Zero-shot Sim-to-Real Transfer for Reinforcement Learning-based Visual Servoing of Soft Continuum Arms

Hsin-Jung Yang[1,*] HJY@IASTATE.EDU
Mahsa Khosravi[1,*] MAHSAK@IASTATE.EDU
Benjamin Walt[2,*] WALT@ILLINOIS.EDU
Girish Krishnan[2] GKRISHNA@ILLINOIS.EDU
Soumik Sarkar[1] SOUMIKS@IASTATE.EDU
[1]*Iowa State University;* [2]*University of Illinois Urbana-Champaign;* [*] *Equal Contribution*

Editors: N. Ozay, L. Balzano, D. Panagou, A. Abate

Abstract

Soft continuum arms (SCAs) soft and deformable nature presents challenges in modeling and control due to their infinite degrees of freedom and non-linear behavior. This work introduces a reinforcement learning (RL)-based framework for visual servoing tasks on SCAs with zero-shot sim-to-real transfer capabilities, demonstrated on a single section pneumatic manipulator capable of bending and twisting. The framework decouples kinematics from mechanical properties using an RL kinematic controller for motion planning and a local controller for actuation refinement, leveraging minimal sensing with visual feedback. Trained entirely in simulation, the RL controller achieved a 99.8% success rate. When deployed on hardware, it achieved a 67% success rate in zero-shot sim-to-real transfer, demonstrating robustness and adaptability. This approach offers a scalable solution for SCAs in 3D visual servoing, with potential for further refinement and expanded applications.

Keywords: Soft continuum arms, Visual servoing, Reinforcement learning, Sim-to-real transfer

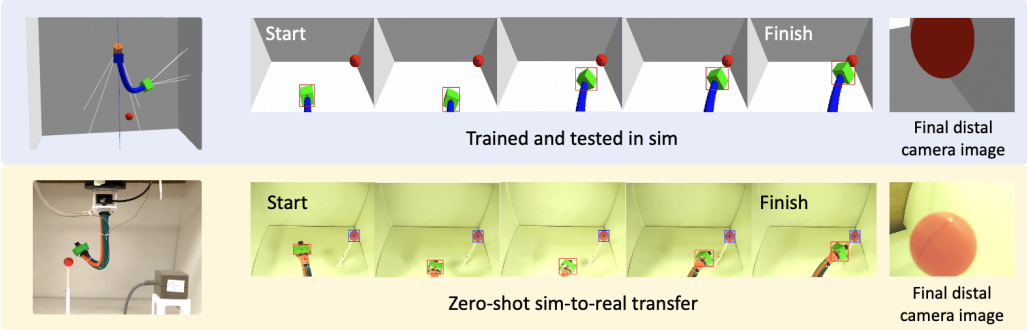


Figure 1: Overview of the proposed RL-based visual servoing control framework for SCAs with zero-shot sim-to-real transfer capability. The framework is used to visual servo to view a target as shown here, in sim (top) and on real hardware (bottom). The image sequence illustrates the base camera views after each policy step, with the final distal camera view (right) showcasing the RL-based controller's ability to locate and center the target. Demo video in supplementary materials.

1. Introduction

Soft continuum arms (SCAs) are increasingly recognized for their ability to safely and effectively interact with complex, unstructured environments. Their ability to conform and apply gentle forces

makes them ideal for tasks such as handling delicate objects or working in close proximity to humans (Chen et al., 2022; Zongxing et al., 2020; Banerjee et al., 2018; Chen et al., 2021; Venter and Dirven, 2017). However, their soft and deformable nature introduces challenges for modeling and control. Learning-enabled methods, such as model-free reinforcement learning (RL), offer a promising solution by learning behaviors directly from data rather than relying on analytically derived models (Falotico et al., 2024). Despite these advantages, one of the primary obstacles to deploying SCAs in real-world is the sim-to-real transfer, where policies trained in simulation fail to generalize well on physical systems. For SCAs, this challenge is amplified due to their unique physical characteristics. Unlike rigid robots, SCAs exhibit continuous deformation and high compliance, leading to non-linear behaviors that are difficult to model and generalize (Rus and Tolley, 2015). While prior work has demonstrated sim-to-real transfer for rigid robots using low-fidelity models and staged fine-tuning in simulation and on hardware (Leguizamo et al., 2022), such methods are not easily extended to SCAs due to their complex dynamics. Moreover, effective control of SCAs often requires significant sensing capabilities, such as accurate positional tracking or detailed environmental feedback, to account for their dynamic interactions with the environment. While high-fidelity simulations, such as the Cosserat rod model (Till et al., 2019; Janabi-Sharifi et al., 2021; Xun et al., 2023), could theoretically address these discrepancies, they are computationally expensive and unsuitable for RL, which relies on large-scale data collection.

