2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

# Task-Based Design and Policy Co-Optimization for Tendon-driven Underactuated Kinematic Chains

Sharfin Islam*†, Zhanpeng He*‡, Matei Ciocarlie†
https://roamlab.github.io/tentamorph/

***Abstract*— Underactuated manipulators reduce the number of bulky motors, thereby enabling compact and mechanically robust designs. However, fewer actuators than joints means that the manipulator can only access a specific manifold within the joint space, which is particular to a given hardware configuration and can be low-dimensional and/or discontinuous. Determining an appropriate set of hardware parameters for this class of mechanisms, therefore, is difficult - even for traditional task-based co-optimization methods. In this paper, our goal is to implement a task-based design and policy co-optimization method for underactuated, tendon-driven manipulators. We first formulate a general model for an underactuated, tendon-driven transmission. We then use this model to co-optimize a three-link, two-actuator kinematic chain using reinforcement learning. We demonstrate that our optimized tendon transmission and control policy can be transferred reliably to physical hardware with real-world reaching experiments.**

## I. Introduction

Underactuated manipulators require a carefully designed transmission, often tendon-driven, to take advantage of a reduced number of actuators in the robot. Such designs range from planar serial chains with relatively few links to complex, hyper-redundant continuum robots [1–7]. In all of these cases, being able to reduce the number of actuators means that we can build smaller and more lightweight designs, place actuation at more proximal locations in the chain, and take advantage of passive compliance in the un-actuated degrees of freedom.

However, the compromise of such designs is that, with fewer actuators than degrees of freedom, underactuated manipulators can directly access only a certain manifold within the overall state space. This manifold, which contains the set of obtainable states for a particular hardware configuration, can be low dimensional and/or discontinuous. These limitations affect our ability to plan controllers that smoothly transition between various states to accomplish a desired task. The process of tuning the hardware parameters in order to ensure the set of accessible states matches the desired task can be slow and cumbersome, particularly if, as is often the case, changes in the design parameters of the underactuated transmission have a counter-intuitive effect the overall behavior of the robot.

Co-optimization, or the process of simultaneously searching the space of both hardware and control, is a possible solution to the problem of ensuring that a given hardware design is capable of a desired task. Such methods have been attempted in the context of underactuated kinematic chains, but often restricted to simulation [8–10], limited validation of the simulated controller on real hardware [11], or only implemented on single-actuator systems [12]. The fundamental difficulty of such approaches lies in formulating a co-design problem that a) enables the use of non-trivial controllers or policies, b) can be solved to an acceptable optimum point, c) guarantees that the final result can be physically realized in real hardware, and finally d) ensures that the optimized control policy also transfers to real hardware without substantial loss of performance. This is a difficult set of goals to achieve simultaneously, and, to the best of our knowledge, no current method has done so for underactuated, tendon-driven transmissions with multiple actuators.

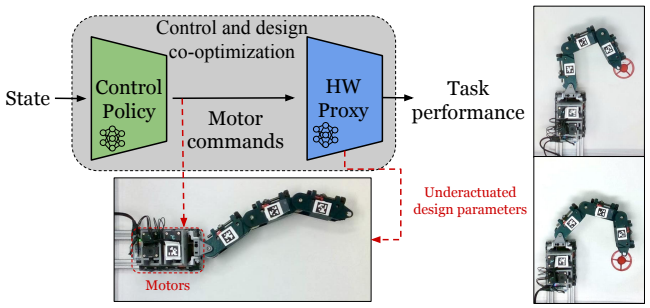


Fig. 1: We optimize an underactuated, tendon-driven transmission for kinematic chains. We formulate and parameterize a general model for N links and M actuators. We apply our model to a three-link, two-actuator tentacle that we co-optimize using reinforcement learning (top row, left). We then validate our results on physical hardware (right).

In this paper, we start with an underactuated tendon transmission model for general planar kinematic chains, which can capture a diverse set of hardware configurations. We use a forward model that captures the behavior of the system as a function of design parameters and control inputs. We then show that this model enables end-to-end co-optimization of design and control policies for a specified task. To solve the co-optimization problem, we adopt MORPH [13], a recently introduced end-to-end co-optimization framework that uses a proxy model to mimic the effect of hardware, and generate gradients of hardware parameters w.r.t. tasks performance. This allows us to learn both control and hardware parameters to accomplish desired tasks. We then show that it is possible to build a real robotic manipulator with this optimized transmission, and validate that our optimized control policy can be transferred reliably to real hardware.

An additional feature of our hardware design is that some design parameters can be modified without requiring re-

*Equal contributions. †Dept. of Mechanical Engineering, ‡Dept. of Computer Science, Columbia University, New York, USA

assembly of the manipulator, and by using the same set of fabricated components. This allows for different levels of flexibility in robot behaviors. If the task requires explorations of diverse behaviors, we can optimize all hardware parameters to explore a large solution space. In the case that the task is simpler, we can optimize only the easily adaptable parameters to avoid reassembly. While in this paper we focus on a single kinematic chain optimized for simple reaching tasks, our directional goal to enable underactuated, tendon transmission models that can be co-optimized along with their control policies, which can in turn transfer to the real world. Overall, the main contributions of this paper include:

- We formulate a model of underactuated, tendon-driven, passively-compliant planar kinematic chains that enables efficient end-to-end task-based optimization of the design and control parameters.
- We show that the results of the co-optimization process can be transferred to real hardware implementing the optimized design parameters. The resulting robots can then run the optimized control policy which also achieves sim2real transfer in task performance.
- To the best of our knowledge, this is the first time that task-based, policy and design co-optimization methods have been demonstrated for underactuated manipulators with multi-dimensional manifolds.

