École Polytechnique Fédérale de Lausanne

Section de Microtechnique, 2024-25

Semester Project

Machine Learning Modeling and Control of Dielectric Elastomer Actuators

---

LAI

09.09.2024 - 03.01.2025

Author:
Jad BENABDELKADER
Robotics M.Sc.

Supervisors:
Marc FAVIER
Stefania KONSTANTINIDI

Table of Contents

Table 0.1: List of Abbreviations

Abbreviation	**Definition**
DEA	Dielectric Elastomer Actuator
RL	Reinforcement Learning
PPO	Proximal Policy Optimization
PID	Proportional-Integral-Derivative
LQR	Linear Quadratic Regulator
MDP	Markov Decision Process
SB3	Stable-Baselines3
RNN	Recurrent Neural Network
EMG	Electromyography
RMSE	Root Mean Squared Error
DAQ	Data Acquisition System

1 Introduction

1.1 Context and Motivation

Dielectric Elastomer Actuators (DEAs) are a promising technology in the field of robotics and automation, often referred to as "artificial muscles" [1] due to their ability to emulate the flexibility, adaptability, and performance of biological muscles. DEAs are soft actuators composed of a dielectric elastomer material sandwiched between compliant electrodes. When high voltage is applied across the electrodes, the elastomer contracts in thickness and expands in the plane, producing mechanical motion. This unique mechanism makes DEAs lightweight, highly flexible, and capable of relatively large deformations, which are valuable properties for various applications. As such, the problem of being able to control the elongation along one axis of the plane of a DEA becomes fundamental in many applications using them across different fields such as bio-inspired robots, for instance [2]. The following is a schematic to visually explain the working principle of DEAs.

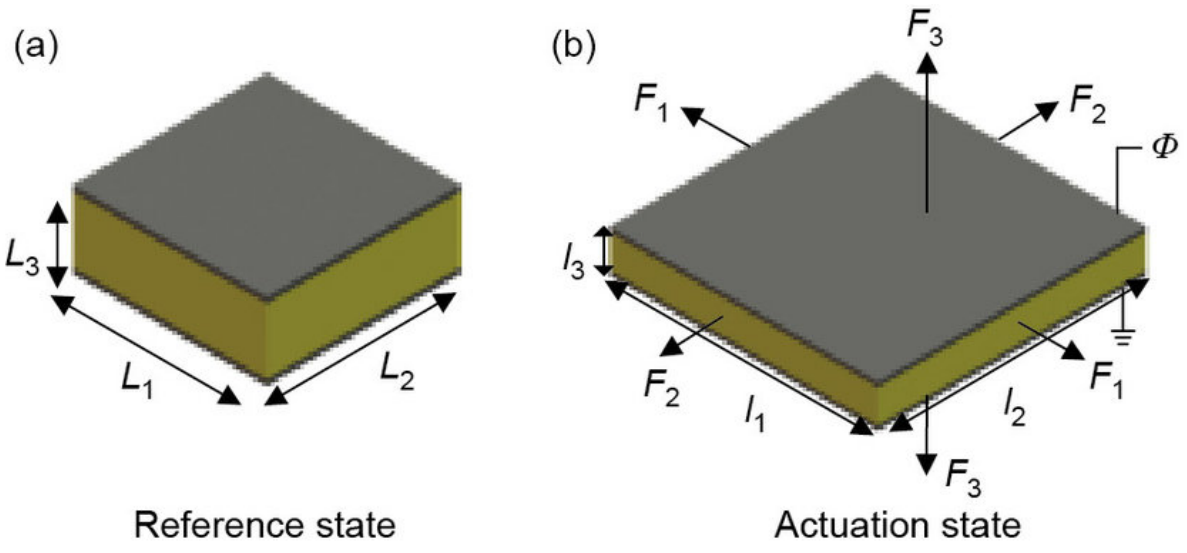

Figure 1.1: Working principle of a Dielectric Elastomer Actuator. [3]

DEA actuation is governed by the Maxwell pressure p induced by the applied voltage V, which generates an electrostatic force across the elastomer. This pressure can be expressed as:

$$p = \varepsilon_0 \varepsilon_r \frac{V^2}{d^2},$$

where:

- ε_0 is the permittivity of free space,
- ε_r is the relative permittivity of the dielectric material,
- d is the thickness of the elastomer layer.

This pressure induces deformation in the elastomer, resulting in a coupling between electrical and mechanical domains. The deformation of the elastomer can be described using strain λ, which represents the ratio of the deformed to the original dimensions. The relationship between the applied voltage and the strain is inherently non-linear and influenced by the material's properties, which are commonly modeled using strain energy density functions such as the Mooney-Rivlin or Ogden models.

In addition to this, DEAs also exhibit viscoelastic effects, where stress σ depends on both strain λ and its rate of change $\dot{\lambda}$. Hence, DEAs also exhibit time-dependent material response, introducing various phenomena to the actuator's behavior, which further complicate control. Despite their advantages, these non-linearities make DEA control a challenging task. For example:

- **Hysteresis:** The actuator exhibits different strain responses for increasing versus decreasing voltages.
- **Electromechanical Instabilities:** At high voltages, the coupling between electrical and mechanical domains can cause instability, where the elastomer collapses unexpectedly.

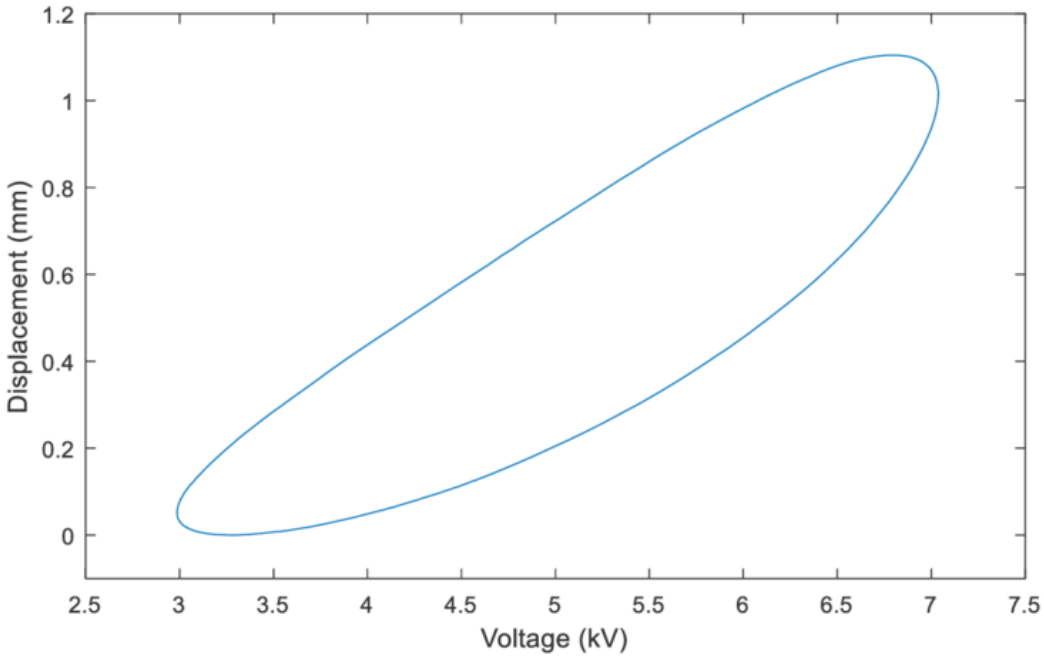

Figure 1.2: Hysteresis property of a Dielectric Elastomer Actuator. [4]

Studies such as Tang et al. (2023) [5] have extensively analyzed these electromechanical coupling effects and their implications for DEA performance and design.

1.2 Problem Statement

As we have seen, despite the significant promise that DEAs offer for soft robotics and other advanced applications, their non-linear and time-dependent behavior presents substantial challenges in terms of control. The inherent non-linearity of DEAs, driven by electrostatic forces, hysteresis, and electromechanical instabilities, complicates the design of conventional control

systems. Furthermore, the lack of a straightforward mathematical model to describe the full dynamics of DEAs introduces additional uncertainty in controlling their performance.