Existing RL-based approaches have attempted to bridge the sim-to-real gap but often remain constrained by sensing limitations or task-specific designs. For instance, (Satheeshbabu et al., 2019; Wu et al., 2020; Morimoto et al., 2021; Li et al., 2024) implemented RL-based methods for 2D tasks with minimal sensing setups, such as single-camera systems or onboard sensors. However, these solutions struggled to extend to more high-dimensional scenarios. In contrast, (Satheeshbabu et al., 2020) extended RL to 3D navigation using Vicon motion capture systems, achieving high accuracy at the cost of extensive infrastructure. (Thuruthel et al., 2018) employed multi-sensor feedback for dynamic control, which showcases adaptability and precision but requires significant sensing resources. Meanwhile, some studies validated their methods without testing or deploying policies on physical hardware (Goharimanesh et al., 2020). Notably, none of these works have demonstrated zero-shot sim-to-real transfer, leaving a critical gap in the field. These limitations often arise from the inherent challenges of scaling RL frameworks to address the non-linear dynamics of SCAs while maintaining robust sim-to-real transfer capabilities. A summary of relevant methods is provided in Table 1. Please see the supplementary materials for detailed related works.

To address these challenges, we propose a framework for SCAs that decouples kinematics from their mechanical properties by employing two components: an *RL kinematic controller* and a *local controller*. The *RL kinematic controller* focuses on learning high-level kinematic policies, such as desired curvature and torsion, while the *local controller* translates these commands into actuation signals, compensating for dynamic uncertainties and physical variations. By focusing on kinematics goals rather than full dynamic fidelity, our approach abstracts away complexities related to actuation and mechanical properties, accelerating the RL training process. Moreover, our framework leverages a minimum sensing approach, reducing the reliance on extensive setups such as multi-camera tracking systems, while primarily leveraging visual feedback from cameras and measurements from simple trackers. The above combination not only reduces computational burden, but also enhances transferability of learned policies by making them less sensitive to the underlying physical system.

We validate our framework using the BR2 manipulator (Uppalapati and Krishnan, 2021) through experiments demonstrating zero-shot sim-to-real transfer in visual servoing tasks in 3D space. Our

results show that we achieved zero-shot sim-to-real transfer with a 67% success rate, showcasing the effectiveness of our proposed framework in bridging the sim-to-real gap. In summary, this paper introduces a control framework that decouples the kinematics of SCAs from their mechanical properties, simplifying policy learning and enhancing policy transferability. By leveraging minimum sensing, the approach negates the need for extensive sensing, while integrating a *local controller* to handle real-world variations. We demonstrate the effectiveness of this framework through zero-shot sim-to-real transfer of an *RL kinematic controller* on the BR2, achieving high accuracy in visual servoing tasks for 3D navigation and object tracking in both simulation and real-world experiments.

Literature	Goal	Model-Free	Minimal Sensing	Closed-Loop	3D Task Space	Zero-Shot Sim-to-Real Transfer
Thuruthel et al. (2018)	Dynamic control	×	×	✓	✓	×
Satheeshbabu et al. (2019)	Position control	✓	✓	×	×	×
Satheeshbabu et al. (2020)	Path tracking	✓	×	✓	✓	×
Morimoto et al. (2021)	Pose control	✓	✓	✓	×	×
Wu et al. (2020)	Position control	×	×	✓	×	×
Goharimanesh et al. (2020)	Trajectory tracking	×	×	✓	×	×
Li et al. (2024)	Visual servoing	✓	✓	✓	×	×
Present Work	Visual servoing	✓	✓	✓	✓	✓

Table 1: A comparison of RL-based approaches for control of SCAs, highlighting key features. Minimal Sensing refers to methods that optimize sensory inputs relative to the complexity of the task, such as leveraging onboard sensors, or a limited number of cameras instead of extensive systems such as multi-camera tracking setups.

2. Preliminaries

2.1. Modeling the BR2 with Constant Curvature and Constant Torsion Model

The BR2 manipulator (Uppalapati and Krishnan, 2021) is a unique type of SCA with a parallel architecture composed of soft pneumatic actuators known as Fiber Reinforced Elastomeric Enclosures (FREEs). Unlike many existing soft manipulators, the BR2 utilizes an asymmetric configuration of FREEs, enabling it to bend and rotate simultaneously and achieve complex spatial deformation patterns. This design allows the BR2 to navigate around obstacles with enhanced flexibility while maintaining a parallel structural framework, which contributes to precise and adaptive control.

Modeling the BR2 requires accounting for both bending and torsional deformations. To achieve this, the BR2 is parameterized along its length, with the position $r(s)$ and orientation $R(s)$ defined at each point s along its length. When actuated, the arm experiences a bending strain, $\kappa(s)$, and torsional strain, $\tau(s)$. These strains, when assuming negligible shear and stretching strains, can be related to the pose via the following differential equations: $r'(s) = R(s)\boldsymbol{v}(s), \quad R'(s) = R(s)\hat{\boldsymbol{u}}(s)$ where $\hat{[\cdot]}$ is the usual mapping from $\mathbb{R}^3$ to $\mathfrak{so}(3)$, $\boldsymbol{u} = [\kappa(s), 0, \tau(s)]^\top$, and $\boldsymbol{v} = [0, 0, 1]^\top$. When combined with equations derived from static equilibrium, these differential equations form the Kirchhoff rod solution. In this work, we assume κ and τ are constant and thus form a constant curvature and torsion model. This simplified model provides a closed-form solution to

relate pose and configuration to the strains (κ and τ), enabling rapid simulation modeling that is essential for the extensive training data required for effective RL.