## II. Related Work

An underactuated manipulator is a mechanical system composed of links and joints that has fewer actuators than degrees of freedom [14]. The transmission of these manipulators are often tendon-driven, as a single actuated tendon can be routed to control multiple joints. Reducing the number of bulky actuators enables designers to build more compact and lightweight manipulators. Moreover, tendon-transmission allow designers to dislocate the motors from the joints. Dislocating the actuators reduces the inertia of the links and makes it easier to make the manipulator robust to water, dust, and other difficult environmental conditions [15]. With so many practical benefits, a myriad of diverse underactuated, tendon-driven manipulators have been proposed over the past several decades.

There is an extensive history of underactuated manipulator design. The first underactuated manipulator, Hirose's soft gripper introduced in 1978, was designed to softly capture a diverse range of objects with uniform pressure [1]. Over time, these manipulator designs became much more advanced and their applications now extend to more robust and intricate grasping behaviors [16–20]. Currently, there is extensive research in the design and control of underactuated manipulators for tasks even more dexterous than grasping, such as in-hand manipulation [21–23]. To realize these more advanced capabilities, the tendon-transmission of these manipulators had to be carefully designed, optimized, and further hand-tuned. This process is cumbersome and time-consuming, but necessary. Co-optimization methods are a possible tool to help design these complicated mechanisms. With recent advancements in reinforcement learning, co-optimization is now more powerful than ever.

Recently, researchers have considered reinforcement learning for task-driven design and control co-optimization [24–27]. The key of this line of work is to derive policy gradient w.r.t both the control and design parameters. Chen and He et al. [12] propose to integrate a differentiable model of the hardware with a control policy to adapt hardware design via policy gradients. A key limitation of this approach is the requirement of differentiable physics simulation. In cases when the forward transition cannot be modeled in a differentiable manner, researchers propose methods to learn design parameters in the input or output space of a policy. For instance, Luck et al. [28] propose to learn an expressive latent space to represent the design parameters and condition a policy with a latent design representation. Transform2Act [29] propose to have a transform stage in their policy that estimates actions to modify a robot's design and a control stage that computes control sequences given a specific design. Our hardware design is not differentiable. Hence, in this work, we apply MORPH [13], a method that co-optimizes design and control in parameter space that does not require differentiable physics.

## III. Method

We formalize our problem as follows. Our goal is to build a kinematic chain (i.e., tentacle/trunk) robot that can achieve flexible behaviors, such as reaching desired parts of the workspace, while maintaining the practical design benefits of underactuation. For this class of mechanisms, the design parameters and controller for a given task are innately coupled. Therefore, we employ a task-based hardware optimization approach to identify hardware parameters that yield state manifolds optimized for the desired task performance.

Our method has three main components: 1) an underactuated, tendon-driven transmission design for compliant kinematic chains; 2) a model that captures the forward dynamics of our designed robot; and 3) an end-to-end co-optimization pipeline that can directly learn the parameters of the proposed design along with a control policy for a given reaching task.

### A. Transmission design

Our design, shown in Figs. 1 and 2, assume a kinematic chain with $N$ links driven by $M$ motors. All motors are located inside the base and actuate winches that are connected to the joints via tendons. For each motor $i$, we denote the radius of it's winch as $R_i^A$. We assume that each motor is driving a tendon that traverses the entire length of the chain, thereby helping flex every joint. We denote the length of each link $j$ as $L_j$. For a the corresponding joint $j$, each motor $i$ will wrap around a pulley. We use $R_{ij}{}^f$ to denote the radius of the flexion pulley for this joint and motor pair.

We assume all actuators provide flexion forces, and the transmission uses purely passive extension mechanisms. At each joint, the mechanism features a passive elastic tendon that stretches over a pulley of constant radius (denoted by $R^e$)