Traditional control strategies, such as PID controllers, are limited in handling these complex dynamics as they often work under the assumption that the system to be controlled is linear and often require extensive knowledge of the system to be properly tuned, which is no trivial task in the case of DEAs. Over recent years, various studies have been conducted to build mathematical models for DEAs. For instance, Wu et al. (2023) [6] have used fractional calculus to model and control DEAs.

To overcome these limitations, there is an increasing interest in using Reinforcement Learning (RL) techniques for controlling DEA actuators in real-time. RL has the potential to learn optimal control policies directly from interaction with the system, without needing an explicit model. This approach could enable DEA control systems that are more flexible, adaptive, and robust, capable of handling the complexities of non-linear behavior and making use of the actuators' full potential in a wide range of applications including :

- **Soft Robotics:** As actuators for robotic grippers, manipulators, and locomotion systems that require compliance and adaptability in unstructured environments [7].
- **Wearable Assistive Devices:** For example, artificial muscles in exoskeletons or prosthetic limbs to restore or enhance human mobility [8].
- **Haptic Feedback Systems:** DEAs can provide tactile sensations in virtual reality or teleoperation systems by emulating the compliance of biological tissues [9].
- **Biomedical Devices:** Such as pumps for drug delivery or actuators in minimally invasive surgical tools [10].

The central problem addressed in this work is the development and application of RL-based control methods to manage DEA actuators in robotic systems. The goal is to design an RL framework that can autonomously learn to control the actuator in real-time, without the need for any explicit model of the system. This includes dealing with the actuator's non-linearities, hysteresis, and the electromechanical coupling effects that traditional control techniques often struggle to manage. This report will serve to present the work that was done throughout the course of this project. First, we will go through the current state-of-the-art in soft actuator control to then outline our own methodology. We will then present our results and end with a discussion around the results obtained.

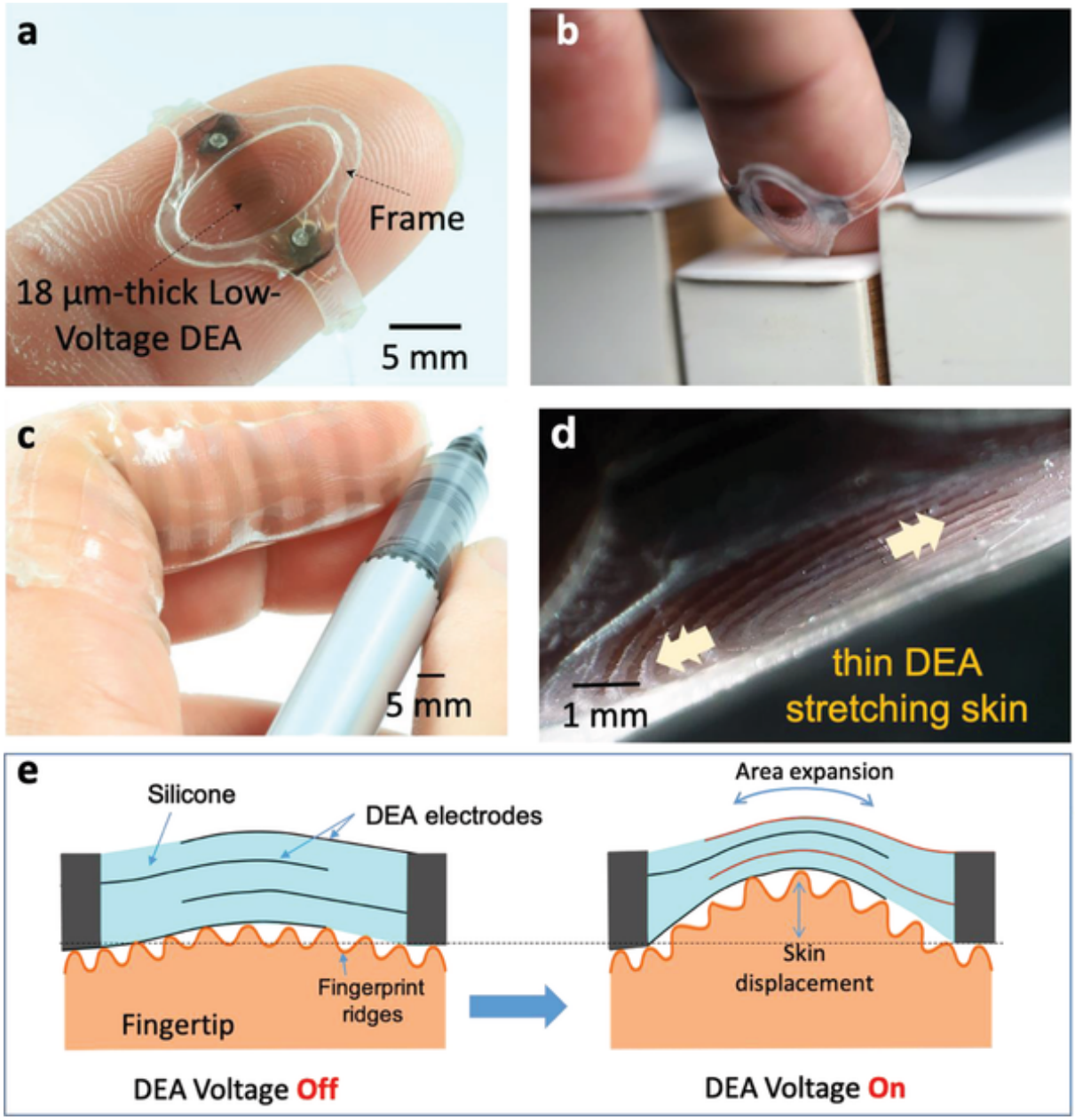

Figure 1.3: Application of DEA as a haptic device [9]

2 State of the Art

Various control methods have been employed to manage DEAs, including traditional (model-based) and learning-based control strategies. These approaches aim to improve performance, reduce errors, and increase reliability. The overwhelming majority of these methods rely on First-level closed-loop control, where a sensor measures the displacement of a single actuator. The error between the current displacement and the target displacement is then fed into a controller which will output appropriate control actions (voltages to be applied to the actuator) to drive the actuator towards its target displacement. Open-loop control methods for DEAs can also be implemented but are much less precise than closed-loop methods as the actual displacement of the actuator is not measured. To control entire robotic systems comprised of multiple actuators, Second-level closed-loop control methods can also be implemented using the kinematic model of the robot.