2.2. Deep Reinforcement Learning

Deep reinforcement learning (DRL) (Mnih et al., 2013) extends traditional RL by leveraging the representational power of deep neural networks. It operates within the framework of a Markov Decision Process (MDP), characterized by a four-tuple (S, A, P, R), where: S is the state space, representing all possible states s_t at time t; A the action space, defining the set of all possible actions a_t at time t; P the transition probability function, which specifies the likelihood of transitioning from state s_t to state s_{t+1} given an action a_t; $R : S \times A \times S' \to \mathbb{R}$ defines the reward function.

The agent's decision-making process is governed by policy π, which maps the state s_t to an action a_t. In DRL, this policy is parameterized by neural networks, represented as π_θ, where θ denotes the learnable parameters of the neural network. The goal of DRL is to find an optimal policy π_θ^* that maximizes the expected reward over an episode τ, defined as a sequence of states and actions $(s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$. The expected reward is expressed as $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1})\right]$, where γ is the discount factor and T is the episode length.

DRL algorithms can be broadly classified into on- and off-policy approaches. Among these, Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Soft Actor-Critic (SAC) (Haarnoja et al., 2018) are two popular algorithms in their respective categories. On-policy algorithms like PPO optimizes policies by directly sampling data from the current policy, making them inherently less sample efficient. In contrast, off-policy algorithms like SAC utilize a replay buffer to leverage past experience, which significantly improves sample efficiency. SAC also incorporates entropy regularization, which encourages exploration by balancing the trade-off between expected rewards and policy stochasticity. This mechanism enhances robustness during training and reduces the likelihood of convergence to suboptimal policies. In this work, we adopt SAC due to its sample efficiency and entropy regularization, which together make it well-suited for our problem.

3. Methodology

Our goal is to develop an RL-based controller for visual servoing of SCAs in 3D space with zero-shot sim-to-real transfer. To enable generalizable control, we propose 1) a novel two-layer framework that decouples kinematics from mechanical properties and 2) adopt an open-vocabulary object detection model for feature extraction. An *RL kinematic controller* plans high-level motion in configuration space, while a *local controller* translates these plans into actuation, compensating for physical uncertainties. Visual feedback from a distal and a base camera is processed by the object detector to extract task-relevant features that guide the RL policy.

3.1. Decoupling Kinematics and Mechanical Properties

As seen in Fig. 2b), we perform this decoupling by leveraging the Configuration Space of the BR2, which is described in terms of strains: Curvature (κ) and Torsion (τ). It is then possible to create two maps: one between Actuation Space and Configuration Space and a second between Configuration Space and Task Space. The first map depends on the mechanical properties of the SCA, such as material, actuation, design, manufacturing, etc., and thus varies greatly - even with time. Creating this map from physical principles is challenging and almost always relies on some data driven approach.

The second map is purely kinematic and independent of the particulars of the BR2 manipulator. By creating an *RL kinematic controller* that works in Configuration Space, i.e., the second map, we are able to ensure that the controller can be applied to different hardware configurations. To handle the first map between Actuation Space and Configuration Space, a *local controller* is employed. It is used to iteratively refine the actuation through a correction loop (Fig. 2c). This allows the system to avoid the need for an custom Configuration-to-Actuation map and still achieve the desired configuration albeit iteratively. A general Configuration-to-Actuation map, created from data generated by a Cosserat rod model created for previous work (Ripperger and Krishnan, 2023), is used for large movements and refined by the *local controller*.

