applied sciences

MDPI

Article

Adjusted MTJ Control Strategy using Deep Reinforcement Learning for Tendon-Driven Continuum Manipulators

Nima Maghooli [1], Omid Mahdizadeh [1], Mohammad Bajelani [1] and S. Ali A. Moosavian [1,*]

[1] Center of Excellence in Robotics and Control, Advanced Robotics and Automated Systems (ARAS), Department of Mechanical Engineering, K. N. Toosi University of Technology, Tehran, Iran; Emails: nima.maghooli@ut.ac.ir , omid.mahdizadeh@email.kntu.ac.ir , bajelani@email.kntu.ac.ir , moosavian@kntu.ac.ir.

* Correspondence: moosavian@kntu.ac.ir.

Abstract: Tendon-Driven Continuum Robots (TDCRs) are a class of robotic arms capable of large-scale, soft, and compliant manipulation. However, due to their complex dynamics and highly non-linearities, control of such systems is challenging. In this paper, a Deep Reinforcement Learning (DRL) control strategy to update and improve control characteristics is developed for TDCRs. This method utilizes a unified framework of the Transpose Jacobian (TJ) and Modified Transpose Jacobian (MTJ) control laws for closed-loop dynamic control, with the control parameters learned via one of the state-of-the-art DRL algorithms, i.e., Deep Deterministic Policy Gradient (DDPG). The proposed strategy has been evaluated on a verified model to track a 3D path. Due to the ability of the proposed method to capture non-linear behaviors, the pre-defined desired trajectory is tracked successfully with an RMSE of 1 cm. In comparison, the desired trajectory spans from –20 to 20 cm (almost covering the whole robot's task space), which is quite promising. The robustness to noise and disturbances is also investigated, which performs better than the control strategies proposed in the literature. Experimental implementations on a real TDCR platform indicate the potential of the proposed algorithm in control of TDCRs for real-time applications. While there is no need for a lengthy identification process, quick learning, robustness, and satisfying performance, even in the presence of noise and disturbances, make the proposed method more appropriate from the available solutions.

Keywords: Tendon-Driven Continuum Robots, Transpose Jacobian (TJ), Modified Transpose Jacobian (MTJ), Deep Reinforcement Learning, DDPG Algorithm.

Citation: To be added by editorial staff during production.

Academic Editor: Firstname Lastname

Received: date
Revised: date
Accepted: date
Published: date

1. Introduction

Recent advancements in the field of robotics have led to increased interest in harnessing the potential of continuum manipulators for a wide range of applications. These manipulators, constructed from flexible and deformable materials, offer unique advantages due to their inherent adaptability and intricate degrees of freedom. Using these flexible materials brings nonlinearities, time dependencies, and sensitivity to external interactions, which makes the modeling and control problems hard to solve. In pursuit of efficient control strategies for continuum manipulators, researchers have explored a multitude of algorithms, ranging from adaptive control strategies to soft computing techniques. Recently, the application of Deep Reinforcement Learning (DRL) in the control of dynamic systems and systems with structured or unstructured uncertainties has been presented [1], [2].

Continuum robots are flexible robotic manipulators that can change their shape and size. They were first developed in the 1960s and have been used in a variety of applications, including medical, manufacturing, aerospace, search and rescue, and nuclear [3]. The backbone of a continuum robot can be divided into three main types: discrete, hard, and soft. Discrete continuum robots have a backbone that consists of universal joints and

is actuated by cables. They are relatively heavier and can produce more moment than other types of continuum robots [4].

Hard continuum robots have a backbone made of shape memory alloy or spring and connected by spacer discs. Soft continuum robots have a backbone that is made of rubber or silicone. Tendon-driven actuation systems are commonly used in continuum robots because they offer flexibility, dexterity, reachability, and safety. However, they can be challenging to control due to the tension that can build up in the tendons as the robot bends. Tension control for tendon mechanisms is still challenging, and it demands additional equipment to compensate for slack during work [5].

One specific DRL algorithm that has shown promise in this domain is the Deep Deterministic Policy Gradient (DDPG) algorithm. DDPG is an off-policy DRL algorithm that combines deep neural networks with deterministic policy gradients. It has been successfully applied to a wide range of robotic control tasks, including those involving soft and continuum robots. DDPG excels in continuous action spaces, making it particularly well-suited for the precise and continuous control required in such robots. By learning policies that directly map states to actions, DDPG enables soft and continuum robots to perform intricate and delicate tasks with improved accuracy and efficiency [6].

As a primary method in the context of manipulator robots, the Transpose Jacobian (TJ) control strategy has emerged as a straightforward approach for governing motion. Despite its simplicity and suitability for certain scenarios, the TJ algorithm exhibits limitations such as fast trajectory tracking and susceptibility to noise-induced performance degradation [7]. To address these shortcomings, the Modified Transpose Jacobian (MTJ) control strategy was introduced [8]. By incorporating stored data from previous control commands and estimation of feedback linearization, it achieved enhanced performance in high-speed tracking tasks and improved noise rejection. The asymptotic stability of such a method is provided by the direct Lyapunov's theorem, which is grounded in mathematical induction [9].

Considering the advantages of two approaches, DRL and MTJ, the present paper introduces a novel approach to enhancing the performance of continuum manipulators. The main contribution of this study is to leverage the power of reinforcement learning to refine the TJ and MTJ control strategies, thereby boosting their adaptability and robustness in real-world applications. Their inherent limitations are intended to be overcome by infusing learning capabilities into these well-established algorithms, and their performance across a diverse range of tasks is intended to be further improved.

The focus of this research lies in developing a comprehensive framework that seamlessly integrates Deep Reinforcement Learning techniques with the TJ and MTJ control algorithms applied to continuum manipulators. The continuum manipulator is enabled to adapt to dynamic and uncertain environments through a systematic exploration of this synergy, thereby opening new avenues for its utilization in real-world scenarios. The proposed approach not only seeks to improve the trajectory tracking accuracy but also aims to enhance noise resistance, system stability, and efficiency.

The paper is organized as follows. Section 2 comprehensively provides preliminary materials for the continuum manipulators' structural, kinematic, and dynamic aspects. Section 3 delves into the traditional TJ and MTJ control strategies, highlighting their strengths and limitations within the introduced continuum robot context. Gain adaptation based on DRL is presented in Section 4, and various perspectives on its utilization are discussed. Section 5 develops RL-TJ and RL-MTJ methods and elucidates how reinforcement learning enhances TJ and MTJ strategies. Section 6 presents simulation results, which include path planning, providing tangible evidence of the effectiveness of our approach. Section 7 discusses experimental results obtained from the experimental set-up of a continuum robot developed in the ARAS laboratory. Lastly, Section 8 concludes our findings, outlines our substantial contributions to the continuum robot control problem, and discusses the future direction of this research, particularly focusing on dynamic environments and continuum robots.

2. Analytical Model for Simulations and Control System Design

The adopted structural configuration comprises a flexible and slender backbone characterized by the uniform distribution of spacer disks spanning its entire length. These spacer disks incorporate apertures that serve as conduits for guiding the tendons along the length of the backbone. This particular design engenders a partially constrained trajectory for the tendons, resulting in the tendon's linear alignment between adjacent spacer disks, thus forming a sequence of line segments traversing the backbone.

Material selection for both the tendons and the guiding channels is important in mitigating potential friction and tendon elongation issues. It is common practice to neglect these factors in modeling, primarily due to their comparatively marginal influence in relation to the dominant actuation forces. Consequently, this study refrains from incorporating considerations of frictional interactions between the tendons and the robot's backbone or spacer disks and accounting for tendon elongation in the analytical framework.

The choice of employing straight tendon routing is motivated by its prevalent utilization in practical applications. Within each segment of the robot's structure, the flexibility exists to use any desired number of tendons routed along the length of the backbone. However, it is imperative to note that the presence of a minimum of three tendons is essential to facilitate two degrees of freedom bending, specifically achieving spatial bending.

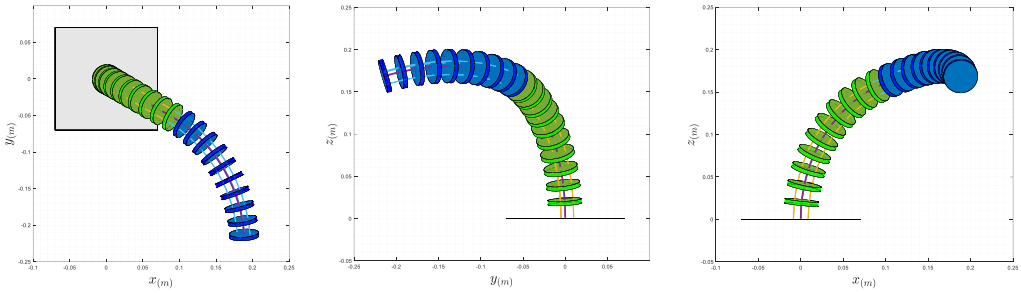


Figure 1. A general view of the TDCR model in three cartesian planes.

In this research, for simulating the behavior of a controlled system, we have employed a static model with the assumption of constant curvature for each sub-segment, commonly referred to as the "piecewise constant curvature model." This choice offers significant computational advantages compared to a variable curvature model while maintaining the necessary accuracy. The variable curvature model considers a time and shape dependent function $r = f(t, s)$ for the momentary position of each point in the central core, offering exceptionally high precision. However, this assumption ultimately leads to a set of nonlinear partial differential equations, resulting in a computationally intensive model that may not be practically feasible due to its computational complexity. On the other hand, the piecewise constant curvature model simplifies the problem by introducing two simplifying assumptions, effectively sidestepping the need for both time and shape dependency in the model. This simplification leads to a set of nonlinear algebraic equations, significantly reducing the computational load. Moreover, this model's accuracy remains satisfactory compared to the variable curvature model [10].

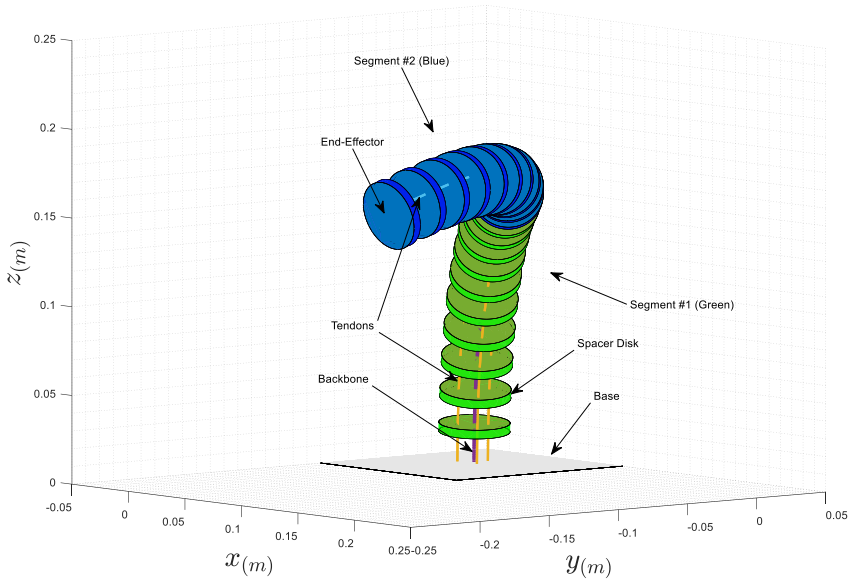


Figure 2. General view for TDCR model in 3D space.

In the following, we will elaborate on these two simplifying assumptions in further detail. Tendon-driven continuum robots typically possess very low mass, rendering their inertia negligible. Consequently, their dynamics can often be overlooked, allowing for their modeling as static (memoryless) systems. In this context, the time derivative (∂r/∂t) of the position vector is not considered in the model. The assumption of constant curvature for each segment is a fundamental premise in modeling piecewise constant curvature. As a consequence of this assumption, there is no need to consider the shape derivative (∂r/∂s) in the model, as the values of κ (curvature) and ϕ (orientation) for each subsegment fully determine the positions of all points along the robot's structure. These assumptions are crucial in simplifying the modeling and analysis of the motion of continuum robots driven by tendons, particularly when their dynamics can be considered quasi-static. The piecewise constant curvature assumption greatly simplifies the positional calculations within the system, making it an effective approach for practical applications and control problems.

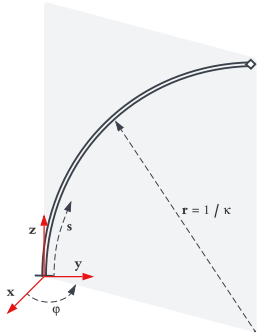


Figure 3. Parameters of configuration space of TDCR with constant curvature assumption.