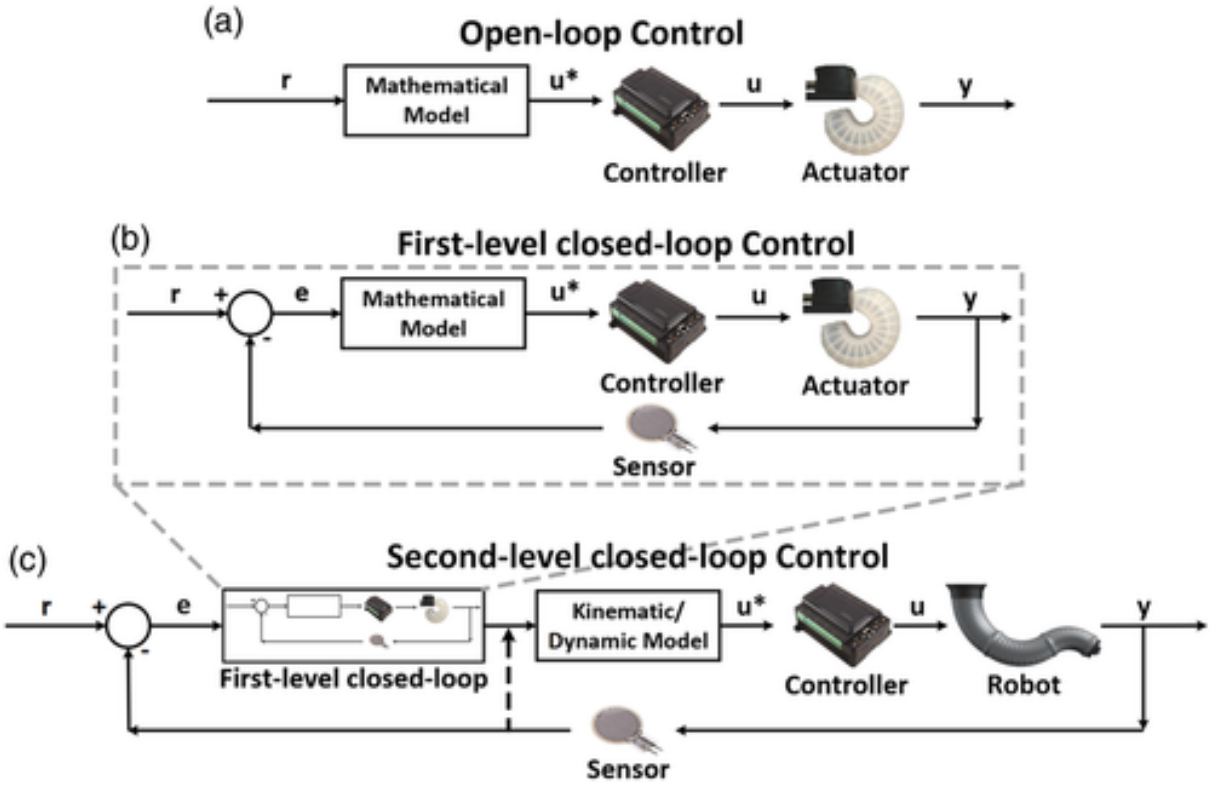

Figure 2.1: Diagram of different kinds of control [11]

In this discussion, we will mainly focus on First-level closed-loop control methods for single actuators.

2.1 Traditional Control Techniques

Traditional control methods such as PID controllers have been widely used in DEA applications. However, more advanced control techniques have emerged to overcome some limitations of PID controllers (they often require model linearization around an operating point to work properly), especially in systems with complex dynamics.

Ye et al. (2017) employed an H_∞ control scheme to control the deformation of a dielectric elastomer actuator with a conic shape with small deformations. This approach showcased better performance than a PID controller, tuned using Matlab, particularly in systems affected by noise and disturbances [12]. Similarly, Branz and Franesconi developed a closed-loop transfer function-based control system to control rotational displacements in a set of antagonistically coupled CDEAs. Their system used dynamic simulations in Ansys to create transfer functions that mapped desired displacements to applied voltages [13].

Zou and Gu focused on minimizing the effects of viscoelasticity in DEAs by integrating a feedforward controller with Zero Vibration Input Shaping, creating a creep compensator for the actuator. The controller used experimentally derived transfer functions to optimize control performance [14]. In another study, Gu et al. applied feedforward control to minimize vibrations in a DEA system with a central oscillating mass [15].

Zhao et al. introduced an active vibration control system using a filtered-x Least Mean Square algorithm. This system reduced vibrations in a conical DEA system, where a mass was attached to the center of the DE membrane, by adapting the control to the system's dynamic properties [16]. Mulembo et al. designed an optimal control strategy using Linear Quadratic Regulator (LQR) combined with a Luenberger state observer to achieve precise micro-metric displacements in DEAs. The LQR controller successfully mitigated errors caused by the nonlinearities and viscoelasticity of the materials involved [17].

All of the approaches discussed here provide convincing results, but lack flexibility and adaptability as explicit mathematical models have to be built to represent the actuator. Simply replacing the DEA used with another one could lead to significantly worse performance as the actuators are not perfectly identical : Replacing the actuator with a newer one might lead to model redesign and retuning of the controller. To address these challenges, modern control techniques have been increasingly applied to soft actuator systems in general and DEAs in particular

2.2 RL-based Control

RL is a subfield of machine learning that provides a framework for decision-making and control in dynamic environments. In RL, an agent learns to make decisions by interacting with its environment, with the aim of maximizing a numerical reward signal. The agent observes the state of the environment, selects actions based on a policy, and transitions to a new state while receiving a reward. Over time, the agent improves its policy to perform better in achieving its goal.

2.2.1 General Principles of Reinforcement Learning

The core components of RL include:

- **Agent**: The entity that makes decisions.
- **Environment**: The external system with which the agent interacts.
- **State** (s_t): A representation of the environment at time t.
- **Action** (a_t): A decision taken by the agent at time t.
- **Reward** (r_t): Feedback signal that indicates the quality of the action taken.
- **Policy** ($\pi(a|s)$): A strategy that maps states to actions, either deterministically or stochastically.
- **Value Function**: Measures the expected cumulative reward, helping the agent evaluate states or state-action pairs.

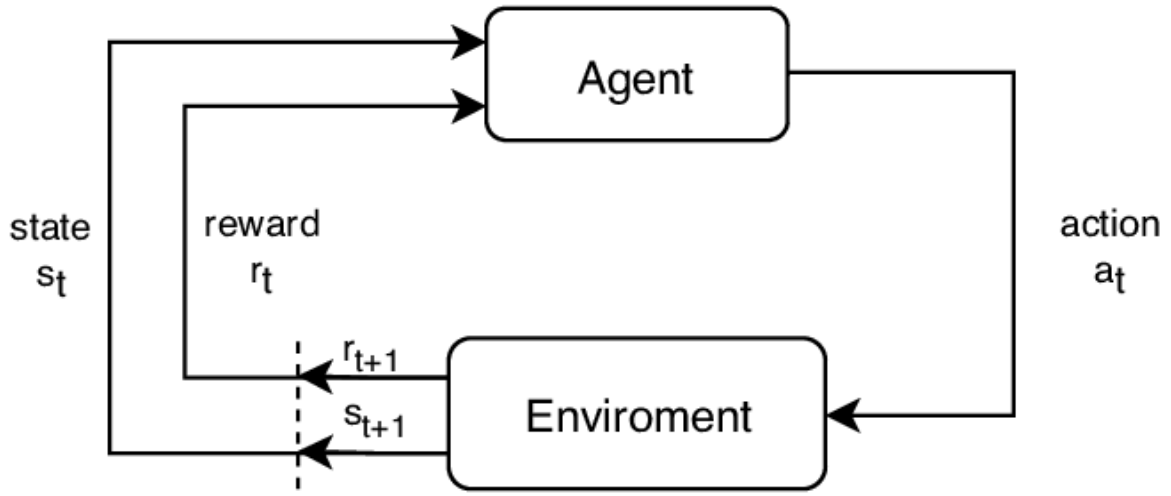

Figure 2.2: Simplified Diagram of an RL model [18]

The learning process of the RL agent happens through a training procedure. During training, the weights of the neural networks that make up the model are updated at each epoch to minimize the loss functions of the networks. The agent's goal is to learn a policy that maximizes the expected cumulative reward over time, typically expressed as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k},$$

where G_t is the return, and $\gamma \in [0, 1]$ is the discount factor that determines the weight of future rewards. In practice, the length of the time-horizon used is generally not infinite as it is simply

set to be the length of an episode which is just a finite sequence of steps in the environment.

2.2.2 Applications of Reinforcement Learning in Control

RL has shown significant success in robotic control and dynamic systems due to its ability to handle complex, high-dimensional, and nonlinear environments. Some notable applications include:

- **Robotics**: RL is used to teach robots tasks such as grasping, locomotion, and manipulation, even in unstructured environments [19].
- **Autonomous Vehicles**: Control systems for path planning, collision avoidance, and adaptive cruise control leverage RL techniques [20].
- **Energy Management**: Optimization of energy distribution in power grids and battery management systems [21].
- **Industrial Automation**: Control of manufacturing processes, including robotic arms and soft actuators [22].