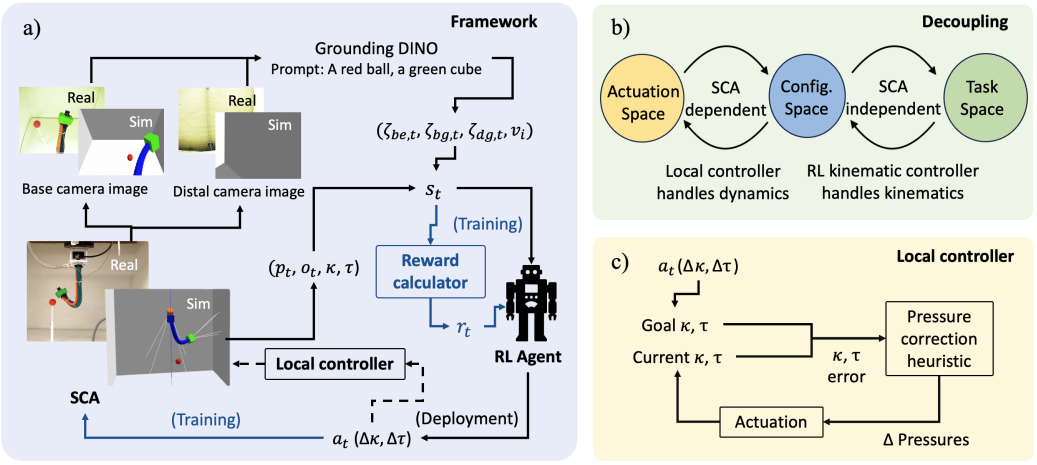


Figure 2: **a)** Training and deployment framework of the RL kinematic controller. During training (black + blue paths), the RL agent learns a policy in simulation. In deployment (black + dashed paths), the trained policy outputs kinematic actions, translated into actuation by the *local controller*. **b)** Decoupling kinematics and mechanical properties: The *RL kinematic controller* handles the kinematics of the SCA, which is independent to specific hardware variations. The *local controller* handles the dynamics of the SCA during operation, which tries to achieve the goal configuration determined by the *RL kinematic controller*. **c)** The *local controller* achieves the goal configuration without using a Configuration-to-Actuation map. Current κ,τ are estimated and the configuration error is passed to the heuristic, which generates an change in actuation. This process is iterated until the goal configuration is achieved.

3.2. RL Problem Formulation

Following the MDP framework, we define the state space, action space, and reward as follows:

State space The state space represents all possible states of the SCA. The state s_t includes the current position p_t and orientation o_t of the end-effector, configuration parameters (κ_t and τ_t), the bounding box centroids of both the end-effector and the target from the base camera image $(\zeta_{be,t}, \zeta_{bg,t})$, the bounding box centroid of the target from the distal camera image $(\zeta_{dg,t})$, and a target visibility boolean based on the detection from the detection model (v_t). Formally, it can be represented as: $s_t = [p_t, o_t, \kappa_t, \tau_t, \zeta_{be,t}, \zeta_{bg,t}, \zeta_{dg,t}, v_t]$.

Action space The action space encompasses all available actions. The actions consist of adjustments in curvature and torsion at each time step. Formally, they can be expressed as: $a_t = [\Delta\kappa, \Delta\tau]$. The actions are bounded between [-1, 1] and are scaled by a set of preset factors during stepping.

Reward The reward function comprises the following components:
— *The distance-based reward*, $r_d = e^{-\ln 2(40d/\pi)^2}$, encourages reduction in the Euclidean distance d in meters between the current position of the end-effector and the target.
— *The angle-based reward*, $r_a = e^{-\ln 2(8\alpha/\pi)^2}$, incentivizes alignment of the end-effector's orientation with the target. Here, α is the angle between the end-effector's normal vector and the vector from the end-effector to the target.
— *The visual information-based reward* r_i, utilizes feedback from the distal camera to refine alignment with the target. This reward minimizes the visual discrepancy between the target's bounding box centroid and the center of the distal camera frame. It is given as $r_a = 5e^{-2\pi(d_i/l)^2}$ if the target is visible in distal camera, 0 otherwise. Here d_i is the distance between the target's bounding box centroid in pixels, $\zeta_{dg,t}$, and the frame's center, ζ_c. l represents half the diagonal length of the distal camera frame, also in pixels.
— *The task completion reward* r_c, reinforces successful completion of the task, which is assigned when the distance between the target's bounding box centroid and the distal camera's frame, d_i, is less than 100 pixels. The task completion reward is $r_c = 128$ if $d_i \leq 100$, 0 otherwise.
— *The time penalty* $r_p = -10$ penalizes the agent for prolonged episode duration, encouraging efficient task completion. The total reward at each time step is given by $r_t = r_d + r_a + r_i + r_c + r_p$.

The reward function is designed to balance the contributions of each component while maintaining a hierarchy of priorities. Exponential terms create sharper gradients near desired values, providing stronger guidance as the agent approaches critical goals, such as minimizing Euclidean distance or achieving alignment. The scaling factors and specific values are chosen to prioritize key objectives. For instance, task completion carries the highest reward to emphasize goal achievement, while rewards such as the distance- and angle-based rewards, are scaled to encourage incremental progress without overshadowing the importance of overall success. This structure ensures that the agent focuses on the most critical aspects of the task, fostering effective learning.

3.3. RL Training Framework

Fig. 2a) illustrates the training framework for the *RL kinematic controller* (black + blue paths). In this framework, Grounding DINO (Liu et al., 2024), is used to detect bounding boxes from the images captured by the distal and base camera. From the base camera image, the positions of both the end-effector and the target are extracted, while from the distal camera image, only the position of the target is extracted when visible. These detections provide visual information that are processed to form part of the state input for the RL agent. Within this framework, the RL agent interacts with the environment iteratively. At each time step, the agent receives the current state of the SCA from the simulator and executes an action to adjust the SCA's configuration. The simulator processes this control signal to update the state based on the underpinning model while the reward calculator evaluates the effectiveness of the agent's action in achieving the desired objectives. This feedback is then returned to the RL agent in the form of a scalar reward, allowing it to refine its control policy over iterative interactions with the environment.