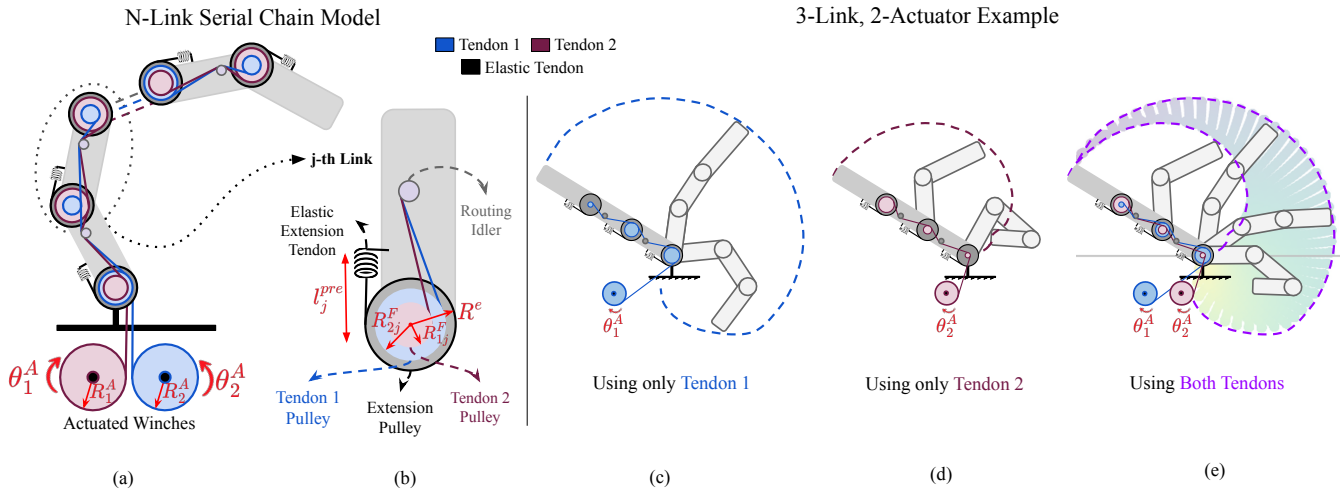


Fig. 2: Our flexible tendon-transmission design for compliant, underactuated kinematic chains. In this design, $N$ links are driven by $M$ actuated flexion tendons. We also implement a passive extension mechanism, which we can precisely pre-tension. The parameters of our transmission are all flexion tendon radii, extension tendon radii, and elastic tendon pre-elongations, as shown on the left. Each actuated tendon gives us access to a single 1-D manifold. By combining the tendons and alternating which tendons are in tension, the workspace of our robot now spans all of the $M$, one-dimensional manifolds created by each actuated flexion tendon. We are assuming quasi-static motion and ignoring gravity, so the robot will settle at a minimum energy position within these manifolds.

to provide a restoring extension torque. Each elastic tendon can be pretensioned individually; we use ${l_j}^{pre}$ to denote the pretensioning elongation of the passive tendon at joint $j$.

This mechanism has a number of desirable characteristics. Underactuation leads to a small number of motors, and placing all motors inside the base frees the links of the kinematic chain from any motors or electronics. However, the movement of the robot is non-trivial to define or control. Critically, robot movement is determined not only by the actuators but also by a number of design parameters. These include all flexion tendon radii, extension tendon radii, and elastic tendon pre-elongations. Furthermore, the ability of a robot to reach specific points in its workspace is clearly also determined by the lengths of the links. Our goal is to devise a procedure capable of optimizing all these design parameters while providing a policy for controlling motor movements in order to achieve a specific task.

In addition, our proposed design makes some of the design parameters easier to change than others. In particular, we mount each elastic tendon on a linear slider mechanism. The position of this slider is set by rotating a lead-screw, thereby controlling the pre-tension elongations of the elastic extension tendons to less than 1mm of precision. This means that some of our design parameters (pulley radii, link lengths) require a full reassembly, while others (elastic tendon elongations) are easier to change depending on the task.

### *B. Forward actuation model for our transmission design*

To enable co-optimization of design and control, we first need a forward actuation model that relates the hardware parameters and the actuator inputs to the resulting state of the manipulator. The input to this model consists of the servo angle of the winches that control flexion tendons, shown in blue and red in Fig 2, which we will define as a column vector $\theta_A$ of size $\mathbb{R}^M$. The state in our model is defined by the set of joint angles $\theta_J$ of size $\mathbb{R}^N$.

As the actuator angles increase, the flexion tendons shorten, causing some joints to flex since these tendons run through all joints. We assume that the flexion tendons are rigid and slack-free when $\theta_j = 0$ for all joints. Thus, our forward models must ensure the change in length of the tendon from the actuator command matches the collective length change of the joint flexing. However, for some joints, the change in length at the joints can be greater than the change in length at the actuated winch - in this case, the tendon will be in slack. If at least one joint angle is non-zero, then there must be at least one tendon that is in tension.

These physical constraints imposed by our rigid tendons define a manifold within the fully-actuated joint space of a N-link serial chain. This manifold is shaped not only by pre-determined hardware parameters but also by the actuator commands. As illustrated in Fig 2, each actuator defines a one-dimensional manifold in the overall joint space, and therefore also the workspace. Using multiple tendons allows us to span these 1-D manifolds and expands our usable workspace. To formalize this, we introduce a generalized constraint that relates these tendon-induced limitations to actuator commands and hardware parameters.

In our model shown in Fig. 2, tendons route over circular pulleys with constant radii. We compose these radii into a matrix of size $\mathbb{R}^{M\times N}$ as follows:

$$\mathbf{R^F} = \begin{bmatrix} R^F_{11} & R^F_{12} & \dots & R^F_{1N} \\ R^F_{21} & R^F_{22} & \dots & R^F_{2N} \\ \dots & \dots & \dots & \dots \\ R^F_{M1} & R^F_{M2} & \dots & R^F_{MN} \end{bmatrix} \tag{1}$$

The radii of the actuated winches can be similarly composed into a diagonal matrix of size $\mathbb{R}^{M\times M}$.

$$\mathbf{R^A} = \begin{bmatrix} R_1^A & 0 & ... & 0 \\ 0 & R_2^A & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & R_M^A \end{bmatrix} \tag{2}$$

With these matrices, we can now define a column vector of the slack collected in the tendons $(w)$.

$$w = \mathbf{R^F}\theta_J - \mathbf{R^A}\theta_A \tag{3}$$

Within the manifold created by the slack, our robot will settle at a point that minimizes the overall energy in the system [14]. In our model (Fig 2), we assume a quasi-static environment and ignoring the effects of gravity. Therefore, the only stored energy in the system comes from the elongation of the elastic extension tendons, we express this as $U$.