These applications highlight RL's ability to learn optimal control strategies without requiring a precise model of the system, making it especially powerful in scenarios where traditional control methods struggle.

2.2.3 Common RL Algorithms for Continuous Control

In this report, for the sake of brevity, we will focus solely on actor-critic methods. All of these methods natively handle continuous action and state spaces, which is not the case for all RL methods as some will rely on discretization. These methods also rely on two separate entities generally represented as neural networks : the actor and the critic. The role of the actor is to determine and deploy the policy to be followed by the agent, while the critic will estimate the expected cumulative reward starting from the state where the agent currently is which will help to stabilize training.

Several RL algorithms have been developed for continuous control tasks, each with unique characteristics

- **Deep Deterministic Policy Gradient (DDPG)**: Combines the actor-critic architecture with deterministic policies. It uses a target network and experience replay to stabilize learning. DDPG is effective for high-dimensional action spaces but is sensitive to hyperparameter tuning [24].
- **Soft Actor-Critic (SAC)**: Builds on DDPG but incorporates entropy maximization into the objective, encouraging exploration and robustness. SAC is well-suited for tasks requiring high stability and efficiency [25].
- **Proximal Policy Optimization (PPO)**: A popular on-policy method that improves stability by limiting the extent of policy updates. PPO optimizes a surrogate objective that constrains the policy update, preventing drastic changes. [26].

More recently, reinforcement learning methods that leverage world-model representations have emerged as powerful tools for continuous control tasks. For instance, **DreamerV3** [27] is built on the concept of learning compact, latent representations of the environment dynamics.

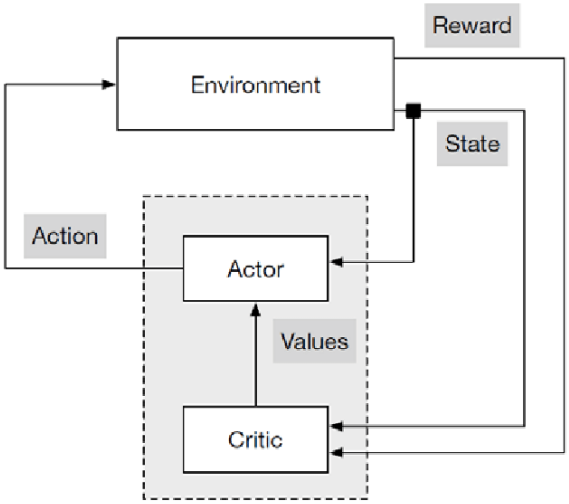

Figure 2.3: The Actor-Critic structure [23]

By predicting future states in a latent space, DreamerV3 effectively reduces the reliance on high-dimensional observations. It uses these learned representations to perform planning and control, enabling sample-efficient learning and strong generalization which makes it particularly well-suited for tasks where interaction with the real environment is costly or limited.

2.2.4 Proximal Policy Optimization (PPO)

PPO is a policy-gradient method that balances exploration and exploitation through clipped surrogate objectives. It addresses the instability issues of earlier policy-gradient algorithms like Trust Region Policy Optimization (TRPO) while maintaining simplicity and computational efficiency.

The key idea in PPO is to optimize a clipped surrogate objective to update the policy. The objective function is defined as:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\Big(r_t(\theta)\widehat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_t\Big)\Big],$$

where:

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio of the new and old policies.
- $\widehat{A}_t$ is the advantage estimate, which measures how much better an action is compared to the average action.
- ϵ is a hyperparameter that controls the clipping range.

The clipping ensures that policy updates do not diverge significantly from the previous policy, promoting stable learning.

The algorithm alternates between collecting trajectories using the current policy and optimizing the policy and value function based on the collected data.

2.2.5 Advantages of PPO

- Simplicity and computational efficiency.
- Robustness to hyperparameters.
- Good performance across a wide range of continuous control tasks.

In this report, PPO will be used to control a soft electro-hydraulic actuator due to its robustness and ability to handle continuous action spaces effectively.

2.3 Related Work

The integration of RL into soft actuator control has been explored in various contexts, showcasing its potential to address the challenges associated with non-linear and time-dependent behavior of soft actuators. In this section, we highlight two notable studies closely related to the control of soft actuators.

Model-Based Reinforcement Learning for Soft Manipulators

Thuruthel et al. [28] proposed a model-based reinforcement learning approach to develop closed-loop control policies for soft robotic manipulators. Their method used a recurrent neural network to model the forward dynamics of the manipulator, enabling trajectory optimization and policy learning. The resulting control policy was validated on both simulated and experimental setups, demonstrating robustness to unmodeled external loads and varying control frequencies. This work is particularly relevant to our project as it highlights the effectiveness of RL in managing the complexities of soft actuator dynamics.

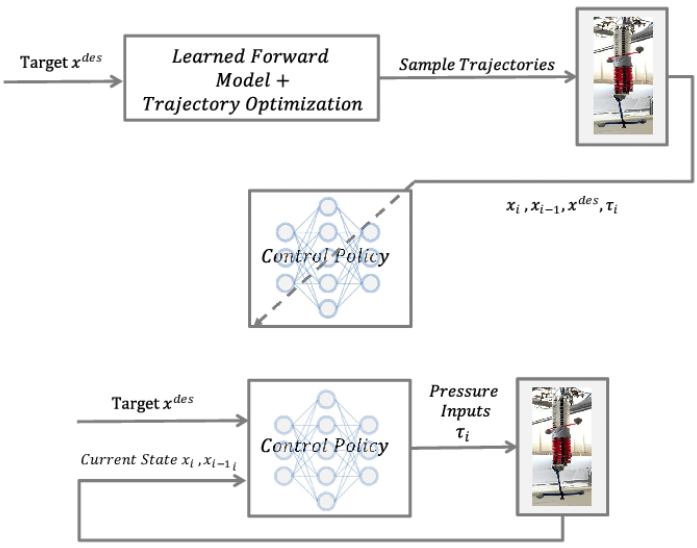

Figure 2.4: Diagram describing the complete procedure for obtaining the closed loop policy [28]

Soft Artificial Muscle-Driven Robots with RL Optimization

Yang et al. [29] presented a reinforcement learning framework for controlling a soft robot driven by DEAs. Inspired by the propulsion mechanism of cuttlefish, the study showcased the use of RL to optimize the actuation strategy, achieving a significant improvement in the robot's swimming speed. This work underscores the potential of RL in enhancing the performance of soft robotic systems, making it a strong foundation for DEA-based applications.

Both of these studies emphasize the growing role of RL in soft robotics, particularly in addressing the challenges posed by non-linear actuation mechanisms. However, they are adressed towards solving the problem of controlling an entire robotics system made of multiple actuators, and not a single actuator (They use Second-Level Closed Loop control). Furthermore, they do not adress the specific case of DEAs who have their own specific and unique challenges.

Deep Reinforcement Learning for DEA Control

Li et al. [30] proposed a model-free method for controlling DEAs based on deep RL, more specifically Deep Q-Learning. Their approach addressed the challenges posed by the viscoelastic and nonlinear electromechanical coupling behavior of DEAs.

The study demonstrated dynamic feedback control by considering the time-dependent behavior of DEAs. The proposed method formulates DEA control as a time-continuous problem and learns a nonlinear control policy capable of addressing viscoelasticity effects. Experimental results validated the robustness of the method across DEAs with different configurations, prestretches, and material property changes over time. Key findings included:

- The RL model was able to achieve accurate control for a variety of target displacement trajectories, including sine waves, triangle waves, and square waves.
- The proposed approach outperformed a baseline proportional-integral (PI) controller that was specifically tuned on select test-trajectories.
- The RL-based method exhibited strong generalization, effectively handling unseen test trajectories and adapting to changes in material properties or actuator structures.