Simulation environment setup The simulation environment was built in Gazebo (Koenig and Howard, 2004), modeling the BR2 under the assumption of constant curvature and torsion. To rep-

resent the SCA, the simulation includes a series of 2 cm spheres spaced along the arm's length, with cameras providing visual feedback from both the base and the distal tip. A 3 cm red sphere represents the target object in the workspace, which is enclosed on three sides to mirror real-world constraints. The environment was wrapped using Gymnasium (Towers et al., 2024), enabling seamless integration with RL algorithms and standardized training pipelines.

3.4. Evaluation Metrics

To assess the performance of the trained RL agent, we define a set of evaluation metrics that capture both task success and control precision:

Success rate The percentage of trials in which success is achieved, i.e., target centered in the distal camera image, or $d_i \leq 100$. Here d_i is the distance in pixels between the target's bounding box centroid and center of the distal camera's frame.

Steps to goal Defined as the number of iterations needed to achieve success.

Repeatability Defined as the percentage of repeated outcomes on a point by point basis.

3.5. Training and Evaluating the *RL kinematic controller*

Training The training for the *RL kinematic controller* is conducted entirely in simulation. Each training episode begins with the BR2 in a randomized initial configuration and the target placed at a random position within the workspace. An episode terminates either when the RL agent successfully completes the task, defined as centering the target in the distal camera frame (refer to success rate in the previous subsection), or when the agent exceeds the maximum allowed steps per episode, set to 8 in this study. The training process employs the implementation of the Soft Actor-Critic (SAC) algorithm (Haarnoja et al., 2018) from Stable-Baselines3 (Raffin et al., 2021), with a total training duration of 150k steps. Details on the hyperparameter settings and the reward plot during the training can be found in supplementary materials.

Evaluation The trained *RL kinematic controller* is evaluated in simulation to ensure its performance in visual servoing tasks before the sim-to-real transfer. The evaluation consisted of 500 randomized episodes, with the testing procedures mirroring the training, including randomized initial configurations for the BR2 and target, and the same termination criteria. The sampled target positions were distributed throughout the workspace, as shown in Fig. 3 b) and c), ensuring that the evaluation comprehensively covered a wide range of configurations.

The evaluation focused on key metrics: success rate, average steps to task completion, and the target centroid distance from the center of the distal camera image. The trained RL kinematic controller achieved a success rate of 99.8%, requiring 3.98 steps on average to complete the visual servoing task. As shown in Fig. 3b) and c), the scatter plots of target positions represent the outcomes of each episode, where blue dots indicate successful episodes carried out by the trained policy, and red dots represent failures. The high density of blue dots demonstrates the controller's robust generalization across the workspace.

3.6. Deployment Framework

The deployment framework, also illustrated in Fig. 2a), integrates the trained *RL kinematic controller* with the *local controller* (black + dashed paths). The process begins with an input containing

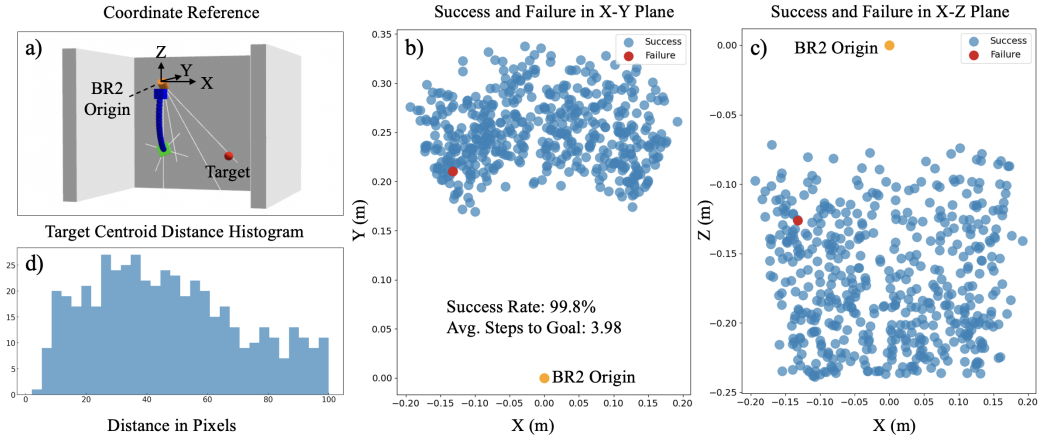


Figure 3: **a)** Coordinate reference. **b)** and **c)** Scatter plots of the sampled target positions in the workspace. Each dot represents the outcome of an episode, where blue dots indicate successful goal-reaching and red dots represent failures. These plots illustrate the generalization capability of the trained RL kinematic controller across the workspace, with a high density of blue dots demonstrating robust performance. **d)** Histogram of the distance between the target bounding box centroid and the center of the distal camera frame.

target end-effector pose information, provided as prompts to the object detector along with images from the distal and base cameras. The RL kinematic controller processes this input and other kinematic data to determine the goal configuration that aligns the SCA with the specified goal. Once the target configuration is established, it is passed to the *local controller*, which translates this kinematic goal into actuation.