2.1.Analysis of the Null Space of the Jacobian Matrix

The concept of the null space of a matrix A represents the set of all non-trivial solutions to the linear system of equations $AX = 0$. This concept finds practical applications in robotics, especially in tendon-driven and cable systems that often exhibit over-actuated

(excess actuators). In the context of controlling a tendon-driven continuum robot in a work space, where the number of system inputs (tendon tension forces) exceeds the number of outputs (end-effector coordinates), the system is considered over-actuated (six inputs and three outputs). For such systems, the Jacobian matrix (J_L) is typically a non-square matrix with dimensions of $m \times n$ $(n > m)$.

To find the pseudo-inverse of this non-square Jacobian matrix, the right pseudo-inverse formula is used:

$$J^{\dagger} = J^{T}(JJ^{T})^{-1} \tag{1}$$

This yields:

$$JJ^{\dagger} = JJ^{T}(JJ^{T})^{-1} = I \tag{2}$$

This pseudo-inverse resolves the over-actuation and ensures that the system remains controlled.

- **Null Space in Velocity Space:** The relationship between joint space velocities ($\dot{L}$) and end-effector velocities ($\dot{X}$) in a continuum robot is expressed as $\dot{X} = J\dot{L}$. By utilizing the null space projection operator $[I - J^{\dagger}J]$, the relationship between joint space velocities and end-effector velocities in the null space can be defined as:

$$\dot{X} = J(I - J^{\dagger}J)\,\dot{L} \tag{3}$$

In the equation above, $\dot{L}$ can take any value, and $\dot{X}$ will always be zero in the null space.

- **Null Space in Generalized Forces Space:** The mapping of generalized forces between joint space and task space in a continuum robot is described as $\mathcal{F} = J^{-T}\tau$. By employing the null space projection operator $[I - [(J^{T})^{\dagger}]^{\dagger}(J^{T})^{\dagger}]$, the relationship between joint space and work space forces in the null space can be formulated as:

$$\mathcal{F} = (J^{T})^{\dagger}[I - [(J^{T})^{\dagger}]^{\dagger}(J^{T})^{\dagger}]\,\tau \tag{4}$$

In this equation, τ can take any value, and $\mathcal{F}$ will always be zero in the null space. Understanding the null space of the Jacobian matrix is crucial for controlling over-actuated systems like tendon-driven continuum robots, as it allows for efficient management of additional degrees of freedom and ensures stable control in the work space [11]

Continuum manipulators in the general form have four separate spaces, and there are dependent and independent mapping between them [12]. As shown in Figure 4, forward and inverse kinematics relations can be used to establish these mappings.

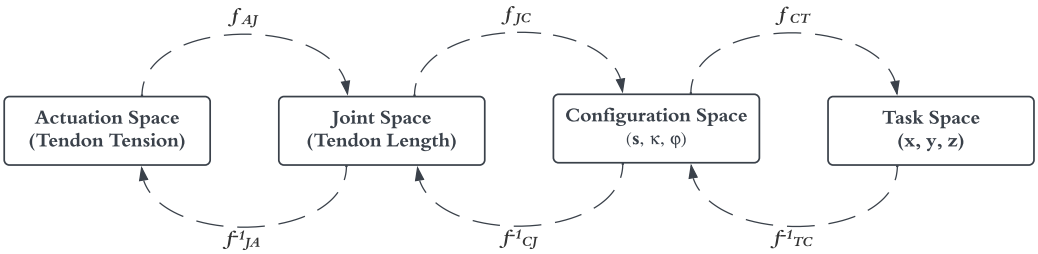


Figure 4. Different spaces of TDCRs in the general form.

Like other robotic systems, in the problem of position control, kinematics and dynamics equations (forward and inverse) are expressed, as shown in Figure 5.

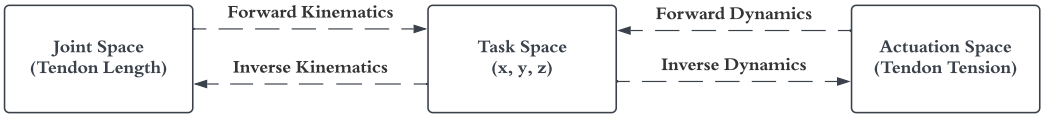


Figure 5. Kinematics and Dynamics (Kinetics) relations of TDCRs for position control problem.

One of the most critical challenges in tendon-driven systems and cable robots is the prevention of tendon slack or, in other words, the loss of tension in the tendons. Various methods have been proposed to address this issue in different systems, with some of the most common approaches including establishing appropriate pre-tension in the tendons and using the null space projection operator of the Jacobian matrix. Below, explanations for these methods are provided:

- **Establishing Appropriate Tendon Pre-Tension:** To prevent tendon slack during motion, one approach is first to simulate the desired path in software and then determine the maximum negative force required for the tendons. This maximum negative force is then considered as pre-tension for all tendons (utilizing the system's symmetry properties) to ensure that none of the tendons become slack throughout the robot's motion.

- **Adding Negative Tendon Force Magnitude to Corresponding Segment:** This method leverages the geometric symmetry property of the system. An equal moment is generated at the end-effector in the associated plane by applying an equal tension force to all tendons within a single segment. The symmetry ensures that no motion occurs in the work space.

- **Adding Negative Tendon Force Magnitude to All Segments:** Similarly, this approach relies on the geometric symmetry of the system. It is applied to all segment tendons across the entire system instead of adding the negative force magnitude to individual segment tendons. Again, no motion is observed in the work space due to the uniform moment generation in the end-effector's plane for all cables.

These methods effectively address the challenge of tendon slack by utilizing the geometric and symmetry properties of the system. They ensure that the tendons maintain adequate tension throughout the robot's motion, thereby preventing any undesirable slack or loss of control.

The following figures represent the results of simulating the circular path in the horizontal plane. Figure 6 illustrates the outcomes of various approaches that utilize the null space of the system to ensure tendon tension. Each method's effectiveness is measured by the maximum tension force achieved in the tendons during the robot's operation. The data displayed in Figure 6 provides valuable insights into the performance of these methods in preventing tendon slack and optimizing control in tendon-driven systems.

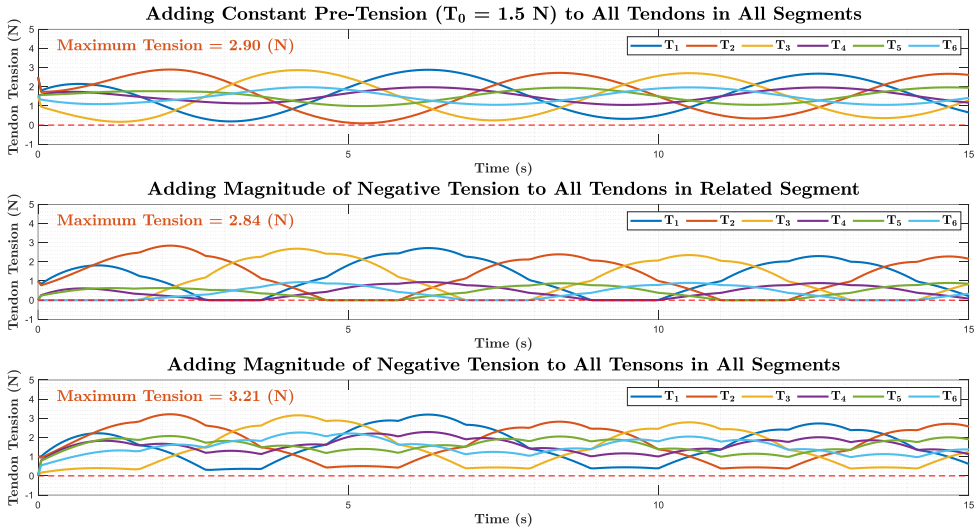


Figure 6. Comparison of three scenarios in the null space for surviving tendon tension.

Based on the presented results, it appears that the method of adding the negative tendon force magnitude to the corresponding segment tendons is the most suitable approach. This is because, compared to the two previous methods, it generates lower maximum tendon tension forces, which signifies better signal management. In Figure 7, the end-effector positions in the work space for the three methods described are displayed, and as expected, the results for all three strategies are identical.

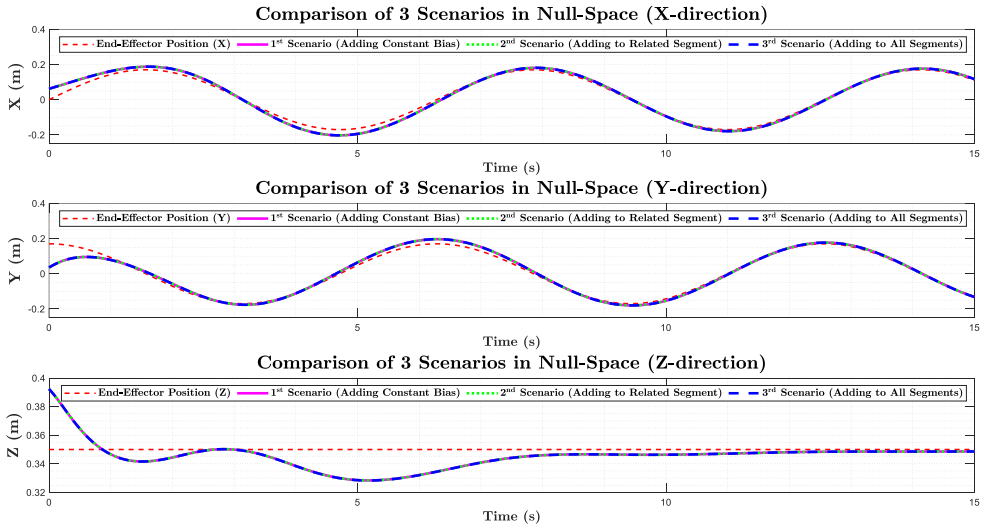


Figure 7. Obtained results of three scenarios in the null space.

As Figure 7 shows, the method of adding the negative tendon force magnitude to the corresponding segment tendons stands out as the most preferable option due to its lower maximum tendon tension forces, which can lead to more efficient and stable control of the system.

3. Transpose Jacobian (TJ) and Modified Transpose Jacobian (MTJ) control strategies for continuum manipulators

The Transpose Jacobian (TJ) algorithm is one of the simplest used in manipulator control. This algorithm is model-free, and besides its advantages, it will face weak

operations in tracking high-speed trajectories and has no systematic gain selection method. In the presence of noise, this high-gain controller also worsens performance. To solve these problems, a modification to this strategy was proposed and called Modified Transpose Jacobian (MTJ). In the following, both algorithms are explained.

3.1. Transpose Jacobian (TJ) control strategy:

One of the most common model-free control algorithms in the work space is the Transpose Jacobian, which performs well in controlling robotic systems' positions despite its very simple structure. If the position vector of the end-effector is defined as $\boldsymbol{X} = [x \quad y \quad z]^T$ and the effective length vector of the tendons is defined as $\boldsymbol{L} = [l_1 \quad l_2 \quad l_3 \quad l_4 \quad l_5 \quad l_6]^T$, then the linear Jacobian matrix that establishes the mapping between the rate of change of these two vectors as $\dot{\boldsymbol{X}} = \boldsymbol{J}_L \dot{\boldsymbol{L}}$ is calculated as follows:

$$\boldsymbol{J}_{L_{nm}} = \frac{\partial X_n}{\partial L_m} \quad , \quad n = 1, \dots, 3 \quad , \quad m = 1, \dots, 6 \tag{5}$$

Finally, the Jacobian matrix is obtained as follows:

$$\boldsymbol{J}_L = \begin{bmatrix} \frac{\partial x}{\partial l_1} & \frac{\partial x}{\partial l_2} & \frac{\partial x}{\partial l_3} & \frac{\partial x}{\partial l_4} & \frac{\partial x}{\partial l_5} & \frac{\partial x}{\partial l_6} \\ \frac{\partial y}{\partial l_1} & \frac{\partial y}{\partial l_2} & \frac{\partial y}{\partial l_3} & \frac{\partial y}{\partial l_4} & \frac{\partial y}{\partial l_5} & \frac{\partial y}{\partial l_6} \\ \frac{\partial z}{\partial l_1} & \frac{\partial z}{\partial l_2} & \frac{\partial z}{\partial l_3} & \frac{\partial z}{\partial l_4} & \frac{\partial z}{\partial l_5} & \frac{\partial z}{\partial l_6} \end{bmatrix} \tag{6}$$

If the vector of generalized forces in the work space is defined as $\boldsymbol{\mathcal{F}} = [F_x \quad F_y \quad F_z]^T$ and the vector of generalized forces in the joint space (tendon tension forces) is also defined as $\boldsymbol{\tau} = [T_1 \quad T_2 \quad T_3 \quad T_4 \quad T_5 \quad T_6]^T$, the mapping of these forces for control in the task space is expressed as follows:

$$\boldsymbol{\tau} = \boldsymbol{J}_L^T \boldsymbol{\mathcal{F}} \tag{7}$$

If the position error vector in the work space is defined as $\boldsymbol{e} = [e_x \quad e_y \quad e_z]^T$, the control input vector is expressed by the following relation:

$$\boldsymbol{\tau} = \boldsymbol{J}_L^T \left[\boldsymbol{K}_p \boldsymbol{e} + \boldsymbol{K}_i \int \boldsymbol{e}\, dt + \boldsymbol{K}_d \dot{\boldsymbol{e}} \right] \tag{8}$$

The gains considered in relation 8 are regarded as diagonal matrices in the following form:

$$\boldsymbol{K}_p = \begin{bmatrix} k_{p_x} & 0 & 0 \\ 0 & k_{p_y} & 0 \\ 0 & 0 & k_{p_z} \end{bmatrix} \quad , \quad \boldsymbol{K}_i = \begin{bmatrix} k_{i_x} & 0 & 0 \\ 0 & k_{i_y} & 0 \\ 0 & 0 & k_{i_z} \end{bmatrix} \quad , \quad \boldsymbol{K}_d = \begin{bmatrix} k_{d_x} & 0 & 0 \\ 0 & k_{d_y} & 0 \\ 0 & 0 & k_{d_z} \end{bmatrix} \tag{9}$$

The block diagram of the control system for the TJ controller designed to control the position of tendon-driven continuum robots is depicted in Figure 8.