The tendons are elongated from two factors – 1) the preloaded elongation and 2) the elongation of the tendon due to the joints flexing. We define the total elongation for a joint $l_j$ as a function of the joint angles and hardware parameters.

$$l_j = R^e\theta_j + l_j^{pre} \tag{4}$$

With formulations for the stored energy and slack, can now pose the forward actuation model to solve for the next state as $\theta'_J = f(\theta_J, \theta_A)$ relating the state to the action as a numerical optimization problem: given $\theta_A$,$\theta_J$,

$$\underset{\theta_J}{\text{argmin}} U = \frac{1}{2}\sum_{j=1}^{N} k l_j^2 \tag{5}$$

$$\text{subject to} \qquad w \geq 0 \tag{6}$$

$$\tag{7}$$

Actions and hardware parameters do not change the global energy landscape, but instead the manifold of the landscape that satisfies our constraints (see Fig.3). Hence, optimizing control $\theta_A$ and hardware parameters (e.g., $\mathbf{R_F}$ and $R_e$) is finding appropriate energy manifolds whose minimum energy states are ones we want to visit for task completion. There exists some combination of hardware parameters that ensures these manifolds are continuous are have a clear energy minima. Arriving at this specific set of parameters relies on observing how the changes in parameters affect the most suitable control strategy for solving a given task. Finally, in this work, we focus on optimizing the following set of hardware parameters: $\phi = (\mathbf{R^F}, l_{pre}, L)$, where $L$ is the set of link lengths and $\mathbf{R^F}$ and $l_{pre}$ are defined in Fig 2.

### C. Task-aware co-optimization of design and control

Equipped with the forward actuation model, we can now focus on the problem of co-optimizing a set of design parameters and a motor control policy for a given task. Using a standard reinforcement learning (RL) formalism, we model each task as a Markov Decision Process (MDP), or a tuple

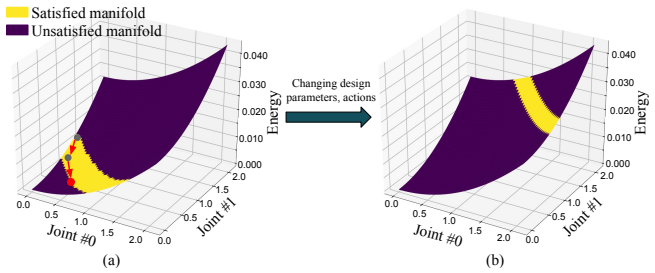


Fig. 3: **Illustrations of our optimization-based forward actuation model.** (a) and (b) show the global energy landscape (z-axis). The yellow regions are manifolds that satisfy our constraints (Eq.6 and Eq.7). The red arrows in (a) represent optimization steps (Eq.5) that find the energy minimum inside the manifold. Changing design parameters and control actions result in a change in the manifold.

$(\mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R}(\boldsymbol{s}, \boldsymbol{a})$ is the reward function, and $\boldsymbol{s'} = \mathcal{F}(\boldsymbol{s}, \boldsymbol{a})$ is the forward transition model, where $\boldsymbol{s}, \boldsymbol{s'} \in \mathcal{S}$, and $\boldsymbol{a} \in \mathcal{A}$.

In this work, we use a state vector $s$ containing the joint angles $\theta_J$, motor positions $\theta_A$, and 2D end-effector positions $\mathrm{x}_{e.e.}$. Actions for a control policy are motor commands $\theta_A$. The reward function $\mathcal{R}(\boldsymbol{s}, \boldsymbol{a})$ encodes the desired performance; for example, if we want the end-effector to reach a specific point, the reward will comprise a negative distance to the goal. For each task, we want to find the parameters of a control policy $\pi_\theta$ as well as the design parameters $\phi$ that optimize task performance, which is evaluated by its expected returns: $\mathbb{E}[\sum_{t=0}^{T} \mathcal{R}(\boldsymbol{s}_t, \boldsymbol{a}_t)]$.

The state transition model $\mathcal{F}$ requires additional considerations. As described in Section III-B, it consists of an optimization-based model $f$ whose behaviors are determined by some of the hardware parameters from $\phi$. However, this model is non-differentiable and, therefore, can not be used in a standard policy gradient optimization routine.

To co-optimize the hardware design with control, we apply MORPH [13], an end-to-end co-optimization method that uses a neural network proxy model $h^{nn}$ to approximate the forward transition $\mathcal{F}$. The proxy model and a control policy are co-optimized with task performance while asking $h^{nn}$ to be close to $\mathcal{F}$. In this work, instead of mimicking just the forward transition, we consider the effect of hardware design parameters in task space. Hence, we ask the hardware proxy to approximate both the forward transition and forward kinematics: $q_\phi = f \circ T_{FK}$. Note that our hardware parameters $\phi$ are encapsulated in different parts of $q$. For example, link lengths only affect forward kinematics, while pulley radii and preloaded tension govern the behaviors of $f$. Hence, the overall optimization objective is:

$$\max_{\theta,} \mathbb{E}_{\pi, h^{nn}}[\underbrace{\sum_{t=0}^{T} \mathcal{R}(s_t, \theta_A)}_{\text{task performance}} + \underbrace{\alpha \| h^{nn}(\theta_J, \theta_A) - q(\theta_J, \theta_A) \|^2}_{\text{hardware constraint}}] \tag{8}$$

where $\alpha$ is a constant number.