Local controller The *local controller*, shown in Fig. 2c), is used to generate actuations to achieve the desired configuration using an iterative three step process: 1) Using the tip sensor pose data, the current arm configuration is estimated. 2) The error between the estimated and desired configuration is used by a heuristic (see supplementary materials for more details) to produce a change in actuation. 3) The new actuation is applied and after reaching steady state, it returns to Step 1. This is iterated until the desired accuracy is achieved. This method avoids being over reliant on a custom Configuration-to-Actuation map unique to each BR2. To set the initial configuration a general but likely inaccurate Configuration-to-Actuation map is used and the above configuration corrective loop is used to achieve the goal configuration.

4. Sim-to-Real Transfer Experiments

4.1. Hardware Setup

Our hardware setup closely mirrors the simulation environment, featuring the BR2 equipped with two cameras — a base camera to capture a fixed view of the workspace and a distal camera mounted on the distal tip. The setup includes target objects positioned on stands of varying heights, as well as a tracking instrument (Polhemus Patriot) to monitor the pose of the end-effector tip. The base camera is angled 45° downward to provide a comprehensive view of the workspace, while the distal camera is aligned with the axial line of the arm, offering a direct view of the tip's orientation relative to the target. The *local controller* is implemented as a simple closed-loop controller to achieve the desired RL policy configurations without the need for a custom Configuration-to-Actuation map.

4.2. Testing Procedure

To validate the effectiveness of the deployed RL policy, we designed a testing procedure to challenge the BR2 to reach various target positions within the workspace, which is as follows:
— *Initial setup*: Each test starts with the BR2 in a random configuration with the tip visible in the base camera and the target object placed at one of 50 test positions through out the workspace and within the frame of the base camera.
— *Planning and execution*: The RL policy plans for the next configuration and the *local controller* executes the plan. The control loop continues until task completion or time out.
— *Evaluation, repositioning and repeat*: After each test, the target object is repositioned to a new location. For each new target position, the procedure is repeated, with key metrics—such as positioning accuracy, alignment with the target, steps taken and success rate recorded for each trial.
— *Additional weights*: To evaluate the effectiveness of the local controller to overcome variations in the Configuration-to-Actuation map (Fig. 2b), weights (10g, 15g and 20g) were added to the tip of the BR2. A subset of the test points were then tested in the same manner noted above.

4.3. Results and Discussion

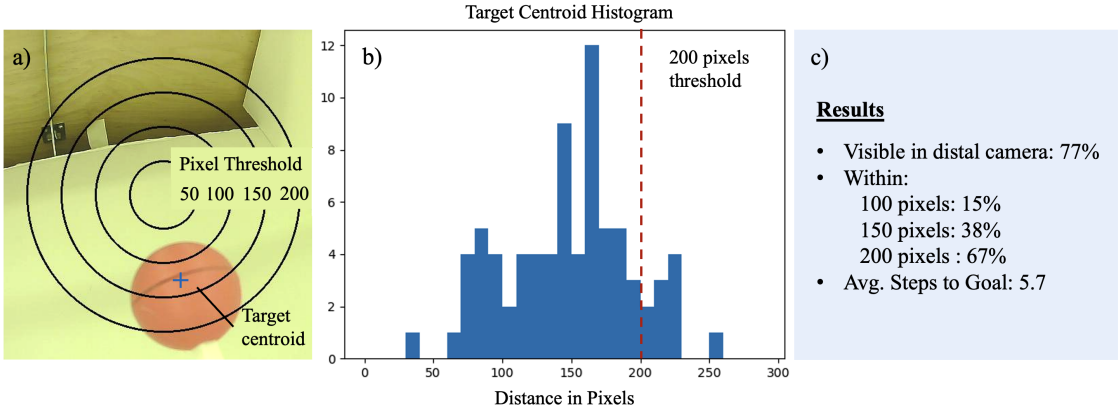
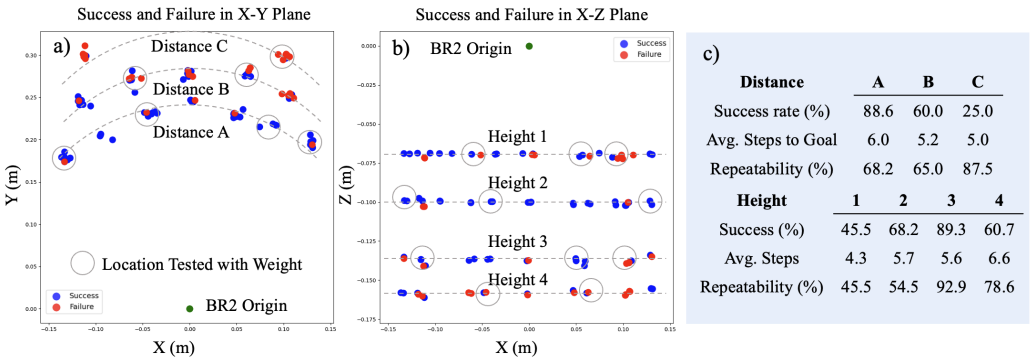


Figure 4: Tip View Results. **a)** A representative view from the tip camera. The rings indicate pixel distance thresholds. **b)** A histogram of best centroid distances in pixels. **c)** A summary of the results of testing.