This study is closely related to our work, as it highlights the potential of RL for addressing the challenges associated with controlling DEAs. However, the proposed controller is not entirely built using RL, as the estimated control voltage ($\widehat{v}_t$) is computed by applying multiple step voltages of varying magnitudes to the actuator and recording the resulting displacement sequences. For each timestep t, a fourth-order polynomial $P_t(d)$ is fitted to map displacement to voltage based on the experimental data. The target displacement at $t+1$ is then substituted into $P_t(d)$ to calculate $\widehat{v}_t$. Deep Q-Learning is then only used to predict an offset selected from a small set of discrete offsets to be added to ($\widehat{v}_t$). To the best of our knowledge, our study is the first one to propose an entirely RL-based controller to regulate the planar elongation of DEAs.

Building on the advances presented throughout this section, our project aims to further investigate RL-based control frameworks specifically tailored for dielectric elastomer actuators, exploring their potential for applications in robotics and automation.

3 Methodology

3.1 Problem Formulation

We will formulate the task of controlling a DEA as an RL problem, where the goal is to minimize the error between the target displacement and the achieved displacement while ensuring smooth and efficient control. At each step in the environment, the agent will measure the displacement of the actuator and compare it to the target displacement. From there, a reward will be computed and an action defined by the actor network will be chosen. During training, the agent will be subject to a series of episodes of finite length. At each episode, the actuator must reach and stabilize around its target displacement.

3.1.1 Reward Function

The reward function is designed to encourage accurate displacement tracking while providing a stabilization bonus for achieving precise control. At each timestep t, the reward r_t is defined as:

$$r_t = -|\text{error}_t| + b_t,$$

where:

- $\text{error}_t = y_t - d_t$ is the displacement error, with y_t being the target displacement and d_t the achieved displacement.
- b_t is a stabilization bonus, defined as:

$$b_t = \begin{cases} 1, & \text{if } |\text{error}_t| \leq 0.1, \\ 0, & \text{otherwise.} \end{cases}$$

This reward function penalizes displacement error through the term $-|\text{error}_t|$, encouraging the agent to minimize the absolute difference between the target and current displacements. Additionally, the stabilization bonus b_t rewards the agent for maintaining the displacement within a small tolerance (± 0.1), promoting precision and steady-state accuracy.

3.1.2 State Representation

The state s_t at each timestep t is represented as:

$$s_t = \{d_t, v_t, y_t\},$$

where:

- d_t: The current displacement of the actuator.

- v_{t-1}: The current speed of the actuator.
- y_t: The target displacement at the current timestep.

This representation captures both the actuator's current state and some of the dynamics of the target trajectory, ensuring the agent has sufficient information for effective decision-making. Each component of the state vector will systematically be normalized to the range $[-1, 1]$ as this ensures that all features of the state vector have similar range, improving stability.

3.1.3 Action Space

The action a_t represents the voltage adjustment applied to the actuator. It is defined as a continuous value normalized between -1 and 1, corresponding to the range of permissible voltage changes:

$$a_t = \text{scale}(\Delta v_t), \quad \Delta v_t \in [v_{\min}, v_{\max}],$$

where scale$(\cdot)$ maps $[v_{\min}, v_{\max}]$ to $[-1, 1]$. This normalization is done for the same reasons as the normalization of the state vector.

3.2 Experimental Setup

To develop, train, and test the RL model for controlling DEAs, several tools and experimental setups were utilized. These setups were designed to validate the model both in simulation and on physical hardware.

Tools and Software Framework

The entire project was programmed using the Python programming language as it provides some of the most powerful RL frameworks and libraries available. The following tools were chosen to build the experimental framework:

- **Gymnasium**: A Python library used to define the custom RL environment, ensuring compatibility with standard RL workflows.
- **PyTorch**: A machine learning framework used for defining and training the neural networks that constitute the RL agent.
- **Stable-Baselines3 (SB3)**: A library providing a robust and modular implementation of the PPO algorithm, enabling efficient training and testing and easy hyperparameter tuning.

Experimental Setups

Three different experimental setups were designed for training and evaluating the RL model:

1. Simulated Linear Actuator In the first setup, a simulation of a linear actuator was developed. This simulation served as a controlled environment for initial training, allowing the RL agent to learn basic control policies without the complexities of real-world dynamics.

The speed and position of the actuator are updated at each timestep according to the following equations:

$$v_t = (a_t \cdot 0.1) - (c \cdot v_t),$$

$$d_t = d_t + v_t \cdot \Delta t,$$

where:

- v_t: The speed of the actuator at timestep t.
- a_t: The applied voltage (the action determined by the RL model) at timestep t.
- $c = 0.1$: The compliance coefficient, which introduces damping.
- d_t: The displacement of the actuator at timestep t.
- $\Delta t = 0.1$: The time step of the simulation.

2. Physical Experiment: Actuator Driving a Moving Platform In the second setup, the DEA was clipped to a fixed base and connected to a moving platform with a mass on top. When voltage was applied to the actuator, it drove the platform, causing vertical movement. A laser sensor was positioned beneath the platform to measure its displacement during operation.

3. Physical Experiment: Free Actuator with a Moving Tip In the third setup, the actuator was attached to a fixed cranium and left free to elongate when voltage was applied. A small piece of paper was attached to the end of the actuator to act as a marker, moving with the actuator's elongation. A laser sensor tracked the marker's movement. This setup will allow different dynamics compared to the Platform setup as it introduces friction between the cranium and the actuator.

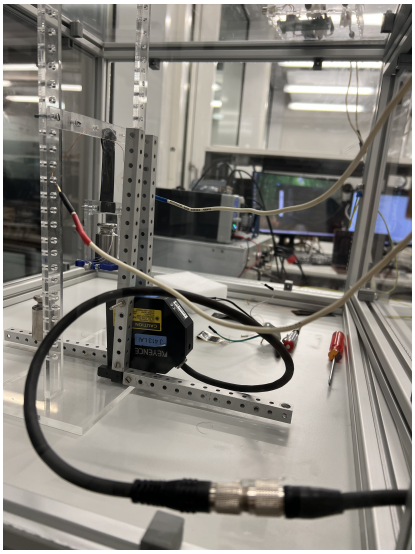

Figure 3.1: Moving Platform Set-Up

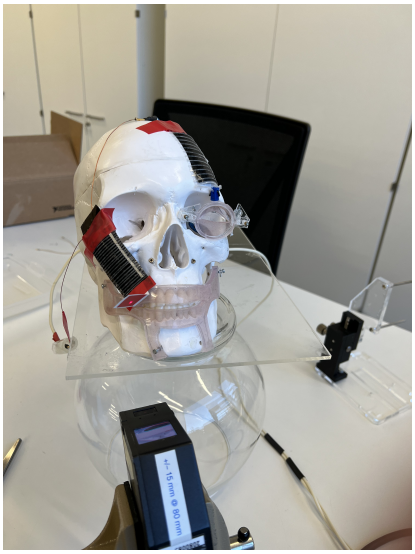

Figure 3.2: Cranium Set-Up

Data Flow

For both physical setups:

- Voltage control signals were generated by the RL model running on a computer and then written to the DAQ.
- The DAQ passed these signals through a voltage amplifier with a $\times 2000$ gain.
- Amplified voltages were applied to the actuator.
- Feedback data, such as displacement measurements from the laser sensor, was read back into the computer via the DAQ.

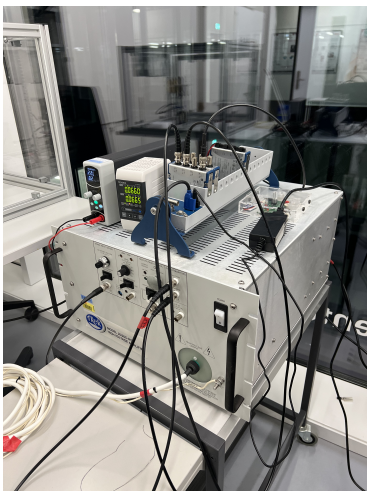

Figure 3.3: DAQ and Amplifier

These setups provide a comprehensive framework for testing the RL-based control strategy, ensuring the model is evaluated under both simulated and real-world conditions.