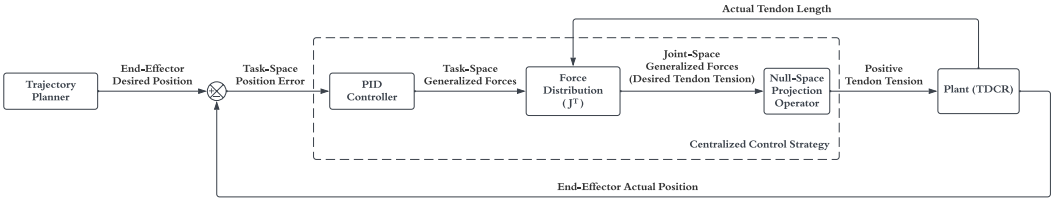


Figure 8. Customized block diagram of Transpose Jacobian (TJ) algorithm for TDCRs.

In addition to having advantages such as the simplicity of the structure (low computational cost) and being model-free, the TJ controller has disadvantages that motivate the

modification of this strategy. Due to having large control gains, this controller is sensitive to noise and strengthens its effect. In addition, when tracking fast trajectories, the quality of following its path decreases and there is no systematic method to determine its control gains.

3.2. Modified Transpose Jacobian (MTJ) control strategy:

According to the explanations presented in the previous section, the MTJ control algorithm is introduced to modify the TJ control strategy, maintaining its advantages and correcting the proposed disadvantages. This strategy is trying to implement a function similar to feedback linearization in model-based controllers (inverse dynamics) by estimating the system dynamics using the control input at the previous moment. Modification in the TJ structure is done by adding a modifier term in the form of vector $\boldsymbol{h} = [h_x \quad h_y \quad h_z]^T$ to equation 8 and reaching the following equation:

$$\boldsymbol{\tau} = \boldsymbol{J}_L^T \left[\boldsymbol{K}_p \boldsymbol{e} + \boldsymbol{K}_i \int \boldsymbol{e}\, dt + \boldsymbol{K}_d \dot{\boldsymbol{e}} + \boldsymbol{h} \right] \tag{10}$$

The modifier term ($\boldsymbol{h}$) is calculated as follows:

$$\boldsymbol{h}_{(t)} = \boldsymbol{K} \boldsymbol{\mathcal{F}}_{(t-\Delta t)} \tag{11}$$

where $\boldsymbol{\mathcal{F}}_{(t-\Delta t)}$ is the control input of the previous time step in the work space and the diagonal matrix $\boldsymbol{K}$ is considered as follows:

$$\boldsymbol{K} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix} \tag{12}$$

The principal diameter elements in the $\boldsymbol{K}$ matrix are calculated with the following relation:

$$k_i = exp\left[-\left(\frac{|e_i|}{e_{max_i}} + \frac{|\dot{e}_i|}{\dot{e}_{max_i}} \right) \right] \quad , \quad i = x\,, y\,, z \tag{13}$$

where e_{max_i} is the error sensitivity threshold and $\dot{e}_{max_i}$ is the error derivative sensitivity threshold for activating the modifier term. The block diagram of the control system for the MTJ controller designed to control the position of tendon-driven continuum robots is depicted in Figure 9.

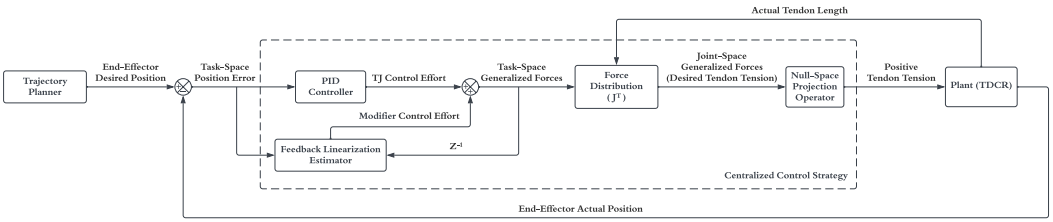


Figure 9. Customized block diagram of Modified Transpose Jacobian (MTJ) algorithm for TDCRs.

Finally, the MTJ algorithm is presented with the structure described above, keeping the advantages of TJ (structure simplicity and being model-free) and eliminating problems such as large control gains (strengthening the effect of noise) and weakness in tracking fast trajectories. For this control strategy, the proof of stability has been done based on Lyapunov's stability theorems, and the algorithm is asymptotically stable. The modified Transpose Jacobian algorithm has been used to control different systems, including under-actuated systems [13], humanoid (biped) robots [14], quadruped (four-legged) robots [15], wheeled mobile robots [16], 3-RRS parallel robots [17], and also in object manipulation by two humanoid robots [18]. In order to develop this strategy, many efforts have

been made so far under the title of Tubed Modified Transpose Jacobian [19], Adaptive Modified Transpose Jacobin [20], Learning-based Modified Transpose Jacobin [21], and Time-delay learning-based control [22].

4. Gain Adaptation based on Deep Reinforcement Learning

The Deep Deterministic Policy Gradient (DDPG) algorithm is a powerful reinforcement learning technique that has found significant utility in control and robotics applications. DDPG excels in continuous action spaces, making it particularly well-suited for controlling robotic systems with a wide range of motion. One of its primary benefits is its ability to learn complex, high-dimensional control policies directly from raw sensory data, allowing robots to adapt and optimize their actions in real-time. This adaptability is especially valuable in dynamic and uncertain environments where traditional control methods may struggle. DDPG has been effectively employed in tasks offering a promising avenue for advancing the capabilities of robots and enabling them to operate more effectively and autonomously in diverse real-world scenarios [23]. This research primarily focuses on gain adaptation based on Deep Reinforcement Learning, introducing key elements such as Figure 10, illustrating the Performance Analogy between Reinforcement Learning and Control Systems. To clarify the concept, similar components are highlighted with the same color.

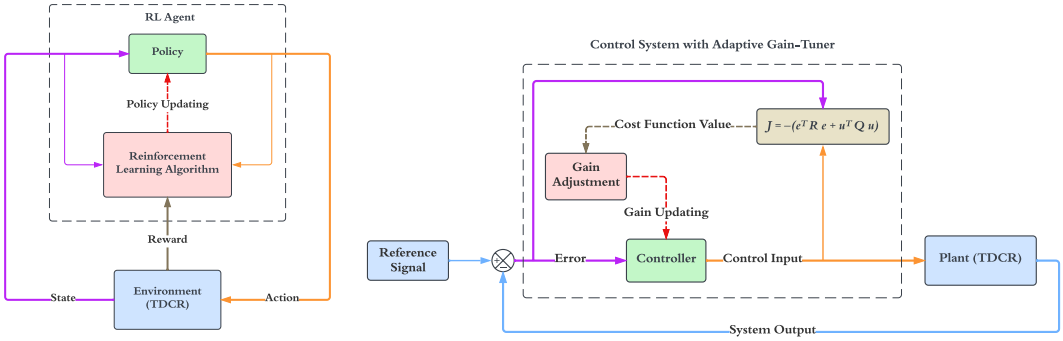


Figure 10. Performance analogy of Reinforcement Learning flowchart and Control System block diagram.

The high-level structure of DDPG, incorporating an actor-critic framework, is depicted in the Figure 11 which plays a central role in elucidating the DDPG algorithm, serving as a pivotal visual aid in comprehending its fundamental workings. This schematic diagram provides a concise yet comprehensive representation of the algorithm's components and their interactions, facilitating a deeper understanding of its implementation within the research context.

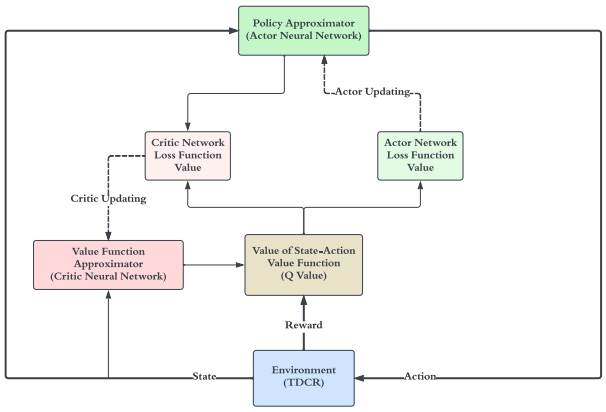


Figure 11. The schematic diagram of the Deep Deterministic Policy Gradient (DDPG) algorithm.

Furthermore, Figure 12 goes beyond a mere portrayal, offering an intricate and detailed description of the DDPG algorithm's inner workings. It serves as a guiding reference, providing researchers and practitioners with invaluable insights into the step-by-step processes and computations involved in DDPG, enabling them to effectively implement and adapt this algorithm to specific control and robotics applications.

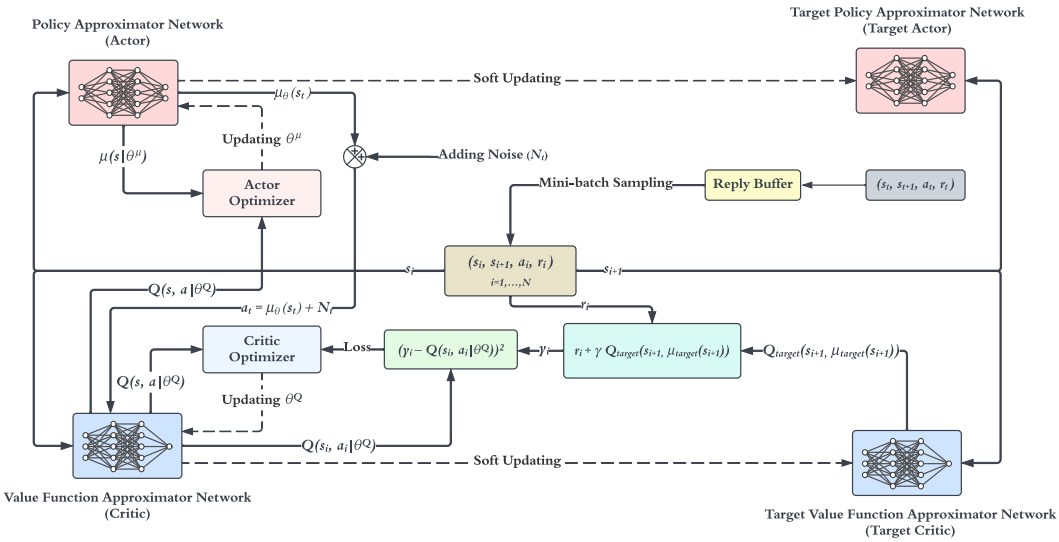


Figure 12. The structure and learning process of Deep Deterministic Policy Gradient algorithm.

Figure 12, on the other hand, takes a holistic approach by expanding upon the structure and learning process of the DDPG algorithm. It provides a visual narrative, guiding readers through the intricate flow of information and control within the algorithm, making it an essential resource for grasping its core principles. Additionally, Table 1 serves as a valuable reference point, meticulously documenting the critical hyperparameters that play a pivotal role in configuring and fine-tuning the DDPG algorithm for various robotic control tasks. Together, these visual and textual components not only enhance the reader's comprehension of DDPG but also offer a practical foundation for its application in the field of control and robotics, where adaptability and precision are paramount.