Distance	A	B	C
Success rate (%)	88.6	60.0	25.0
Avg. Steps to Goal	6.0	5.2	5.0
Repeatability (%)	68.2	65.0	87.5
Height	**1**	**2**	**3**	**4**
Success (%)	45.5	68.2	89.3	60.7
Avg. Steps	4.3	5.7	5.6	6.6
Repeatability (%)	45.5	54.5	92.9	78.6

Figure 5: Regional Results. **a)** A top down view of the test area showing three distances tested. **b)** A view into the test area showing the four test heights. Locations of tip weight testing are marked with circles. **c)** A summary of the results based on test point region.

Three levels of success were analyzed based on the target centroid's distance from the distal camera center measured: 100 pixels, 150 pixels and 200 pixels (Fig. 4a). Fifty target positions throughout the workspace were tested twice for a total of 100 samples. Overall it was detected in the tip view by Grounding DINO in 77% of the tests. The histogram in Fig. 4b shows that few targets centroids were brought within the 100 pixel threshold used in the simulation. This is due to challenges in performing small corrective movements on the real hardware as these corrections were close to the controller error threshold. Fig. 4c demonstrates the results for the different levels of success. These results are weaker than the results observed in the simulation, but this is to be expected given the challenges of sim-to-real transfer. We selected a 200 pixel threshold as the standard for the hardware tests because, at this threshold, it is still possible to fully view the target in the tip camera and the image is useful for tasks such as inspection and counting.

A regional analysis of success and failure can be seen in Fig. 5, but still demonstrate that policy works on hardware. This demonstrates the policy is most accurate in the central region, middle height and close to the reach of the BR2. This region uses the least rotational actuation which leads to less model error and there is less perspective distortion with the world image. It is worth noting that in every test case, even if it was not able to view the target with the tip camera, the BR2 arm servoed toward the target. In addition to success, we examined the repeatability of the results. There was an overall repeatability of 70%. A regional breakdown can be seen in Fig. 5c. Naturally strong success or failure leads to repeatability. The results indicate that the lower test points in addition to having lower success, are also inconsistent in their performance. The primary failure mode was excessive curvature, which appears to be driven by differences in real world BR2 response compared to the constant curvature and torsion assumption used in training. Additionally, a lack of depth information likely contributes to failures with greater target distance.

Weights (10g, 15g and 20g) were attached to the tip of the BR2 and a subset of the test points were tested and for the 200 pixel threshold. The 10g achieved 57.1% and the 15g and 20g both achieved 50%. These results indicate that the controller can overcome map inaccuracies. However, as the weight increased, the system struggled to achieve the more extreme positions as actuations reached the pressure limits. While a non-negligible gap remains, our results demonstrate meaningful sim-to-real transfer under realistic, minimally-instrumented conditions. Unlike prior works relying on extensive sensing or hardware-specific tuning, our single policy—trained entirely in simulation—handles randomized 3D visual servoing using only visual feedback. This highlights the robustness and generality of our approach for SCAs. Please refer to supplementary materials for detailed discussion and more analysis of the sim-to-real errors.

5. Conclusion and Future Work

In this work, we demonstrated a zero shot sim-to-real transfer of an RL policy to visually servo a BR2. The policy was trained on a reduced order constant curvature and torsion model based simulation and successfully tested on real hardware. In future work, we hope to expand on the success of our present work. We aim to improve the system's success rate by refining learning and control strategies. Particularly, we plan to enhance tip view alignment for tighter centering thresholds and expand the workspace by adding degrees of freedom, enabling tasks beyond image centering, such as grasping or multi-angle inspection. Lastly, we also intend to leverage to the power of Grounding DINO to work with a variety of targets in unstructured environments.

Acknowledgments

This work is funded from a joint NSF-USDA COALESCE grant 2021-67021-34418 and NSF CPS Frontier #1954556. Additional funding by AIFARMS National AI Institute in Agriculture, backed by Agriculture and Food Research Initiative (Grant Number: 2020-67021-32799).

References

Hritwick Banerjee, Zion Tsz Ho Tse, and Hongliang Ren. Soft robotics with compliance and adaptation for biomedical applications and forthcoming challenges. *Int. J. Robot. Autom*, 33(1):68–80, 2018.