3.3 Training Details

The training of the reinforcement learning model was conducted using the default settings of the SB3 implementation of PPO. The important hyperparameters and the network structure (same structure for both actor and critic) are presented in Table 3.1.

Parameter	**Value**
Learning Rate	3×10^{-4}
Discount Factor (γ)	0.99
Clipping Parameter (ϵ)	0.2
Batch Size	64
Entropy Coefficient	0.0
Number of Epochs per Update	10
Number of Hidden Layers	2
Number of Neurons per Hidden Layer	64
Activation Function	`tanh`

Table 3.1: Hyperparameters used for PPO

The training process was conducted over a fixed number of timesteps. At each timestep, the agent interacted with the environment to collect trajectories, which were used to update the policy and value networks using the PPO optimization algorithm. The clipping parameter (ϵ) ensured stability during training by limiting policy updates, and the discount factor (γ) determined the agent's weighting of future rewards.

4 Results and Discussion

4.1 Simulation

For the simulated linear actuator, training lasted 500,000 time steps, with episodes consisting of 256 time steps each. During each episode, the initial displacement of the actuator was chosen randomly, and it was tasked with reaching a single randomly chosen target. The actuator was driven by the actions chosen by the actor network at each time step to reach this target.

At the end of training, the mean cumulative reward per episode was visualized to evaluate whether training had converged. These plots were generated using TensorBoard, which is directly supported by the SB3 library.

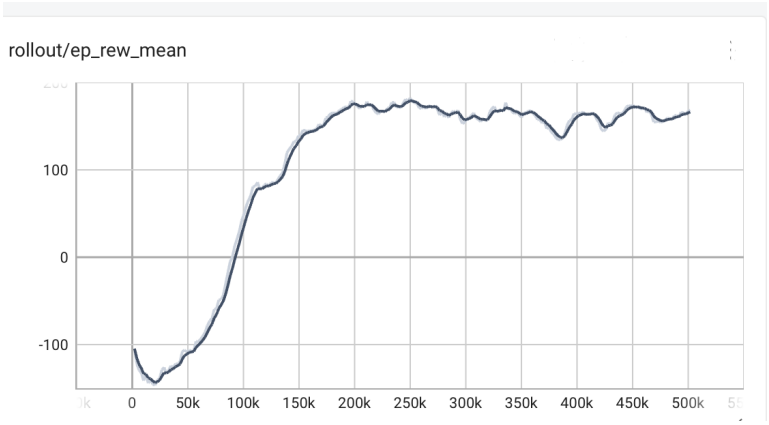

Figure 4.1: Mean cumulative reward over time during simulation training.

As shown in Figure 4.1, the reward starts off very low at the beginning of training due to the random initialization of the neural network weights. As training progresses, the policy network learns to maximize its reward by selecting appropriate actions, resulting in a steady increase in the mean reward. Over time, the reward stabilizes around a high value with minor oscillations, indicating successful convergence.

For testing, the actuator's inherently slow dynamics (to simplify the simulation for the RL model) made completing full trajectories within the episode length infeasible. Testing instead focused on a single target that the actuator must reach, serving as a proof-of-concept to validate the RL controller's robustness before moving on to physical setups. The Root Mean-Squared Error (RMSE) was used to evaluate performance.

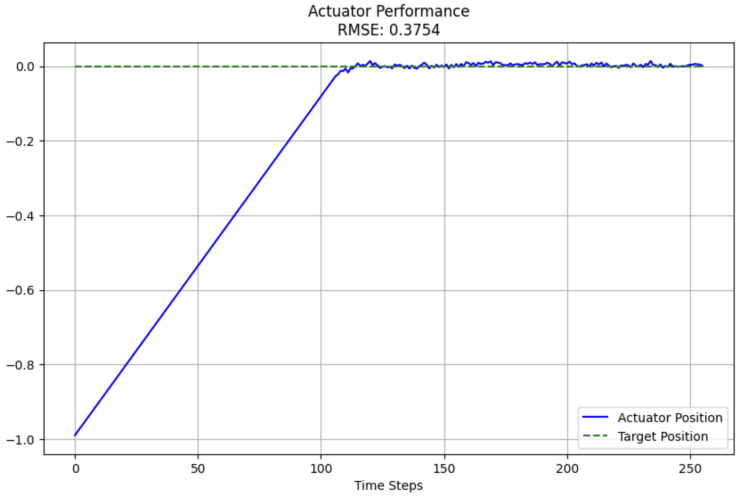

Figure 4.2: Simulated test trajectory of the actuator reaching a single target.

Figure 4.2 shows that the actuator moves towards the target and stabilizes around it. The static RMSE, calculated after the actuator has settled, was approximately 0.07, while the total RMSE was 0.37 due to the slow response dynamics.

4.2 Moving Platform Setup

For the Moving Platform Setup, training lasted 170,000 time steps, with episodes consisting of 256 time steps each. After each episode, the actuator was returned to its resting displacement by adding a 2-second delay in the environment's `reset` method. During training, the agent interacted with four random targets per episode (64 time steps per target), enabling it to learn complete trajectories rather than a single target. The entire training process lasted approximately 3.2 hours on the test machine, with a training and testing frequency of 13Hz.

As with the simulation, the mean cumulative reward per episode was plotted over time.

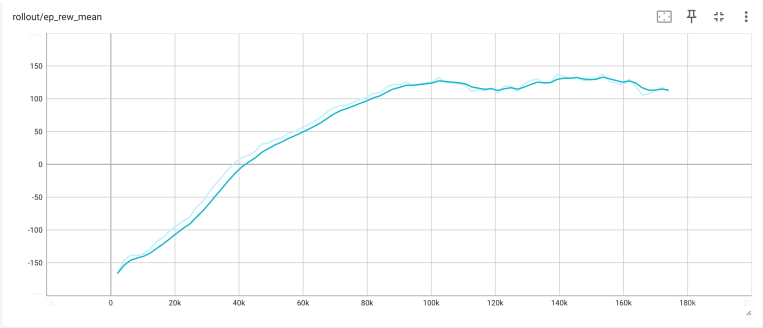

Figure 4.3: Mean cumulative reward over time for the Moving Platform Setup.

Figure 4.3 shows reward progression similar to the simulation results, with an initially low reward

that increases steadily as the RL agent learns. Training eventually stabilizes, indicating successful convergence of the policy network in the physical setup.

For testing, the controller was evaluated using two distinct trajectories: a step trajectory and a sawtooth trajectory. These trajectories were designed to assess the controller's performance under varying dynamic conditions.

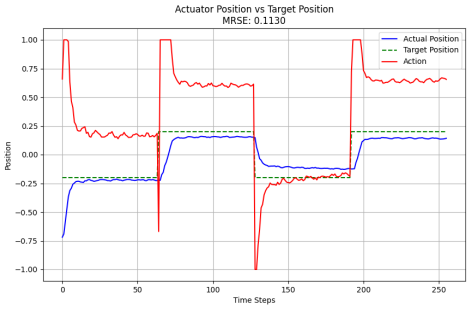

Figure 4.4: Step Trajectory.

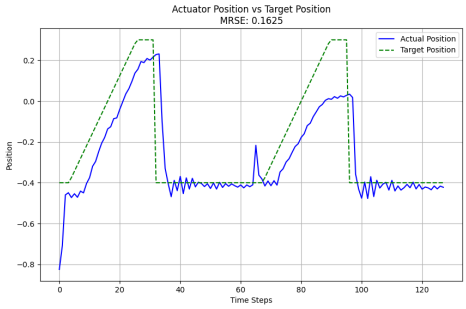

Figure 4.5: Sawtooth Trajectory.

Step Trajectory Results Discussion

For the step trajectory, the RMSE was approximately 0.11, with an average settling time of 1.6 seconds. While acceptable, the actuator's displacement did not settle as close to the target as desired. This behavior likely stems from the stabilization bonus in the reward function, which introduces an "inertia" effect. Once the actuator reaches the stabilization zone, it is no longer pressured to move closer to the target.

