. Neural graph evolution: Towards efficient automatic robot design. *arXiv preprint arXiv:1906.05370*, 2019.

[75] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[76] John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

[77] Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R Walter. Jointly learning to construct and control agents using deep reinforcement learning. In *2019 international conference on robotics and automation (ICRA)*, pages 9798–9805. IEEE, 2019.

[78] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

[79] Elisha Lerner, Zhe Chen, and Jianguo Zhao. Reconfigurable origami with variable stiffness joints for adaptive robotic locomotion and grasping. *Philosophical Transactions A*, 382(2283):20240017, 2024.

[80] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[81] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[82] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.