Shoue Chen, Yaokun Pang, Yunteng Cao, Xiaobo Tan, and Changyong Cao. Soft robotic manipulation system capable of stiffness variation and dexterous operation for safe human–machine interactions. *Advanced Materials Technologies*, 6(5):2100084, 2021.

Xiaoqian Chen, Xiang Zhang, Yiyong Huang, Lu Cao, and Jinguo Liu. A review of soft manipulator research, applications, and opportunities. *Journal of Field Robotics*, 39(3):281–311, 2022.

Egidio Falotico, Enrico Donato, Carlo Alessi, Elisa Setti, Muhammad Sunny Nazeer, Camilla Agabiti, Daniele Caradonna, Diego Bianchi, Francesco Piqué, Yasmin Tauqeer Ansari, et al. Learning controllers for continuum soft manipulators: Impact of modeling and looming challenges. *Advanced Intelligent Systems*, page 2400344, 2024.

Masoud Goharimanesh, Ali Mehrkish, and Farrokh Janabi-Sharifi. A fuzzy reinforcement learning approach for continuum robot control. *Journal of Intelligent & Robotic Systems*, 100(3):809–826, 2020.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Farrokh Janabi-Sharifi, Amir Jalali, and Ian D. Walker. Cosserat rod-based dynamic modeling of tendon-driven continuum robots: A tutorial. *IEEE Access*, 9:68703–68719, 2021. doi: 10.1109/ACCESS.2021.3077186.

Nathan P. Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004. doi: 10.1109/IROS.2004.1389727.

David Felipe Leguizamo, Hsin-Jung Yang, Xian Yeow Lee, and Soumik Sarkar. Deep reinforcement learning for robotic control with multi-fidelity models. *IFAC-PapersOnLine*, 55(37):193–198, 2022.

Jinzhou Li, Jie Ma, Yujie Hu, Li Zhang, Zhijie Liu, and Shiying Sun. Vision-based reinforcement learning control of soft robot manipulators. *Robotic Intelligence and Automation*, 44(6):783–790, 2024.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection, 2024. URL https://arxiv.org/abs/2303.05499.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

Ryota Morimoto, Satoshi Nishikawa, Ryuma Niiyama, and Yasuo Kuniyoshi. Model-free reinforcement learning with ensemble for a soft continuum robot arm. In *2021 IEEE 4th International Conference on Soft Robotics (RoboSoft)*, pages 141–148. IEEE, 2021.

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. URL http://jmlr.org/papers/v22/20-1364.html.

Evan Ripperger and Girish Krishnan. Design space enumerations for pneumatically actuated soft continuum manipulators. In *Volume 8: 47th Mechanisms and Robotics Conference (MR)*, page V008T08A097, 08 2023. doi: 10.1115/DETC2023-116930. URL https://doi.org/10.1115/DETC2023-116930.

Daniela Rus and Michael T Tolley. Design, fabrication and control of soft robots. *Nature*, 521 (7553):467–475, 2015.

Sreeshankar Satheeshbabu, Naveen Kumar Uppalapati, Girish Chowdhary, and Girish Krishnan. Open loop position control of soft continuum arm using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5133–5139. IEEE, 2019.

Sreeshankar Satheeshbabu, Naveen K Uppalapati, Tianshi Fu, and Girish Krishnan. Continuous control of a soft continuum arm using deep reinforcement learning. In *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 497–503. IEEE, 2020.

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

Thomas George Thuruthel, Egidio Falotico, Federico Renda, and Cecilia Laschi. Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators. *IEEE Transactions on Robotics*, 35(1):124–134, 2018.

John Till, Vincent Aloi, and Caleb Rucker. Real-time dynamics of soft and continuum robots based on cosserat rod models. *The International Journal of Robotics Research*, 38(6):723–746, 2019.

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.

Naveen Kumar Uppalapati and Girish Krishnan. Design and modeling of soft continuum manipulators using parallel asymmetric combination of fiber-reinforced elastomers. *Journal of Mechanisms and Robotics*, 13, 2 2021. ISSN 1942-4302. doi: 10.1115/1.4048223.

Dean Venter and Steven Dirven. Self morphing soft-robotic gripper for handling and manipulation of delicate produce in horticultural applications. In *2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pages 1–6. IEEE, 2017.

Qiuxuan Wu, Yueqin Gu, Yancheng Li, Botao Zhang, Sergey A Chepinskiy, Jian Wang, Anton A Zhilenkov, Aleksandr Y Krasnov, and Sergei Chernyi. Position control of cable-driven robotic soft arm based on deep reinforcement learning. *Information*, 11(6):310, 2020.

Lingxiao Xun, Gang Zheng, and Alexandre Kruszewski. Cosserat-rod based dynamic modeling of soft slender robot interacting with environment, 2023. URL https://arxiv.org/abs/2307.06261.

Lu Zongxing, Li Wanxin, and Zhang Liping. Research development of soft manipulator: A review. *Advances in Mechanical Engineering*, 12(8):1687814020950094, 2020.