Additionally, the voltage strategy (depicted in red in the figure) was aggressive, with the controller frequently saturating the voltages toward extreme values for each step. This suggests that the controller applies large voltage steps rather than proportional adjustments.

Sawtooth Trajectory Results Discussion

For the sawtooth trajectory, performance degradation was observed, with the RMSE increasing to 0.16. This degradation occurred across all smooth trajectories tested. The likely cause was the same "inertia" effect observed in the step trajectory results.

The actuator initially moved close enough to the target to enter the stabilization zone but began to lag as the target continued to move smoothly. This delay impacted the controller's ability to track the trajectory effectively, leading to increased errors.

4.3 Cranium Setup

For the Cranium Setup, the RL training methodology was identical to the Moving Platform Setup, with the same training length, episode structure, and frequency. As in the previous setups,

the mean cumulative reward per episode was tracked to ensure convergence.

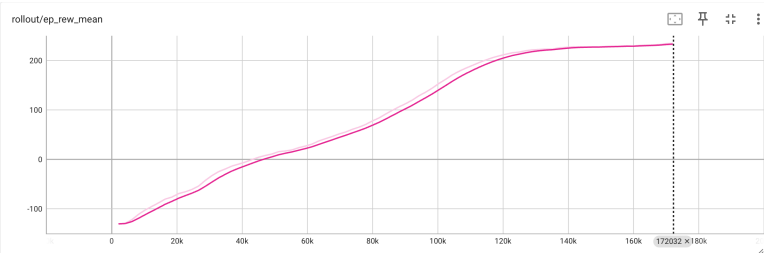

Figure 4.6: Mean cumulative reward over time for the Cranium Setup.

Figure 4.6 shows a progression similar to earlier setups, with rewards stabilizing over time, indicating that the RL agent successfully adapted to the cranium environment's specific constraints.

For testing, the RL controller was evaluated using two trajectories: the step trajectory and a trajectory derived from EMG signals of a healthy person pronouncing "hello." This signal was repeated twice over a 20-second duration.

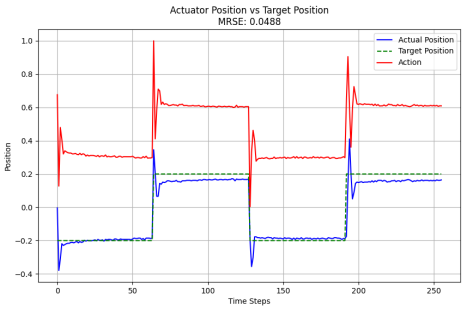

Figure 4.7: Step Trajectory for Cranium Setup.

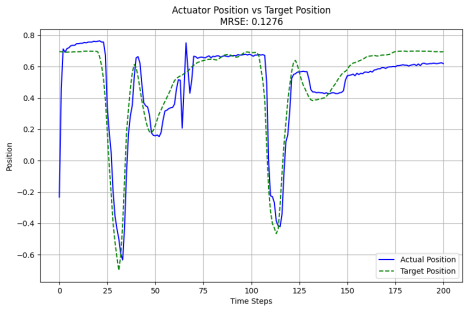

Figure 4.8: "Hello" Trajectory for Cranium Setup.

Step Trajectory Testing Discussion

In the step trajectory (Figure 4.7), the actuator achieved better performance than before, with an RMSE of 0.05 and an average settling time of 0.4 seconds. The voltage strategy was also less aggressive, with fewer extreme voltage saturations compared to the Moving Platform Setup.

"Hello" Trajectory Testing Discussion

In Figure 4.8, it is very noticeable that the tracking performance on smooth trajectories has improved significantly, as this is typically the kind of trajectories where tracking would have completely failed on the previous setup. The RMSE is around 0.13 and the overall tracking performance is satisfactory.

It is quite clear that this setup favors the RL controller quite significantly more than the previous one. This is likely due to the fact that the oscillations introduced by the movement of the platform and amplified by the weight are not present in this set-up. RL, due to the underlying assumption that the interactions with the environment can be depicted as simple Markov Decision Processes (MDPs) where the next state only depends on the current state and the chosen action, tends to struggle with oscillations as they often break this assumption. It is possible that simply increasing the frequency of the control loop (13Hz currently) could lead to better performance on the Moving Platform setup as this would allow us to better capture the effects of oscillations.

4.4 Comparison with PID Control

Testing data for the PID controller on the Cranium Setup was provided by Stefania Konstantinidi, who linearized a new mathematical model of the DEA to develop the PID controller. Two runs, each lasting 3.30 seconds with one repetition of the "hello" signal, were conducted.

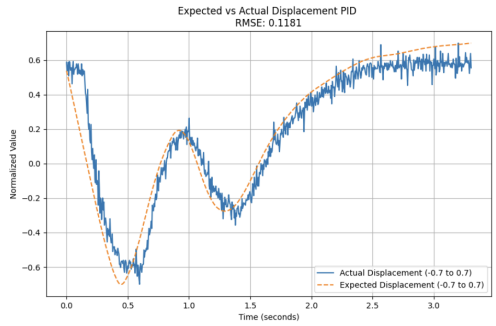

Figure 4.9: First PID testing run.

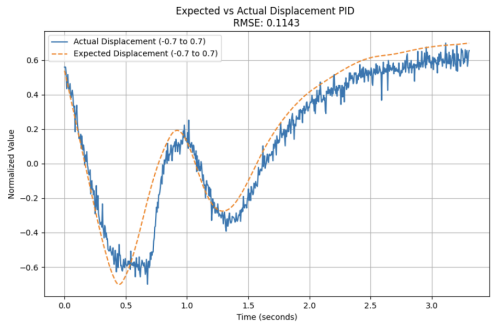

Figure 4.10: Second PID testing run.

Controller	**Trajectory**	**RMSE**
PID (Run 1)	"Hello" Signal (1 Rep.)	0.118
PID (Run 2)	"Hello" Signal (1 Rep.)	0.114
RL	"Hello" Signal (2 Reps.)	0.128

Table 4.1: Comparison of RMSE values for PID and RL controllers.

As we can see, the first PID testing run has yielded an RMSE of 0.118 while the second run yielded an RMSE of 0.114. The RMSE performance is slightly better than what we have obtained with our RL controller with an RMSE of about 0.128. However, it is important to note that the RL controller was tested on a longer period of time using two repetitions of the "hello" signal

versus only one for the PID controller. At the time of writing this report, we do not have access to data comparing the PID and RL controller in the exact same testing configuration.

Nonetheless, the RL based controller still achieves very comparable performance to the PID controller without any need for explicit modeling or extensive tuning. There are also many improvement paths that still can be taken to further improve the RL controller as we will see in the next section.

5 Outlook and Conclusion

5.1 Outlook

The results obtained in this project demonstrate the potential of RL for controlling highly non-linear actuators DEAs. However, there are several avenues for improvement to enhance the performance and robustness of the RL-based controller:

- **Smoothing Voltage Outputs:** To reduce aggressive voltage fluctuations and avoid the voltage saturation phenomenon discussed previously, incorporating filtering techniques, such as a moving average, could smooth the applied voltages and improve control stability. Another option would be to introduce a voltage smoothness penalty directly into the reward function.
- **Refining the Stabilization Zone:** Reducing the size of the attraction region around the setpoint in the reward function could minimize the observed "inertia" effect, encouraging the actuator to move closer to the target.
- **Learning Rate Adjustments:** Experimenting with lower learning rates might help the RL model converge more gradually and reduce overshooting or oscillatory behaviors. Indeed, it is possible that the RL model falls into a local optima during training. This is supported by the fact that the reward curves during training on both physical set-ups exhibit very high slopes from the beginning, which could be an indication that the agent quickly finds a "good enough" policy and then simply starts to iterate on that policy to improve it further. Reducing the learning rate could lead to more exploration of the environment and its reward dynamics which could lead the model to ultimately settle on a more efficient policy.
- **Introducing Recurrent Architectures:** Incorporating recurrent neural networks (RNNs) could help the model capture temporal dependencies in the actuator's dynamics, particularly in cases where oscillations impact performance such as the Moving Platform setup.
- **Exploring Alternative Models:** More advanced RL models, such as DreamerV3, which leverage world-model representations, could provide better sample efficiency and adaptability for controlling DEAs.