Algorithm 1: Deep Deterministic Policy Gradient (DDPG) algorithm [6]

Randomly initialize critic network $Q(s,a \mid \theta^Q)$ and actor $\mu(s \mid \theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
Initialize a random process $\mathcal{N}$ for action exploration
Receive initial observation state s_1
for t = 1, T **do**
Select action $a_t = \mu(s_t \mid \theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
Execute action a_t and observe reward r_t and observe new state s_{t+1}
Store transition (s_t, a_t, r_t, s_{t+1}) in R
Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
Set $y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1} \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right)$
Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i \left(y_i - Q(s_i, a_i \mid \theta^Q)\right)^2$
Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a \mid \theta^Q)\Big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s \mid \theta^\mu)\Big|_{s_i}$$

Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

end for
end for

Table 1. Hyperparameters of DDPG algorithm.

Hyperparameter	Value
Experience Buffer Length	10^6
Target Smooth Factor	10^{-3}
Mini Batch Size	1024
Standard Deviation of Noise	0.5
Standard Deviation Decay Rate	10^{-5}
Sample Time	0.03 s
Episode Time	3 s
Discount Factor	0.99
Maximum Number of Episodes	1000

Reinforcement Learning (RL) has significant potential for solving engineering problems, particularly control engineering [24]. Figures 13-15 illustrate the relationship between an RL problem and a control problem. In general, three strategies can be considered for applying Reinforcement Learning in the control loop [25], [26]:

- **1st Strategy:** Intelligent Control and Force Distribution

In the first strategy, the learning agent acts as an intelligent controller and force distributor. The primary goal is to discover suitable learning structures and parameters as a model-free controller and correctly distribute forces to system actuators under various operating conditions. This strategy has no explicit model of the system's dynamics and kinematics. Figure 13 illustrates the block diagram related to this strategy.

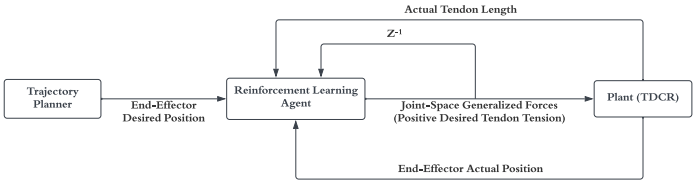


Figure 13. RL Agent in the role of Intelligent Control and Force Distribution for TDCRs.

- **2nd Strategy:** Intelligent Control

In the second strategy, the learning agent takes on the role of an intelligent controller. The primary objective is to learn appropriate control structures and parameters as a model-free controller under various operational conditions. This strategy has no explicit model of the system's dynamics, but the system's kinematics (Jacobian matrix) is available and utilized to establish force distribution. Figure 14 illustrates the block diagram associated with this strategy.

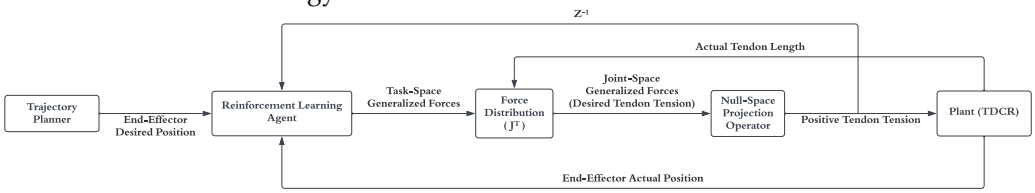


Figure 14. RL Agent in the role of Intelligent Control for TDCRs.

- **3rd Strategy:** Adaptive Gain-Tuner

In the third strategy, the learning agent assumes the role of an adaptive gain-tuner. The primary objective is to find the control gain gains as a model-free controller under various operational conditions. This strategy also operates without an explicit model of the system's dynamics, relying solely on the system's kinematics (Jacobian matrix) for force distribution. Figure 15 illustrates the block diagram associated with this strategy.

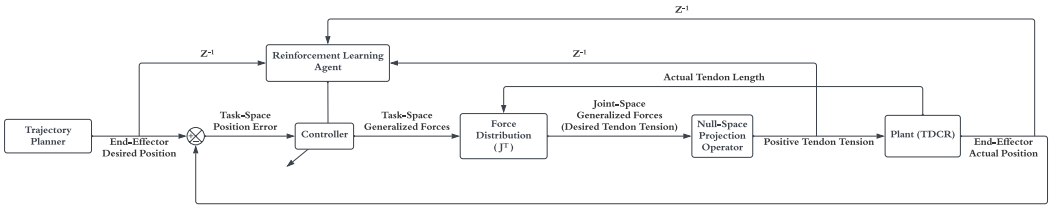


Figure 15. RL Agent in the role of Adaptive Gain-Tuner for TDCRs.

The selection of the third strategy is based on two critical considerations. First, deriving the Forward Kinematics equations for obtaining the Jacobian matrix is not particularly challenging, rendering the first strategy less appealing. Second, using a reinforcement learning (RL) agent as an intelligent controller necessitates a longer learning period and greater computational resources, rendering the second strategy less practically viable. The third strategy offers several notable advantages, including a reduction in learning time since the learning agent need not acquire the controller's structure, ensuring safety in the learning process, and the ability to develop robust controllers with adaptive gains. Moreover, the resulting policy is lightweight and suitable for real-time implementation on hardware. At the same time, the strategy's information range for controller gains minimizes issues related to instability and other challenges during the learning process. Ultimately, the third strategy aligns with the primary objective of designing a controller with variable and suitable gains for tracking any desired path within the system's work space.

5. RL-TJ and RL-MTJ:

In model-free control strategies, hyperparameters play an important role in determining the performance of the system. Many efforts have been made in developing the TJ and MTJ algorithms, and here, these parameters will be adjusted by Deep Reinforcement Learning. RL-TJ and RL-MTJ refer to control strategies that integrate Reinforcement Learning (RL) techniques with the TJ and MTJ methods, respectively shown in Figures 16 and 17. These approaches leverage RL algorithms to dynamically adjust control gains, allowing adaptability in response to changing environments or task requirements. Based on RL, the Adaptive Gain-Tuner is a mechanism designed to autonomously optimize control gains during operation autonomously, enhancing the system's ability to respond effectively to varying conditions. This adaptive approach contributes to improved performance and robustness in controlling robotic manipulators, particularly in complex and dynamic environments.

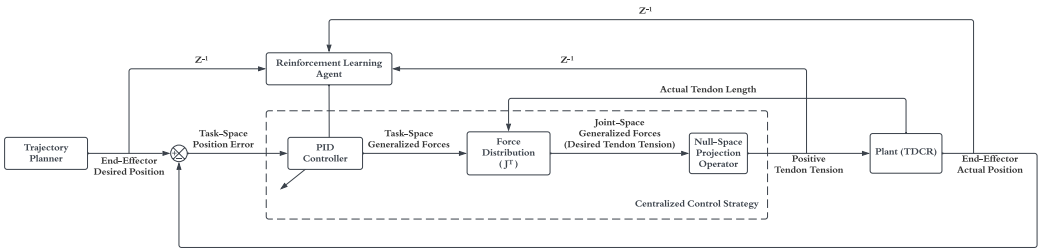


Figure 16. Proposed block diagram for RL-TJ Control Strategy for TDCRs.

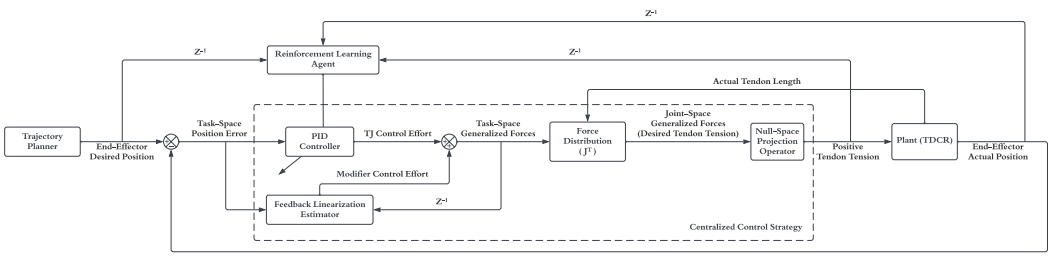


Figure 17. Proposed block diagram for RL-MTJ Control Strategy for TDCRs.

The learning agent's task is to determine suitable values for the control gains within the matrices $\boldsymbol{K_P}$, $\boldsymbol{K_I}$ and $\boldsymbol{K_D}$. In the application of reinforcement learning to engineering problems, it is essential to define the reward function, states (observations), and actions:

- **Reward Function:** Given the information from the controlled system (environment), the reward function is defined as a combination of the sum of squared errors (SSE) and obtained gains for the controller.

$$Reward\ Function = -[SSE + 10\ f_G]$$
$$f_G = \sum_{i=x,y,z} \left[\left(K_{P_i} \le K_{I_i}\right) + \left(K_{P_i} \le K_{D_i}\right) + \left(K_{I_i} \le K_{D_i}\right)\right] \quad (14)$$

- **States (Observations):** The system states are determined as set of the end-effector position, position error, and joint space forces.

$$States = \{x, y, z, e_x, e_y, e_z, T_1, T_2, T_3, T_4, T_5, T_6\} \quad (15)$$

- **Actions:** Actions, which serve as the agent's outputs, are chosen based on the intended objective for the learning agent (adjusting controller gains) and include the proportional, integral, and derivative gains for each position coordinate.

$$Actions = \{K_{p_x}, K_{i_x}, K_{d_x}, K_{p_y}, K_{i_y}, K_{d_y}, K_{p_z}, K_{i_z}, K_{d_z}\} \quad (16)$$

In this context, the reinforcement learning strategy aligns to fine-tune controller gains for optimal system performance. In Figure 18, the structures of multi-layer perceptron (MLP) neural networks, along with their details (number of layers, number of neurons in each layer, activation functions, etc.), which have been used for policy estimation (actor) and value function estimation (critic), are visible. These neural networks are an integral part of the reinforcement learning framework, where the actor network is responsible for approximating the policy that determines the agent's actions, and the critic network estimates the value function, assessing the desirability of states or state-action pairs. These networks play a crucial role in the decision-making process of the learning agent and are instrumental in optimizing the control gains for the given control problem.

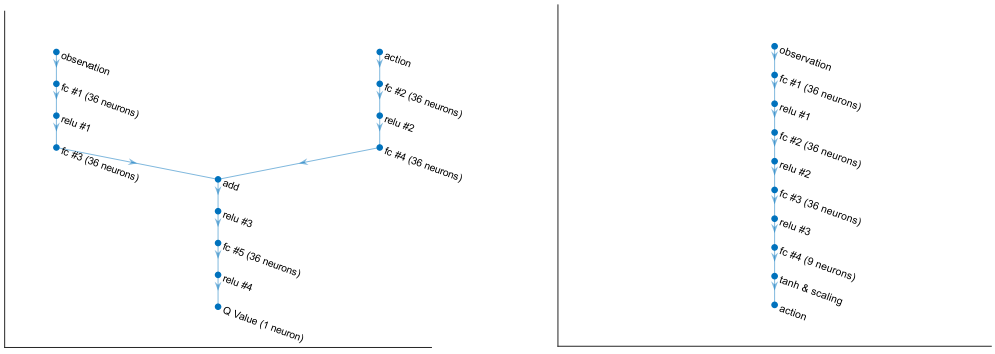


Figure 18. Proposed structures for Actor (left) and Critic (right) MLP Neural Networks.

In Figure 19, the changes in discounted rewards obtained by the learning agent in each episode, along with their average, are depicted. Given that all rewards are defined as negative values in the reward function, the ideal state would be to find a policy that receives a zero reward throughout the episodes. As shown in the image, after approximately 700 episodes, the learning agent has successfully approximated the desired policy as the rewards approach zero. This indicates that the agent has learned to follow a policy that leads to favorable outcomes according to the defined reward function.