To derive explicit hardware design parameters, for every $N$ epochs, we also use CMA-ES [30] to search $\phi$ to match

the updated hardware proxy $h^{nn}$:

$$\min_{\phi} \|h^{nn}(\theta_J, \theta_A) - q_\phi(\theta_J, \theta_A)\|^2 \tag{9}$$

The final algorithm is an iterative process: We first co-optimize the hardware proxy and the control policy to improve task performance, then extract explicit hardware parameters that match the current version of the hardware proxy.

## IV. Experimental Set-up

### A. Design and control co-optimization

To evaluate our method, we optimize the aforementioned design with several goal-reaching tasks, i.e., reaching a goal location in 2D with its end-effector. We have three sets of experiments demonstrating three ways to utilize our co-optimization pipeline for flexible motor skills:

1) **Goal reaching via re-fabrication**: We adapt all design and control parameters to different goals but only train each policy for a single goal. In this specific experiment, reaching multiple goals requires the re-fabrication of a real robot since the link length and pulley radii cannot be changed after assembly without fabricating new parts and reassembling the manipulator.
2) **Goal reaching via online hardware updates**: In this experiment, we optimize the hardware design in two stages: we first optimize design parameters that require re-fabrication (i.e., pulley radii) to a specific goal. Then, we fix all parameters except the pre-tension elongation and co-optimize for a different goal.
3) **Goal reaching via multi-goal control policies**: We optimize all parameters to learn a shared design and control that can achieve multiple goals with a single control policy. In this case, we do not need to adapt any of the hardware parameters to reach multiple goals after the policy is trained.

The first experiment is our design's most limited application due to the need for re-fabrication. However, it demonstrates that our transmission can be optimized to reach different areas of the fully actuated workspace. We also train control policies with fixed initial design parameters to compare task performance. Fig.5 shows the goal distribution for the first experiment set. The second set of experiments leverages a key aspect of our design - the ability to easily adapt some design parameters without having to go through tedious re-design, fabrication, and assembly. For this second set, we keep the same pulley radii and only optimize the pretension elongation for a goal different from the first experiment. In the first and second experiments, we use the state space described in Section.III for inputs of our control policy.

The last experiment demonstrates that our robot, while being underactuated, can achieve different goals with the same design parameters and control policy—if the desired goals are not too far away. The state space is extended from the first two experiment sets with the addition of the goal locations. To establish the specific set of goals we hope to achieve, We sample goals randomly around a center location $(-0.16, 0)$ within a radius of $50\text{mm}$. For evaluation, we randomly sample

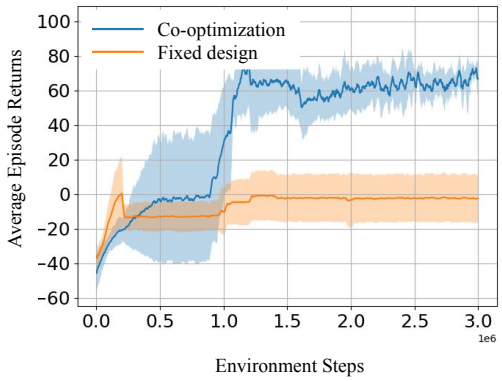


Fig. 4: Average episode returns for goal reaching via re-fabrication.

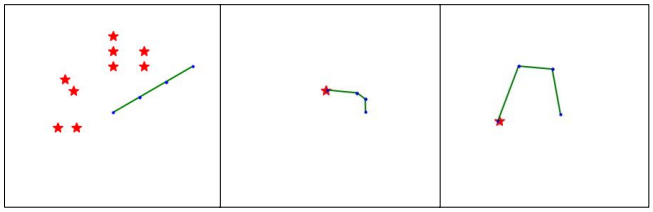


Fig. 5: Qualitative results for optimizing underactuated robots to reach different goals in simulation (goal reaching via re-fabrication). (a) shows original design and goal locations. (b) and (c) show two optimized underactuated tentacles reaching different goals.

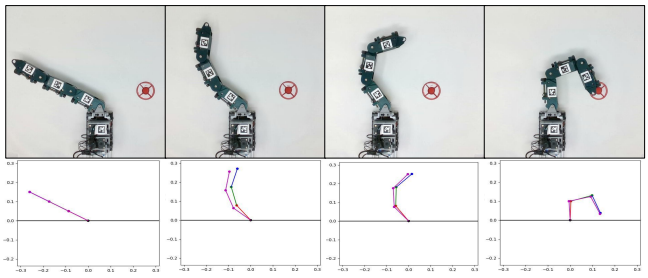

Fig. 6: Example of policy rollout on the real robot. Top: real robot rollout. Bottom: comparisons between simulated robots and real robot tracking. The real robot is shown in purple in the bottom row of figures.

20 goals and execute the policy 10 times for each goal. This experiment requires re-assembly of the entire tentacle as we optimize all the design parameters. Therefore, we only run this experiment in simulation.

For all experiments, We use a dense reward function: $r = ||\mathrm{x}_{e.e.} - \mathrm{x}_{goal}||^2 + b$, where $\mathrm{x}_{e.e}$ and $\mathrm{x}_{goal}$ are the position of the end-effector and the goal, $b$ is a task completion bonus that is provided when the distance of E.E. and the goal is less than $2\text{mm}$. We initialize the hardware parameters to be $R_1^F = [0.005, 0.005, 0.005]$ and $R_2^F = [0.005, 0.01, 0.02]$, $L = [0.8, 0.8, 0.8]$ and $l^{pre} = [0, 0, 0]$. The units of all these parameters is meters.