Future work should focus on systematically testing these improvement paths to determine their efficacy in addressing the current limitations of the RL controller. Additionally, expanding the experimental setups to include more complex multi-actuator systems could help evaluate scalability and generalization.

5.2 Conclusion

This project explored the application of RL for controlling DEAs, achieving promising results in both simulated and real-world environments. Initially, a linear actuator was simulated to validate the RL framework. Subsequently, the model was trained and tested on two distinct

physical setups: the Moving Platform and the Cranium setup.

Key observations from the results include:

- The RL controller successfully learned to control DEAs in both experimental setups, with better performance observed in the Cranium setup which is likely due to the absence of oscillatory dynamics.
- Performance on both smooth and abrupt trajectories improved significantly in the Cranium setup, with a reduction in settling time and RMSE.
- A comparison with a PID controller showed that the RL-based approach achieved comparable accuracy without requiring explicit modeling of the actuator dynamics.

Overall, while the RL controller demonstrated its potential for controlling DEAs, further optimization is required to enhance its stability, smoothness, and generalization capabilities. This work serves as a strong foundation for future research, paving the way for more advanced RL applications in soft robotics and actuator control.

References

[1] Yuhao Wang, Xuzhi Ma, Yingjie Jiang, Wenpeng Zang, Pengfei Cao, Ming Tian, Nanying Ning, and Liqun Zhang. Dielectric elastomer actuators for artificial muscles: A comprehensive review of soft robot explorations. *Resources Chemicals and Materials*, 1(3):308–324, 2022.

[2] M. Franke, A. Ehrenhofer, S. Lahiri, E.-F. M. Henke, T. Wallmersperger, and A. Richter. Dielectric elastomer actuator driven soft robotic structures with bioinspired skeletal and muscular reinforcement. *Frontiers in Robotics and AI*, 7, 2020.

[3] N.F. Wang, Cui Chaoyu, Hao Guo, Bicheng Chen, and Xianmin Zhang. Advances in dielectric elastomer actuation technology. *Science China Technological Sciences*, 61, 11 2017.

[4] Lili Meng, Fucai Li, and Hong-Guang Li. Electromechanical hysteresis model and identification for soft dielectric elastomer actuator. 03 2019.

[5] Chao Tang, Boyuan Du, Songwen Jiang, Zhong Wang, Xin-Jun Liu, and Huichan Zhao. A review on high-frequency dielectric elastomer actuators: Materials, dynamics, and applications. *Advanced Intelligent Systems*, 6, 07 2023.

[6] Jundong Wu, Zhichao Xu, Yue Zhang, Chun-Yi Su, and Yawu Wang. Modeling and tracking control of dielectric elastomer actuators based on fractional calculus. *ISA Transactions*, 138:687–695, 2023.

[7] Markus Franke, Adrian Ehrenhofer, Santanu Lahiri, Ernst-Friedrich Markus Vorrath, T. Wallmersperger, and Andreas Richter. Dielectric elastomer actuator driven soft robotic structures with bioinspired skeletal and muscular reinforcement. *Frontiers in Robotics and AI*, 7, 12 2020.

[8] Myles Lidka, Aaron D. Price, and Ana Luisa Trejos. Development and evaluation of dielectric elastomer actuators for assistive wearable devices. In *2018 IEEE Canadian Conference on Electrical Computer Engineering (CCECE)*, pages 1–4, 2018.

[9] Xiaobin Ji, Xinchang Liu, Vito Cacucciolo, Yoan Civet, Alae El Haitami, Sophie Cantin, Yves Perriard, and Herbert Shea. Untethered feel-through haptics using 18-µm thick dielectric elastomer actuators. *Advanced Functional Materials*, 31:2006639, 10 2020.

[10] John Bashkin, Roy Kornbluh, Harsha Prahlad, and Annjoe Wong-Foy. *Biomedical Applications of Dielectric Elastomer Actuators*, pages 395 – 410. 04 2009.

[11] Jue Wang and Alex Chortos. Control strategies for soft robot systems. *Advanced Intelligent Systems*, 4(5):2100165, 2022.

[12] Zhihang Ye, Zheng Chen, Ramazan Asmatulu, and Hoyin Chan. Robust control of dielectric elastomer diaphragm actuator for human pulse signal tracking. *Smart Materials and Structures*, 26(8):085043, jul 2017.

[13] F Branz and A Francesconi. Modelling and control of double-cone dielectric elastomer actuator. *Smart Materials and Structures*, 25(9):095040, aug 2016.

[14] Jiang Zou, Guo-Ying Gu, and Li-Min Zhu. Open-loop control of creep and vibration in dielectric elastomer actuators with phenomenological models. *IEEE/ASME Transactions on Mechatronics*, 22(1):51–58, 2017.

[15] Guo-Ying Gu, Ujjaval Gupta, Jian Zhu, Li-Min Zhu, and Xiang-Yang Zhu. Feedforward

deformation control of a dielectric elastomer actuator based on a nonlinear dynamic model. *Applied Physics Letters*, 107(4):042907, 07 2015.

[16] Yunhua Zhao, Qiwei Guo, Song Wu, Guang Meng, and Wenming Zhang. Design and experimental validation of an annular dielectric elastomer actuator for active vibration isolation. *Mechanical Systems and Signal Processing*, 134:106367, 2019.

[17] Titus Mulembo, Waweru Njeri, Gakuji Nagai, Hirohisa Tamagawa, Keishi Naito, Takahiro Nitta, and Minoru Sasaki. Linear-quadratic regulator for control of multi-wall carbon nanotube/polydimethylsiloxane based conical dielectric elastomer actuators. *Actuators*, 9(1), 2020.

[18] Mateus Mota, Daniel Araújo, Hugo Costa, André de Almeida, and F. Rodrigo Cavalcanti. Adaptive modulation and coding based on reinforcement learning for 5g networks. 11 2019.

[19] Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes, 2024.

[20] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey, 2021.

[21] Ehsan Hosseini, Pablo García-Triviño, Raul Sarrias-Mena, Carlos García Vázquez, and Luis M. Fernández-Ramírez. Reinforcement learning based energy management system for grid-connected pv plants and energy-stored quasi-z-source cascaded h-bridge multilevel inverter. In *2023 IEEE International Conference on Environment and Electrical Engineering and 2023 IEEE Industrial and Commercial Power Systems Europe (EEEIC / ICPS Europe)*, pages 1–6, 2023.

[22] Asiri Iroshan. Deep reinforcement learning: A study of reinforcement learning with neural networks in industrial automation. *SSRN Electronic Journal*, 01 2023.

[23] Elmar Diederichs. Reinforcement learning - a technical introduction. *Journal of Autonomous Intelligence*, 2:25, 08 2019.

[24] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. 2019.

[25] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. 2018.

[26] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. 2017.

[27] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. 2024.

[28] 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*, PP:1–11, 11 2018.

[29] Tao Yang, Youhua Xiao, Zhen Zhang, Yiming Liang, Guorui Li, Mingqi Zhang, Shijian Li, Tuck-Whye Wong, Yong Wang, Tiefeng Li, and Zhilong Huang. A soft artificial muscle driven robot with reinforcement learning. *Scientific Reports*, 8, 09 2018.

[30] Lu Li, Junnan Li, Lei Qin, Jiawei Cao, Mohan Kankanhalli, and Jian Zhu. Deep reinforcement learning in soft viscoelastic actuator of dielectric elastomer. *IEEE Robotics and Automation*

Letters, PP:1–1, 02 2019.