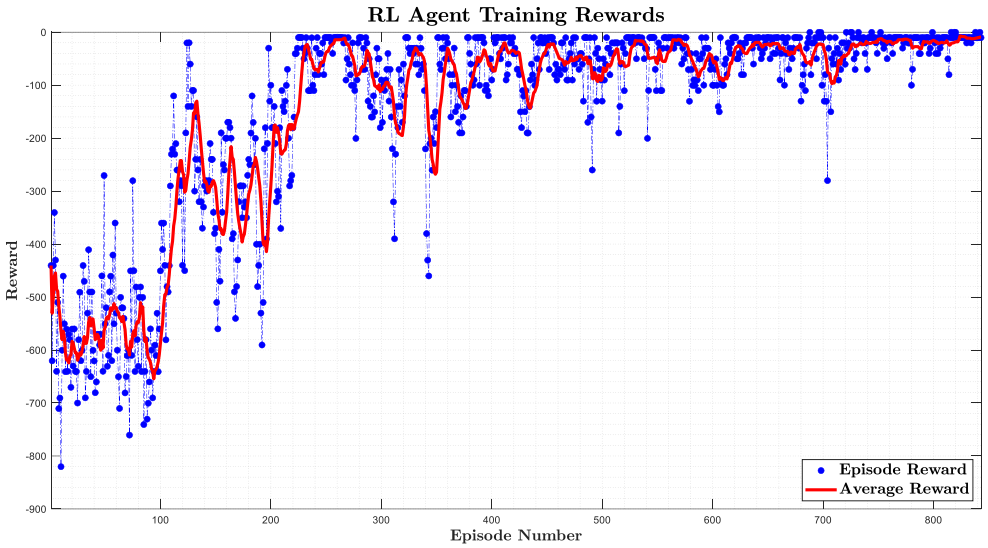


Figure 19. Variation of episode reward and average reward with respect to episode number.

Figure 20 presents the changes in the number of steps taken in each episode by the learning agent, along with their average. This graph serves as another evidence of the success of the learning agent in approximating the desired policy. After approximately 700 episodes, the number of steps taken during each episode has increased. This implies that the episode termination condition (defined as unfavorable conditions) has not been triggered, indicating that the agent is effectively following a policy that avoids unfavorable outcomes, leading to more steps being taken in each episode.

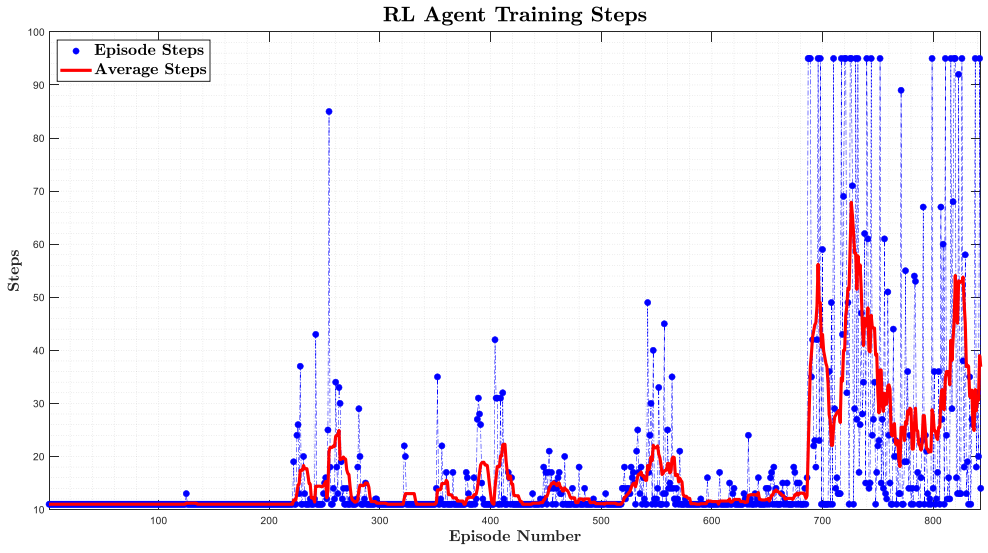


Figure 20. Variation of training steps and average steps with respect to episode number.

Figure 21 shows a comparison between the average cumulative discounted rewards and the estimated value function generated by the critic network. This graph is the primary evidence supporting the claim of the learning agent's success in approximating the desired policy. According to the graph, the estimated value function produced by the critic network converges to a stable state after approximately 400 episodes. This can be interpreted as the expected value of the received rewards during each episode reaching a specific value. Now, considering Figure 21 of the average rewards received, after about 700 episodes, the actor-network has successfully found the desired policy, and the average

cumulative discounted rewards are nearly equal to the output of the critic network, which represents the estimated value function or, in other words, the expected value of the rewards received during each episode.

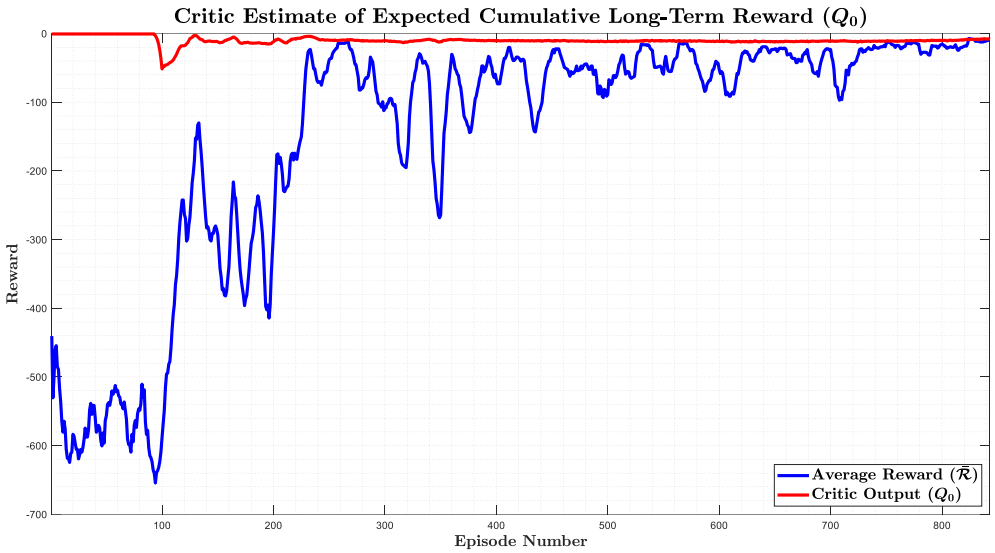


Figure 21. Variation of critic output and average reward with respect to episode number.

6. Simulation Results

To evaluate the effectiveness of our proposed method, extensive simulations in software (analytical model in MATLAB-Simulink) and experimental tests on hardware (physical platform in the ARAS laboratory) will be conducted using a representative continuum manipulator system. The study will compare the performance of the reinforced TJ and MTJ algorithms, shedding light on the advantages and potential challenges of our approach.

To evaluate the performance quality of the learning agent in adjusting the controller gains, a specific path (the result of designed trajectories) is depicted in Figure 22. This path is considered in a simulation for assessment purposes.

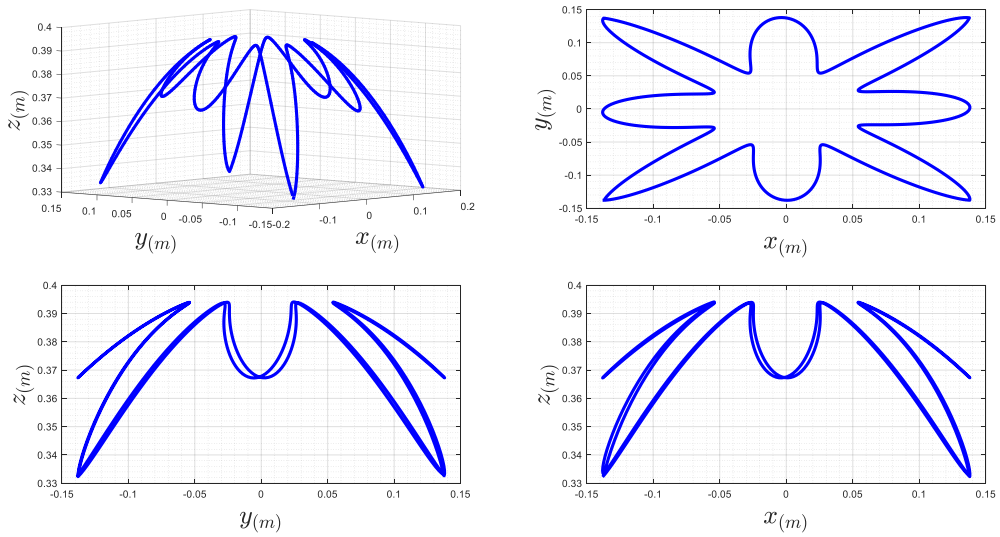


Figure 22. Desired path for TDCR in 3D space.

For visualizing the trajectory tracking quality, an animation was created by MATLAB-Simulink using source codes shared in [27], shown in Figure 23.

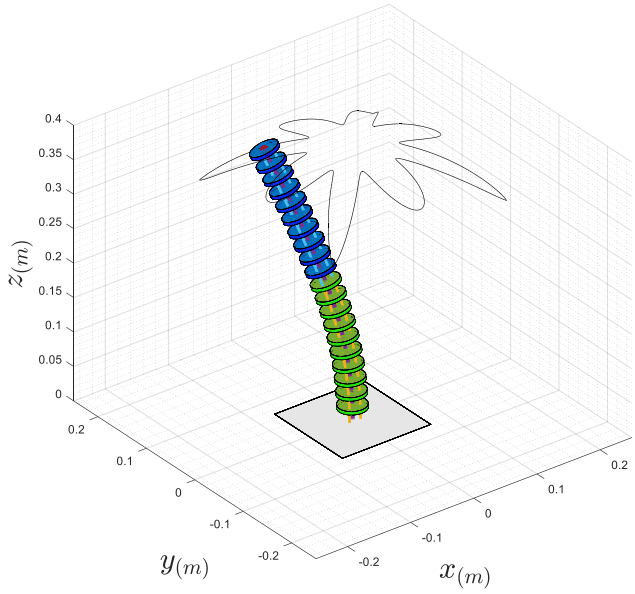


Figure 23. TDCR animation for trajectory tracking and related path in the 3D space.

6.1. First Scenario: Ideal Conditions (without noise and disturbance)

The obtained results from simulating the control systems described for traversing this path using a model with constant piecewise curvature are presented in the following figures. Figure 24 shows control gains' plots adjusted online by the learning agent as the path is traversed. An interesting observation in the obtained plots is the difference in the scale of the control gains between MTJ and TJ. This discrepancy arises because a significant portion of control is performed through the correction term with linear feedback approximation. As a result, the MTJ algorithm yields significantly smaller control gains compared to TJ. This phenomenon leads to the reduced sensitivity of this algorithm to noise, making it more robust in handling disturbances.

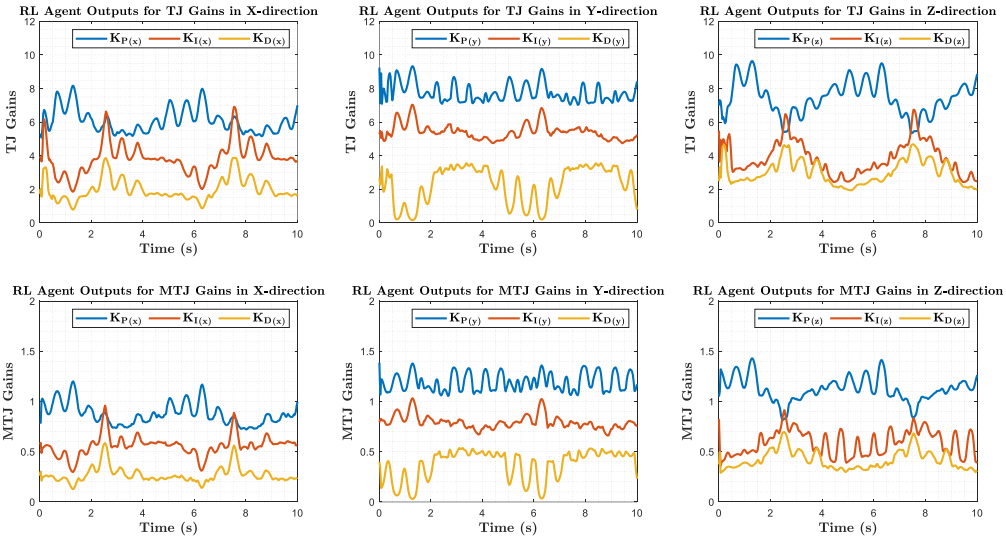


Figure 24. Comparison of controller gains obtained from the DDPG algorithm for RL-TJ and RL-MTJ strategies in the ideal conditions.

In Figure 25, the path-tracking performance for TJ and MTJ controllers with gain adjustments by the learning agent is depicted. The root mean square error (RMSE)

throughout the simulation for each designated end-effector position has been calculated and is visible in the images to better understand their performance. This metric provides insight into the quality of path tracking achieved by both controllers.