### B. Hardware implementation and sim-to-real transfer

Shown in Fig.1 and 6, we physically build the manipulator with the optimized pulley radii in the first two experiment

TABLE I: **Final distance to goal position of optimized robot for real-world reaching tasks.** Open and closed represent open-loop and closed-loop execution, respectively. All units of distance in this table are mm, and each value is calculated for 20 samples.

| | Avg. distance to goal | | Std. deviation | |
|---|---|---|---|---|
| Stage | Open | Closed | Open | Closed |
| Pulley radii | 13.27 | 6.30 | 7.28 | 2.87 |
| Preloads | 21.27 | 12.97 | 7.26 | 4.39 |

TABLE II: **Quantitative results for sim-to-real hardware transfer.** Average distance between real hardware states and simulated states for both open-loop and closed-loop policy rollouts. Each value is calculated over a total of 40 samples.

| | Open-loop | Closed-loop |
|---|---|---|
| Joint angle (rad) | 0.176 | 0.0932 |
| End-effector position (m) | 17.523 | 10.877 |

sets. To simplify both the fabrication and assembly, we set $(R^e)$ = 15mm. Our transmission in this robot is driven by two Dynamixel XM-430 servos that sense and control the angle $(\theta_a)$ of the actuated winches. During our experiments, we fix the manipulator to a experimental rig, which has a mounted camera looking down on the robot. The camera is used in lieu of joint angle encoders, as we calculate the joint angles of our robot $(\theta_J)$by tracking the pose of AR tags attached to our robot as shown in Fig. 6. We use the joint angles collected from the AR tags to execute closed-loop policies for the first two experiment sets. In this case, we close the loop by taking observations from the real robot during policy execution and use our optimized control policy to determine the next action. In addition to closed-loop policies, we also simply train a control policy to reach the goal in simulation, and then directly transfer this policy to the real-robot without adjusting any of the actions during runtime.

We run both the first and second experiment set on the real robot hardware for both closed-loop and open-loop policies. In the next section, we evaluate the accuracy and precision over 20 samples on the real robot for each set of data.

## V. Results and Analysis

### A. Co-optimization results

Our results from the first set of experiments show that our model learns different design parameters for different goal locations with an error 1.80mm. As shown in Fig.5, our model can adapt hardware parameters to different goals. The initial design can achieve only 1 out of 9 goals, achieving much lower average returns than our method (Fig.4) when we only learn a control policy with a fixed design. Although the robot's workspace may contain the goal, the initial hardware parameters make learning a stable control policy difficult. After the design parameters are optimized, the control policy can learn a stable action sequence to arrive at the goal position.

In the second experiment set, our results show we can optimize only the pulley radii to achieve one goal $(0.13, 0.3)$ with 1.8mm error in simulation. Then, we fix the pulley radii and only optimize the preloaded extension to another goal $(0.16, -0.08)$ with 3.2mm error in simulation. Using derived design parameters, we build a real tentacle robot and directly transfer the control policy from simulation to the real world.

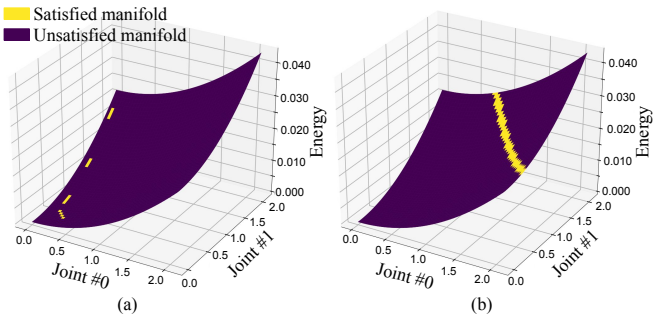


Fig. 7: Manifold comparisons before (a) and after (b) hardware optimization. Each grid represents the constraint satisfaction (yellow for constraints satisfied, dark purple for not satisfied) of its center. We set the joint angles of #2 to be 0 for visualization purposes.

Our results show that the transfer policy achieves, on average, 6.3mm error to the goal location. During training, since the control policy is co-optimized with the non-stationary hardware proxy, it is robust to hardware parameters perturbation. We test both open- and closed-loop performance on the real robot. In the open-loop test case, we record action sequences executed from the simulation and directly execute them on the real robot. The robot can touch the goal with 12.97mm errors.

### B. Sim-to-real accuracy

A key focus in our paper is the ability to transfer the optimized design and control policy to physical robotic hardware. We achieve accurate transfer in terms of task performance, as shown in Table I. Our closed-loop results show that taking observations directly from the real robot improves task performance substantially. The closed-loop policies' standard deviation of the distance to the goal is much lower, which indicates that they yield more consistent results in the real world. Additionally, the average final distance to the goal for the closed-loop policies is much lower, indicating higher accuracy. Our policy is robust to actuation errors by reasoning about the current robot state and producing actions that correct its errors. In Table II, we calculate the average difference between the real robot state and the simulated robot state for both the closed-loop and open-loop sets of experiments. The average error in joint space for closed-loop policy execution is about half the error for open-loop execution. Similarly, the average distance between the real and simulated end-effector positions for the entire action sequence of each experiment is much lower for closed-loop policies.