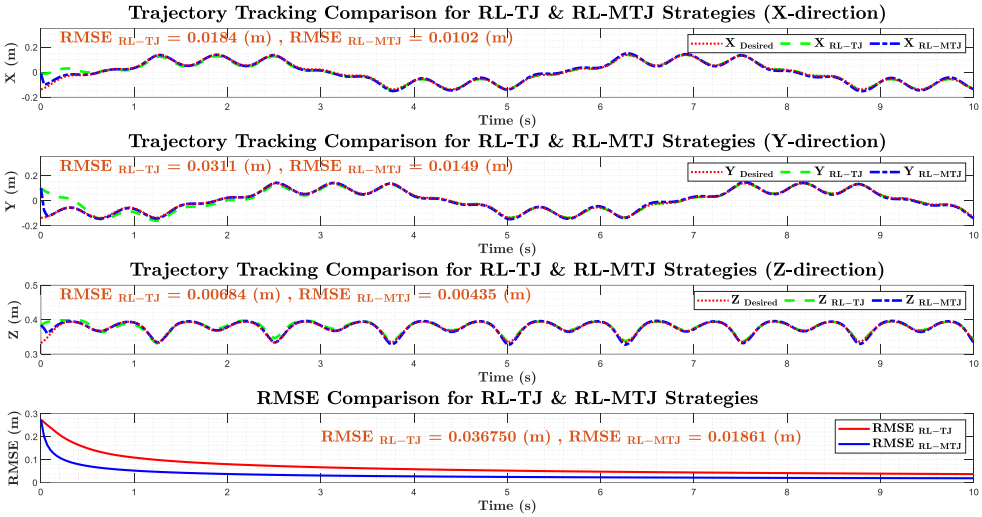


Figure 25. Comparison of trajectory tracking quality of RL-TJ and RL-MTJ strategies in the ideal conditions.

For a better understanding of the changes in the RMSE over time, the last plot of Figure 25 illustrates these variations for both strategies. The values of RMSE for the error vector over time are visible.

Finally, Figure 26 displays the tendon tension forces for both strategies. As can be observed, the tension forces are roughly within a similar range (in magnitude), indicating the better control and error minimization achieved by the MTJ algorithm in managing control inputs.

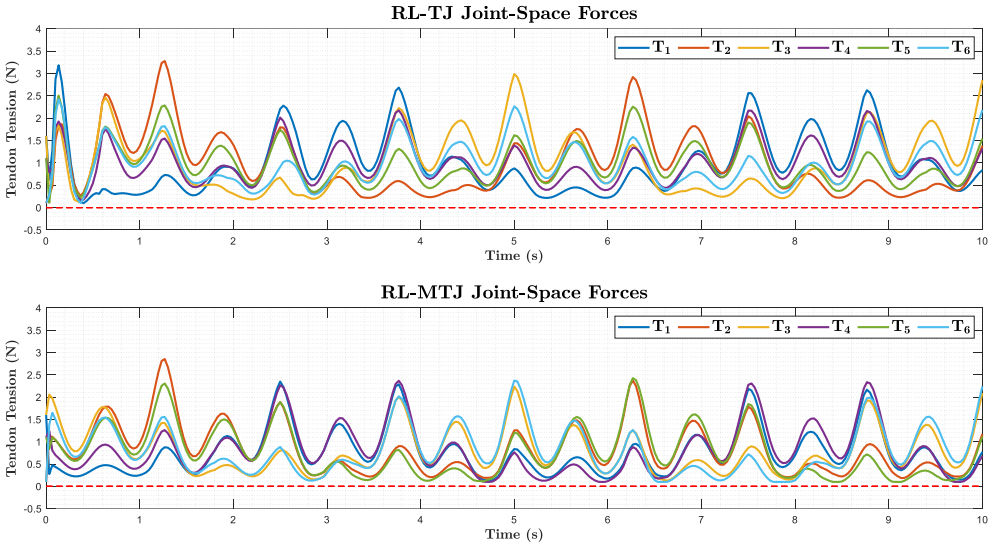


Figure 26. Comparison of tension force of tendons for RL-TJ and RL-MTJ strategies in the ideal conditions.

6.2. Second Scenario: Realistic Conditions (white noise and 15% disturbance)

To evaluate the path-tracking performance of controllers designed with gain tuning by the reinforcement learning agent, it is essential to approximate real-world conditions as

closely as possible. Achieving this state requires introducing disturbances (unwanted signals affecting the system's input) and noise (unwanted signals affecting the system's output) during the simulation to assess the controller's performance in their presence. For simulating disturbances, the input signal to the controlled system (i.e., the control input in joint space) has been altered by 15% in a completely random manner. White noise (with an average of zero and a standard deviation of approximately 5 millimeters) has been considered for simulating noise, as shown in Figure 27.

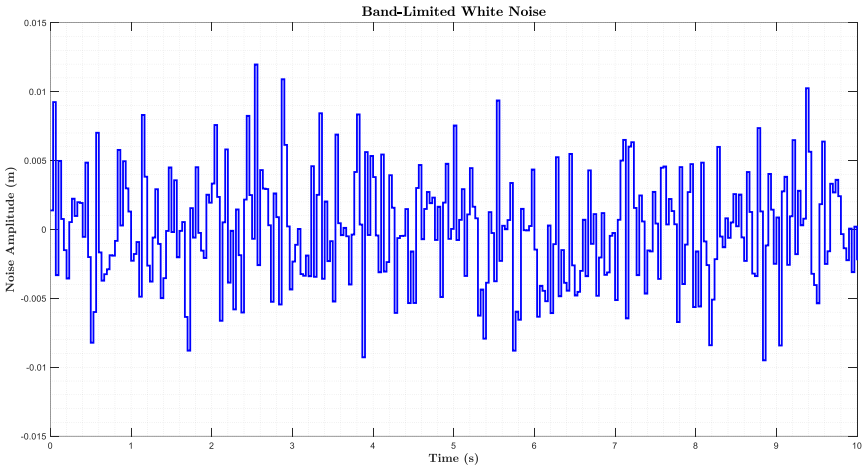


Figure 27. White noise with zero mean a standard deviation of 0.005.

The following output represents the controller gains tuned by the reinforcement learning agent during the path-tracking process by simulating the control systems under these conditions. A notable observation in the obtained output is the gains difference in scale. In this case, the gains for MTJ are also significantly smaller than those for TJ. The effect of noise on the gain charts for TJ, obtained on a larger scale, is clearly visible.

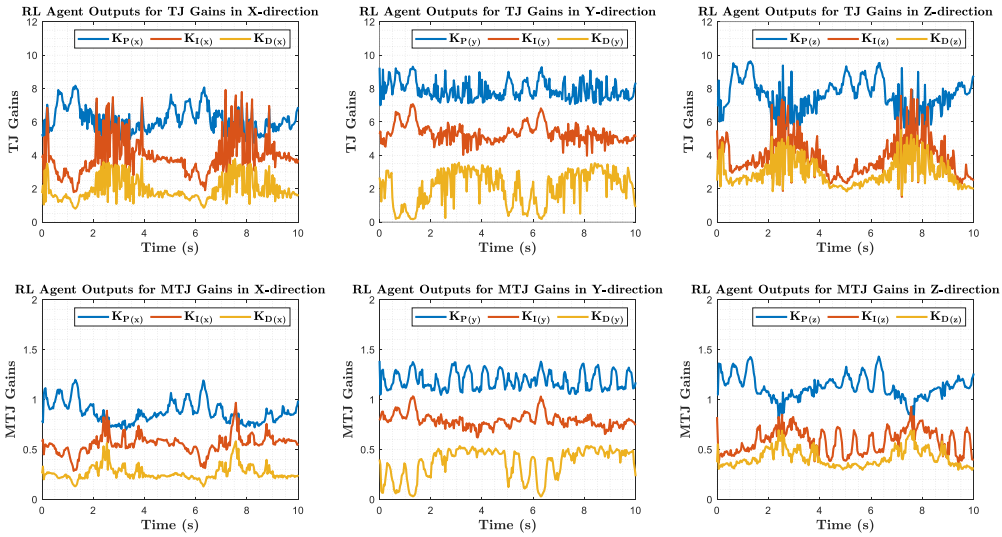


Figure 28. Comparison of controller gains obtained from the DDPG algorithm for RL-TJ and RL-MTJ strategies in the presence of noise and disturbance.

In Figure 30, the path-tracking performance of TJ and MTJ controllers in the presence of noise and disturbance is illustrated. The root mean square error (RMSE) throughout the simulation for each joint position is calculated and presented in the charts to better understand their performance.

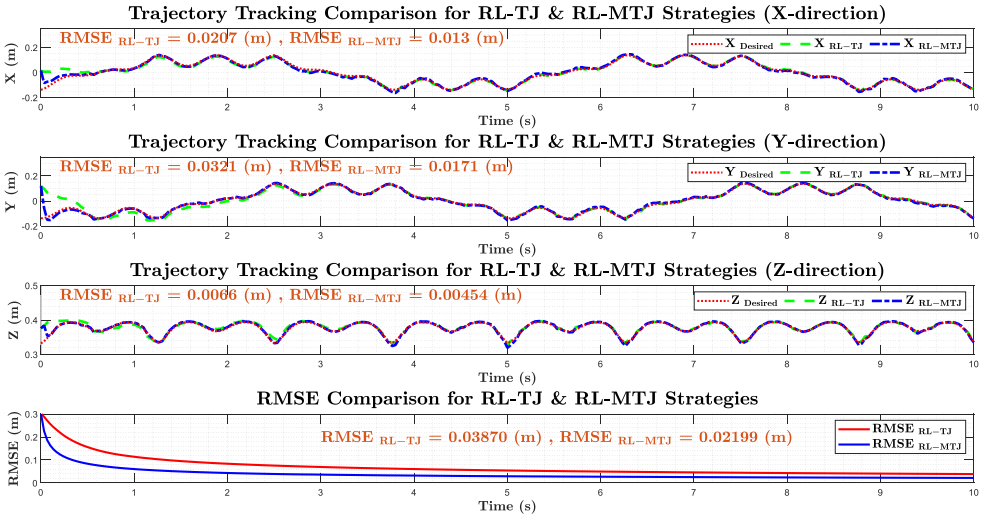


Figure 29. Comparison of trajectory tracking quality of RL-TJ and RL-MTJ strategies in the presence of noise and disturbance.

The last plot of Figure 29 provides a comprehensive view of the changes in the root mean square error (RMSE) over time for both strategies in the presence of noise and disturbance.

In Figure 30, the graphs depict the tendon tension forces for both strategies in the presence of noise and disturbance. The tendon tension forces are roughly within a specific range in both cases. This signifies the better management of the MTJ algorithm in adjusting control inputs and minimizing errors. Another significant observation is the considerable impact of noise and disturbance on the input signals to the system (tendon tension forces) in the TJ algorithm. It has sometimes even decreased tension forces (slackening of tendons). Conversely, for the MTJ algorithm, this effect is much less pronounced, and the signals are relatively smoother. This is due to the fact that the majority of control is performed by the corrective term with linear feedback estimation, and MTJ, with smaller control gains, exhibits lower sensitivity to noise and disturbance.

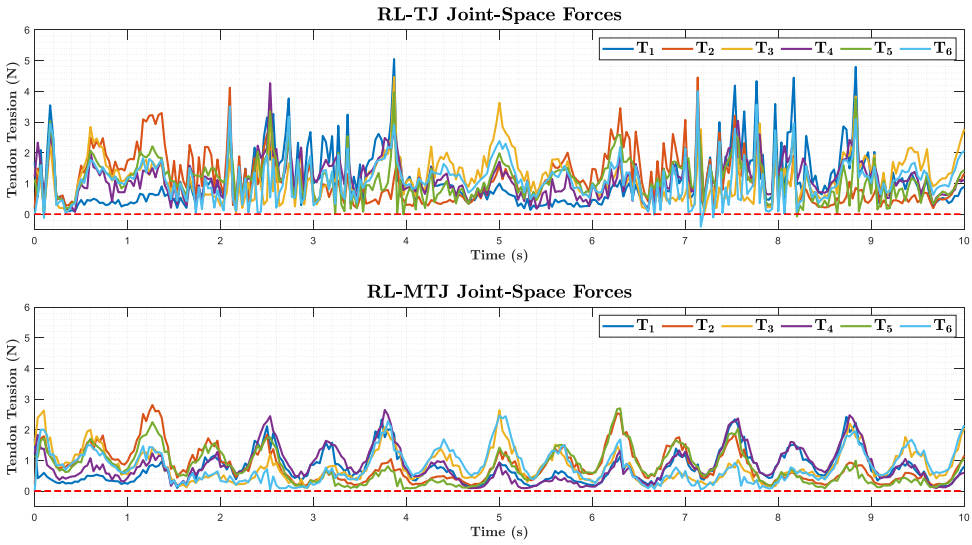


Figure 30. Comparison of tension force of tendons for RL-TJ and RL-MTJ strategies in the presence of noise and disturbance.

7. Experimental Results and Discussion

A very important point that reveals the difference between the simulation in the software environment and the implementation on the robot hardware is the type of input signals to the system under control (forward dynamics point of view). To clarify this issue, as previously explained about the model used for simulation, the input to the system is the tension force of the tendons, and by determining these inputs for the system, dynamic (kinetic) control is performed. On the other hand, due to the use of kinematic actuators (Dynamixel servo-motors used to control the position and speed) in the structure of the continuum robotic arm, if the input to the system is directly the angle of the motors, the kinematic control strategy is implemented. It will not be a good option for the continuum robot.

This claim is explained by three reasons. Firstly, the kinematic control has no sense of the tension force of the tendons, and in case the robot's body or the final executive collides with the work environment or the system is caught in the void, the controller will not have the ability to create correct commands to solve these problems. Secondly, in kinematic control, due to the lack of information about the tension force of the tendons, if this force increases to the threshold of their tolerance, the controller will not have any information about this issue, and this case can lead to tearing the tendons or damaging the spacers. Thirdly, the use of the transpose of the Jacobian matrix as a force-distributing element is possible only in the kinetic control mode, according to the relation $\tau = J^T \mathcal{F}$, which is the mapping between the forces in the joint space and the task space, τ, which is the input to the system under control, is tendon tension. As a result, if the kinematic control strategy is used, it is possible to use the transpose of the Jacobian matrix as a force distributing element will not exist. According to the above reasons, it is clear that the appropriate strategy for controlling the position of the continuum robotic arm is dynamic (kinetic) control.

The method of implementing kinetic control, despite having kinematic operators, is to use a cascade structure and create an internal loop to adjust the tension force of the tendons. In this structure, using the feedback of loadcells, the tension force of the tendons is calculated at every time steps, and after comparing it with the desired tension force, it is entered into the inner-loop controller (here PI). The image processing system calculates the instantaneous position coordinates of the end effector at every moment, and the outer-loop works with camera output data. In Figure 31, an illustration of the cascaded control structure used to control a continuum robotic arm is presented.

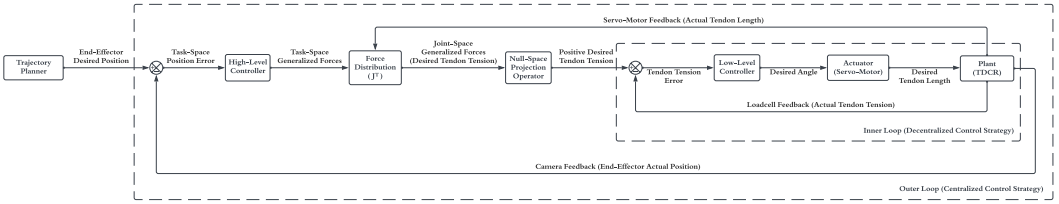


Figure 31. Proposed block diagram for closed-loop dynamic control in the task space for TDCRs to implement RL-MTJ strategy as high-level controller and simple PID for low-level controller.

In Figure 32, images of the tendon-driven continuum robotic arm developed in the ARAS laboratory are shown. The image on the right shows the equilibrium state and the image on the left shows a specific motion of the system.

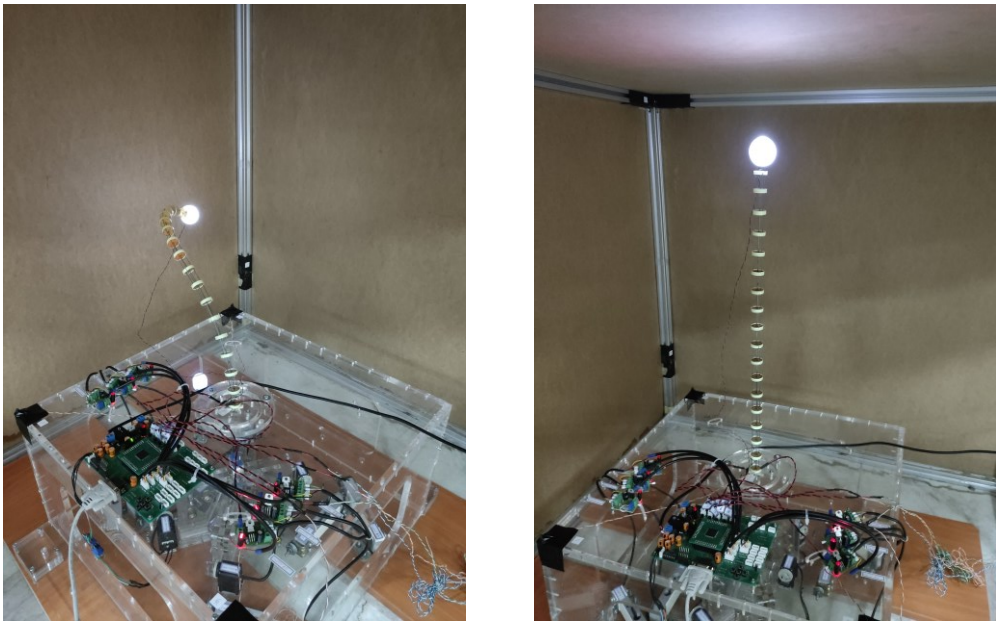


Figure 32. The TDCR was developed in the ARAS laboratory.

In order to evaluate the quality of controllers designed by adjusting gains by DRL, the results of implementing the RL-MTJ strategy on the real platform are presented here. In Figure 33, the graphs of the controller gains obtained online by the RL agent while tracking the trajectory, and the diagram of the joint space forces are presented.

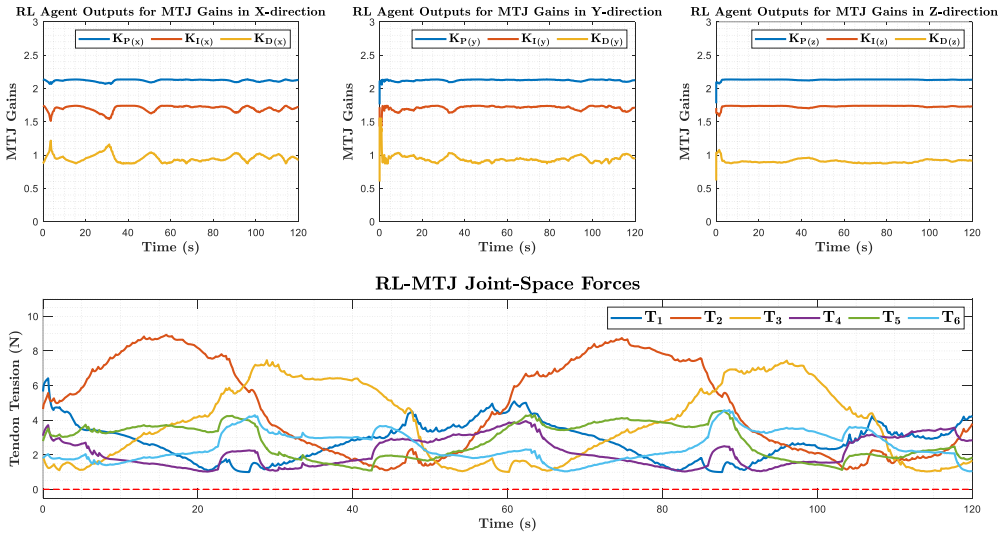


Figure 33. Comparison of controller gains obtained from DDPG algorithm for RL-MTJ strategy and implemented on hardware and tension force of tendons.

In Figure 34, the path tracking quality for the RL-MTJ controller is shown. For a better understanding of its performance, the root mean square error during the simulation and for each position coordinate is calculated, as seen in the figure. To show better the root mean square error changes over time, the last plot shows these changes, and its value for the error vector over time can be seen.

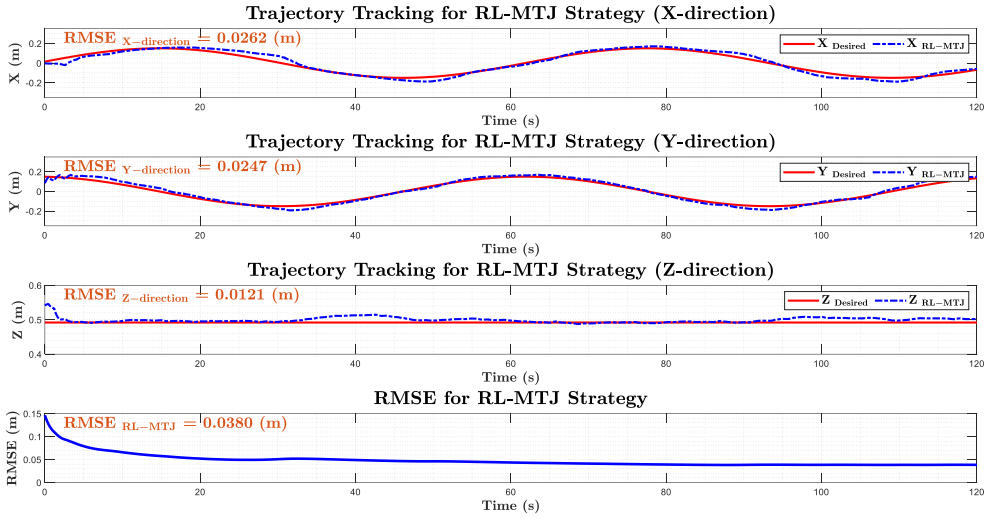


Figure 34. Trajectory tracking quality for RL-MTJ strategy and RMSE variations.

8. Conclusions

This paper proposes a novel DRL-based control strategy for Tendon-Driven Continuum Robots (TDCRs). The proposed approach was experimentally validated on a physical robot and was shown to be more robust to noise and disturbances than traditional control strategies. The proposed method uses the Modified Transpose Jacobian (MTJ) control law, while the control gains are learned using the DRL algorithm. The adaptive gain adjustment system was implemented using the Deep Deterministic Policy Gradient (DDPG) algorithm. The results showed that the adjusted MTJ strategy was able to track the desired path successfully with an RMSE of 1 cm in simulation and 3 cm in experimental tests. In brief, the main contributions of this paper are the development of a novel DRL-based control strategy for TDCRs and its experimental validation on a physical robot. The proposed strategy is a significant step forward in developing control strategies for TDCRs. It is a promising approach for controlling these robots in real-world environments. The proposed method is robust to the designated reference trajectory and initial condition of the robot. However, the experiments were conducted in a controlled environment, and the proposed strategy was only evaluated on a single TDCR. Future works can be focused on addressing these limitations and evaluating the proposed approach on a wider range of continuum manipulators.

Supplementary Materials: The video of the trajectory tracking in software simulation (MATLAB-Simulink) and Implementation on hardware are presented at https://nima-maghooli.github.io/.

Author Contributions: N. Maghooli designed and evaluated the case studies and implemented and developed the RL-TJ and RL-MTJ control strategies. He also helped write parts of the paper and implemented the control system in MATLAB-Simulink. O. Mahdizadeh wrote a major part of the paper and helped develop the controllers in MATLAB-Simulink and with implementation. M. Bajelani reviewed the paper and helped develop methods. S. Ali A. Moosavian is the senior author of the paper. He conceptualized, advised, and regularly discussed research questions and results with all authors. He also edited the paper. All authors contributed to conceptualizing and revising the paper through regular discussions on the theory and implementation details.

Funding: This research received no external funding.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: MATLAB simulation codes for modeling are shared on the Continuum Robotics Laboratory GitHub page (https://github.com/ContinuumRoboticsLab/tdcr-modeling). Control system design codes using Deep-RL will be added to https://nima-maghooli.github.io/.

Acknowledgments: Special thanks to Dr. Mohammad Dehghani and Armin Shahsavari for previous works on continuum robots, transferring experiences, and basic codes of implementation on hardware.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

In this section, Forward Kinematics Equations (direct mapping from the joint space to the task space) and the Jacobian Matrix are presented. These equations are derived from the Constant Curvature assumption and the relations between the spaces described in [27], [28]. According to the previous explanations, the necessary condition for control in the task space is to have the Jacobian matrix to map the generalized forces from the task space to the joint space. Since this research is based on using the closed form for the Jacobian matrix, it is necessary to ensure their validity before using these equations for force distribution.

- **The Forward Kinematics Equations:** This section presents the forward kinematics equations that establish the mapping from the joint space to the task space. It should be noted that in finding these relationships, modeling with the assumption of constant curvature was used for each joint, and by removing the configuration space and its parameters (s, κ, ϕ), a direct mapping from the joint space ($\boldsymbol{L} = [l_1 \quad l_2 \quad l_3 \quad l_4 \quad l_5 \quad l_6]^T$) to the work-space ($\boldsymbol{X} = [x \quad y \quad z]^T$) was obtained**.** In order to summarize more, in the above sentence, expressions with the symbol σ_i have been used, and these terms are displayed below.