### C. Energy manifold optimization

Our forward model is an optimization-based model. Each forward step requires many optimization steps to find a global energy minimum, which is crucial for accurately simulating

our robot. Each set of hardware parameters, along with a control action, corresponds to a manifold in the energy landscape. Although the global energy landscape does not have any local optima, a manifold can be discontinuous and has multiple local minimums. This introduces difficulties for optimization and constrains our choice of optimization algorithm to global optimization (e.g., genetic algorithms). However, given the time budget, global optimization can be inefficient and slow down our simulated robot since it generally searches in a large state space.

As mentioned in Section III-B, when we optimize the control and hardware parameters of this robot, we optimize the manifold for energy optimization. Our experiments show that our co-optimization process results in an energy manifold that is easy to learn by task. As shown in Fig.7, the original hardware design parameters provide manifolds that are discontinuous, spread in different regions in the energy landscape, and have several local minimums. After MORPH, the resulting manifold (see Fig.7b) is a continuous and smooth manifold. To further analyze the optimized manifold's effect, we apply Sequential Least Squares Programming optimizer (SLSQP) [31], a local optimization algorithm with a time budget of 500 optimization steps and 100 random actions, to both the unoptimized and optimized manifolds and compare their results to those of a global search algorithm implemented in-house. When the hardware is unoptimized, given the same time budget, the simulated results have a much higher error (0.42) than the optimized hardware design (0.18). This means that the resulting manifold of our optimization is more suitable for fast simulation with local optimization.

## VI. Conclusion

In this work, we present a method for co-optimizing the design and control of underactuated kinematic chains. The key to our approach is an optimization-based forward actuation model that effectively captures the behavior of our robot design, and a co-optimization pipeline is capable of learning with non-differentiable physics. Our experimental results from simulation and the real world demonstrate that our design, along with the flexibility provided by hardware optimization, results in flexible robot capabilities while enjoying the benefits of underactuation. A key limitation of our current work is the task complexity. While being general, our forward actuation model assumes quasi-static, making contact-rich tasks difficult. In future works, we aim to extend our work to more complex tasks. Another future direction is to further utilize our online adaptable design and explore novel mechanisms to make part of the design adaptable.

## References


[1] Shigeo Hirose and Yoji Umetani. "The development of soft gripper for the versatile robot hand". In: *Mechanism and Machine Theory* 13.3 (1978), pp. 351–359. ISSN: 0094-114X. DOI: https://doi.org/10.1016/0094-114X(78)90059-9. URL: https://www.sciencedirect.com/science/article/pii/0094114X78900599.

[2] Matei Ciocarlie and Peter Allen. "Data-driven optimization for underactuated robotic hands". In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 1292–1299. DOI: 10.1109/ROBOT.2010.5509793.

[3] Matei Ciocarlie, Fernando Mier Hicks, and Scott Stanford. "Kinetic and dimensional optimization for a tendon-driven gripper". In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 2751–2758. DOI: 10.1109/ICRA.2013.6630956.

[4] Angus B. Clark, Lois Liow, and Nicolas Rojas. "Force Evaluation of Tendon Routing for Underactuated Grasping". In: *Journal of Mechanical Design* 143.10 (Apr. 2021), p. 104502. ISSN: 1050-0472. DOI: 10.1115/1.4050382. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/143/10/104502/6679773/md_143_10_104502.pdf. URL: https://doi.org/10.1115/1.4050382.

[5] Priyanka Rao, Quentin Peyron, Sven Lilge, and Jessica Burgner-Kahrs. "How to Model Tendon-Driven Continuum Robots and Benchmark Modelling Performance". In: *Frontiers in Robotics and AI* 7 (2021). ISSN: 2296-9144. DOI: 10.3389/frobt.2020.630245. URL: https://www.frontiersin.org/articles/10.3389/frobt.2020.630245.

[6] Matteo Russo, Seyed Mohammad Hadi Sadati, Xin Dong, Abdelkhalick Mohammad, Ian D. Walker, Christos Bergeles, Kai Xu, and Dragos A. Axinte. "Continuum Robots: An Overview". In: *Advanced Intelligent Systems* 5.5 (2023), p. 2200367. DOI: https://doi.org/10.1002/aisy.202200367. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aisy.202200367. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/aisy.202200367.

[7] Bowen Yi, Yeman Fan, Dikai Liu, and Jose Guadalupe Romero. *Simultaneous Position-and-Stiffness Control of Underactuated Antagonistic Tendon-Driven Continuum Robots*. 2023. arXiv: 2306.03865 [cs.RO].

[8] Raymond R. Ma and Aaron M. Dollar. "Linkage-Based Analysis and Optimization of an Underactuated Planar Manipulator for In-Hand Manipulation". In: *Journal of Mechanisms and Robotics* 6.1 (Nov. 2013), p. 011002. ISSN: 1942-4302. DOI: 10.1115/1.4025620. eprint: https://asmedigitalcollection.asme.org/mechanismsrobotics/article-pdf/6/1/011002/6251362/jmr_006_01_011002.pdf. URL: https://doi.org/10.1115/1.4025620.

[9] 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)* (2017), pp. 1213–1218. URL: https://api.semanticscholar.org/CorpusID:3037063.

[10] Luca Barbazza, Damiano Zanotto, Giulio Rosati, and Sunil K. Agrawal. "Design and Optimal Control of an Underactuated Cable-Driven Micro–Macro Robot". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 896–903. DOI: 10.1109/LRA.2017.2651941.

[11] Jie Xu, Tao Chen, Lara Zlokapa, Michael Foshey, Wojciech Matusik, Shinjiro Sueda, and Pulkit Agrawal. "An End-to-End Differentiable Framework for Contact-Aware Robot Design". In: *ArXiv* abs/2107.07501 (2021). URL: https://api.semanticscholar.org/CorpusID:235651322.

[12] Tianjian Chen, Zhanpeng He, and Matei Ciocarlie. "Hardware as Policy: Mechanical and ComputationalCo-Optimization using Deep Reinforcement Learning". In: *Conference on Robotic Learning (CoRL)*. 2020.

[13] Zhanpeng He and Matei T. Ciocarlie. "MORPH: Design Co-optimization with Reinforcement Learning via a Differentiable Hardware Model Proxy". In: *ArXiv* abs/2309.17227 (2023). URL: https://api.semanticscholar.org/CorpusID:263310528.

[14] Zhong-Ping Jiang. "Controlling Underactuated Mechanical Systems: A Review and Open Problems". In: *Advances in the Theory of Control, Signals and Systems with Physical Modeling*. Ed. by Jean Lévine and Philippe Müllhaupt. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 77–88. ISBN: 978-3-642-16135-3. DOI: 10.1007/978-3-642-16135-3_7. URL: https://doi.org/10.1007/978-3-642-16135-3_7.

[15] Ryuta Ozawa, Kazunori Hashirii, and Hiroaki Kobayashi. "Design and control of underactuated tendon-driven mechanisms". In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 1522–1527. DOI: 10.1109/ROBOT.2009.5152222.

[16] Aaron M. Dollar and Robert D. Howe. "The SDM Hand: A Highly Adaptive Compliant Grasper for Unstructured Environments". In: *Experimental Robotics*. Ed. by Oussama Khatib, Vijay Kumar, and George J. Pappas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 3–11. ISBN: 978-3-642-00196-3.

[17] Yanyu Su, Yan Wu, Kyuhwa Lee, Zhijiang Du, and Yiannis Demiris. "Robust grasping for an under-actuated anthropomorphic hand under object position uncertainty". In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. 2012, pp. 719–725. DOI: 10.1109/HUMANOIDS.2012.6651599.

[18] Giorgio Grioli, Manuel Catalano, Emanuele Silvestro, Simone Tono, and Antonio Bicchi. "Adaptive synergies: An approach to the design of under-actuated robotic hands". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 1251–1256. DOI: 10.1109/IROS.2012.6385881.

[19] Cosimo Della Santina, Cristina Piazza, Giorgio Grioli, Manuel G. Catalano, and Antonio Bicchi. "Toward Dexterous Manipulation With Augmented Adaptive Synergies: The Pisa/IIT SoftHand 2". In: *IEEE Transactions on Robotics* 34.5 (2018), pp. 1141–1156. DOI: 10.1109/TRO.2018.2830407.

[20] Tianjian Chen, Tianyi Zhang, and Matei Ciocarlie. "Design Paradigms Based on Spring Agonists for Underactuated Robot Hands: Concepts and Application". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 7100–7106. DOI: 10.1109/ICRA48506.2021.9561832.

[21] L. Udawatta, K. Watanabe, K. Izumi, and K. Kiguchi. "Control of underactuated robot manipulators using switching computed torque method: GA based approach". English. In: *Soft Computing* 8.1 (2003), pp. 51–60. ISSN: 1432-7643. DOI: 10.1007/s00500-002-0257-8.

[22] Lael U. Odhner and Aaron M. Dollar. "Dexterous manipulation with underactuated elastic hands". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 5254–5260. DOI: 10.1109/ICRA.2011.5980263.

[23] Andrew S. Morgan, Kaiyu Hang, Bowen Wen, Kostas Bekris, and Aaron M. Dollar. "Complex In-Hand Manipulation Via Compliance-Enabled Finger Gaiting and Multi-Modal Planning". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4821–4828. DOI: 10.1109/LRA.2022.3145961.

[24] Lucy Jackson, Celyn Walters, Steve Eckersley, Pete Senior, and Simon Hadfield. "ORCHID: Optimisation of Robotic Control and Hardware In Design using Reinforcement Learning". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 4911–4917. DOI: 10.1109/IROS51168.2021.9635865.

[25] David R Ha. "Reinforcement Learning for Improving Agent Design". In: *Artificial Life* 25 (2018), pp. 352–365. URL: https://api.semanticscholar.org/CorpusID:52945447.

[26] Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R Walter. "Jointly learning to construct and control agents using deep reinforcement learning". In: *IEEE Intl. Conf. on Robotics and Automation*. IEEE. 2019, pp. 9798–9805.

[27] Tingwu Wang, Yuhao Zhou, Sanja Fidler, and Jimmy Ba. "Neural Graph Evolution: Automatic Robot Design". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=BkgWHnR5tm.

[28] 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*. 2019. URL: https://api.semanticscholar.org/CorpusID:208139268.

[29] Ye Yuan, Yuda Song, Zhengyi Luo, Wen Sun, and Kris M. Kitani. "Transform2Act: Learning a Transform-and-Control Policy for Efficient Agent Design". In: *ArXiv* abs/2110.03659 (2021). URL: https://api.semanticscholar.org/CorpusID:238419578.

[30] Nikolaus Hansen and Andreas Ostermeier. "Completely derandomized self-adaptation in evolution strategies". In: *Evolutionary computation* 9.2 (2001), pp. 159–195.

[31] D. Kraft. *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988. URL: https://books.google.com/books?id=4rKaGwAACAAJ.