End-Effector position in X-direction:

$$x = \frac{(\cos(\sigma_4) - 1)\,\sigma_8 \left(\frac{3\,l_1}{400} + \frac{3\,l_2}{400} + \frac{3\,l_3}{400}\right)}{2\,\sigma_5\,\sigma_7} - \frac{\sin(\sigma_4)\,\sin(\sigma_3)\,\sigma_8\,\sigma_1}{2\,\sigma_5\,\sigma_6} - \frac{\sqrt{3}\,\sin(\sigma_2)\,(\cos(\sigma_3) - 1)\,(l_2 - 2\,l_1 + l_3)\,\sigma_1}{2\,\sigma_5\,\sigma_6} + \frac{\cos(\sigma_4)\,\cos(\sigma_2)\,(\cos(\sigma_3) - 1)\,\sigma_8\,\sigma_1}{2\,\sigma_5\,\sigma_6}$$

$$\sigma_1 = \frac{3\,l_4}{400} + \frac{3\,l_5}{400} + \frac{3\,l_6}{400} \quad , \quad \sigma_2 = \text{atan2}(\sigma_9, 3\,l_3 - 3\,l_2) - \text{atan2}\left(\sqrt{3}\,l_5 - 2\sqrt{3}\,l_4 + \sqrt{3}\,l_6, 3\,l_6 - 3\,l_5\right)$$

$$\sigma_3 = 18\,\text{asin}\left(\frac{400\,\sigma_6}{81}\right) \quad , \quad \sigma_4 = 18\,\text{asin}\left(\frac{400\,\sigma_7}{81}\right) \quad , \quad \sigma_5 = \sqrt{{\sigma_9}^2 + {\sigma_8}^2}$$

$$\sigma_6 = \sqrt{{l_4}^2 - l_4\,l_5 - l_4\,l_6 + {l_5}^2 - l_5\,l_6 + {l_6}^2} \quad , \quad \sigma_7 = \sqrt{{l_1}^2 - l_1\,l_2 - l_1\,l_3 + {l_2}^2 - l_2\,l_3 + {l_3}^2}$$

$$\sigma_8 = 3\,l_2 - 3\,l_3 \quad , \quad \sigma_9 = \sqrt{3}\,l_2 - 2\sqrt{3}\,l_1 + \sqrt{3}\,l_3$$

End-Effector position in Y-direction:

$$y = \frac{\sqrt{3}\,\sin(\sigma_5)\,\sin(\sigma_4)\,\sigma_2\,\sigma_1}{2\,\sigma_6\,\sigma_7} - \frac{\sin(\sigma_3)\,(\cos(\sigma_4) - 1)\,\sigma_{10}\,\sigma_1}{2\,\sigma_6\,\sigma_7} - \frac{\sqrt{3}\,(\cos(\sigma_5) - 1)\,\sigma_2 \left(\frac{3\,l_1}{400} + \frac{3\,l_2}{400} + \frac{3\,l_3}{400}\right)}{2\,\sigma_6\,\sigma_8} - \frac{\sqrt{3}\,\cos(\sigma_5)\,\cos(\sigma_3)\,(\cos(\sigma_4) - 1)\,\sigma_2\,\sigma_1}{2\,\sigma_6\,\sigma_7}$$

$$\sigma_1 = \frac{3\,l_4}{400} + \frac{3\,l_5}{400} + \frac{3\,l_6}{400} \quad , \quad \sigma_2 = l_2 - 2\,l_1 + l_3$$

$$\sigma_3 = \text{atan2}(\sigma_9, 3\,l_3 - 3\,l_2) - \text{atan2}\left(\sqrt{3}\,l_5 - 2\sqrt{3}\,l_4 + \sqrt{3}\,l_6, 3\,l_6 - 3\,l_5\right)$$

$$\sigma_4 = 18\,\text{asin}\left(\frac{400\,\sigma_7}{81}\right) \quad , \quad \sigma_5 = 18\,\text{asin}\left(\frac{400\,\sigma_8}{81}\right)$$

$$\sigma_6 = \sqrt{{\sigma_9}^2 + {\sigma_{10}}^2} \quad , \quad \sigma_7 = \sqrt{{l_4}^2 - l_4\,l_5 - l_4\,l_6 + {l_5}^2 - l_5\,l_6 + {l_6}^2}$$

$$\sigma_8 = \sqrt{{l_1}^2 - l_1 l_2 - l_1 l_3 + {l_2}^2 - l_2 l_3 + {l_3}^2} \quad , \quad \sigma_9 = \sqrt{3}\, l_2 - 2\sqrt{3}\, l_1 + \sqrt{3}\, l_3 \quad , \quad \sigma_{10} = 3\, l_2 - 3\, l_3$$

End-Effector position in Z-direction:

$$z = \frac{\sin(\sigma_3)\left(\frac{3\, l_1}{400} + \frac{3\, l_2}{400} + \frac{3\, l_3}{400}\right)}{2\, \sigma_5} + \frac{\cos(\sigma_3)\, \sin(\sigma_2)\, \sigma_1}{2\, \sigma_4}$$
$$+ \frac{\sin(\sigma_3)\cos\left(\text{atan2}\left(\sqrt{3}\, l_2 - 2\sqrt{3}\, l_1 + \sqrt{3}\, l_3, 3\, l_3 - 3\, l_2\right) - \text{atan2}\left(\sqrt{3}\, l_5 - 2\sqrt{3}\, l_4 + \sqrt{3}\, l_6, 3\, l_6 - 3\, l_5\right)\right)(\cos(\sigma_2) - 1)\, \sigma_1}{2\, \sigma_4}$$

$$\sigma_1 = \frac{3\, l_4}{400} + \frac{3\, l_5}{400} + \frac{3\, l_6}{400} \quad , \quad \sigma_2 = 18\, \text{asin}\left(\frac{400\, \sigma_4}{81}\right) \quad , \quad \sigma_3 = 18\, \text{asin}\left(\frac{400\, \sigma_5}{81}\right)$$

$$\sigma_4 = \sqrt{{l_4}^2 - l_4 l_5 - l_4 l_6 + {l_5}^2 - l_5 l_6 + {l_6}^2} \quad , \quad \sigma_5 = \sqrt{{l_1}^2 - l_1 l_2 - l_1 l_3 + {l_2}^2 - l_2 l_3 + {l_3}^2}$$

Before dealing with the Jacobian matrix, the verification for the forward kinematics equations is checked first because the Jacobian is obtained based on the partial derivation of these equations. If they are verified, the next step can be followed. To verify the forward kinematics equations, harmonic inputs with phase difference are considered for the angle of the servo-motors, and the momentary position of the end-effector is also recorded by the camera. Finally, the harmonic inputs with phase difference are given as input to the forward kinematics equations, and their output is compared with the output of the camera. Figure A1 compares camera output and forward kinematics equations for each end-effector coordinate in the task space. According to the figure, forward kinematics equations (due to the existence of structured and unstructured uncertainties of the system) have appropriate accuracy. They can be used to find the Jacobian matrix.

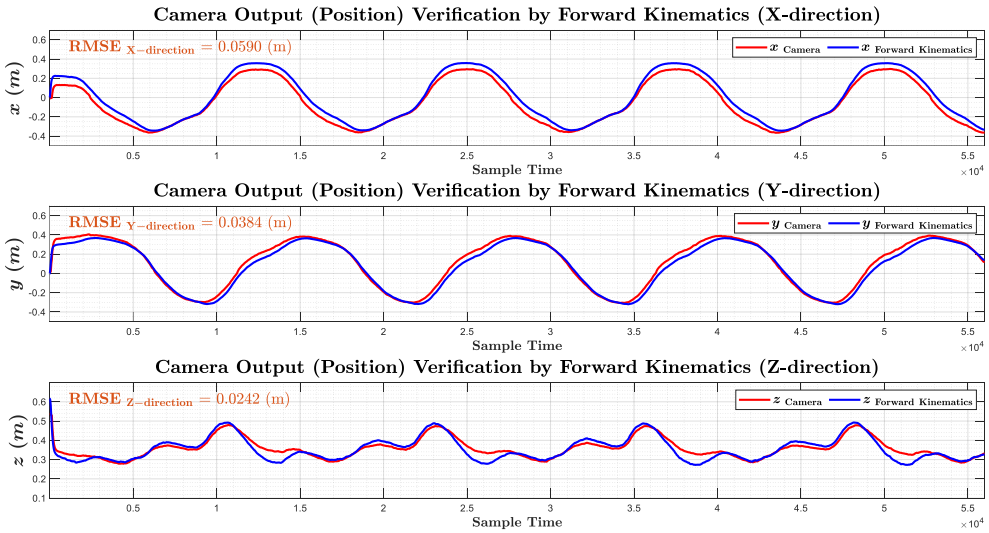


Figure A1. Camera output (Position) verification by forward kinematics equations.

- **The Jacobian Matrix:** In this section, the linear Jacobian matrix that establishes the mapping between the rate of changes of the joint space and the task space in the form of $\dot{\boldsymbol{X}} = \boldsymbol{J}_L \dot{\boldsymbol{L}}$ is presented. In general, this matrix is calculated in the form of equation 5. And finally, it is obtained in the form of the matrix described in expression 6. In the following, the element of the first row and first column of this matrix is obtained by analytical patrial derivation, and other elements are also obtained in a similar way. To summarize more (such as forward kinematics relations) in the above expression, expressions with the symbol σ_i are used, and these terms are shown below.

$$
\begin{aligned}
J_{11} = {} & \frac{3\,(\cos(\sigma_5)-1)\,\sigma_{13}}{800\,\sqrt{\sigma_{12}}\,\sqrt{\sigma_{11}}} + \frac{\sqrt{3}\,\sin(\sigma_3)\,(\cos(\sigma_6)-1)\,\sigma_1}{\sqrt{\sigma_{12}}\,\sigma_9} - \frac{\sin(\sigma_3)\,(\cos(\sigma_6)-1)\,{\sigma_8}^2\,\sigma_1}{\sigma_4\,\sigma_9} \\
& + \frac{(\cos(\sigma_5)-1)\,\sigma_{13}\,\sigma_8\,\sigma_2}{4\,\sqrt{\sigma_{12}}\,{\sigma_{11}}^{3/2}} + \frac{\sqrt{3}\,(\cos(\sigma_5)-1)\,\sigma_{13}\,\sigma_8\,\sigma_2}{3\,\sigma_4\,\sqrt{\sigma_{11}}} \\
& - \frac{1800\,\sin(\sigma_5)\,\sigma_{13}\,\sigma_8\,\sigma_2}{\sqrt{\sigma_{12}}\left(-{l_1}^2 + l_1\,l_2 + l_1\,l_3 - {l_2}^2 + l_2\,l_3 - {l_3}^2\right)\sigma_{10}} \\
& - \frac{3\,\cos(\sigma_3)\,(\cos(\sigma_6)-1)\,\sigma_{13}\,\sigma_8\,\sigma_1}{{\sigma_{12}}^{3/2}\,\sigma_9} \\
& - \frac{9\sqrt{3}\,\cos(\sigma_5)\,\sin(\sigma_3)\,(l_2-l_3)^2\,(\cos(\sigma_6)-1)\,\sigma_1}{({\sigma_{14}}^2 + 9\,(l_2-l_3)^2)^{3/2}\,\sigma_9} - \frac{\sqrt{3}\,\sin(\sigma_5)\,\sin(\sigma_6)\,\sigma_{13}\,\sigma_8\,\sigma_1}{3\,\sigma_4\,\sigma_9} \\
& + \frac{\sqrt{3}\,\cos(\sigma_5)\,\cos(\sigma_3)\,(\cos(\sigma_6)-1)\,\sigma_{13}\,\sigma_8\,\sigma_1}{3\,\sigma_4\,\sigma_9} + \frac{1800\,\cos(\sigma_5)\,\sin(\sigma_6)\,\sigma_{13}\,\sigma_8\,\sigma_1}{\sigma_7} \\
& + \frac{1800\,\sin(\sigma_5)\,\cos(\sigma_3)\,(\cos(\sigma_6)-1)\,\sigma_{13}\,\sigma_8\,\sigma_1}{\sigma_7}
\end{aligned}
$$

$$\sigma_1 = \frac{3\,l_4}{400} + \frac{3\,l_5}{400} + \frac{3\,l_6}{400} \quad , \quad \sigma_2 = \frac{3\,l_1}{400} + \frac{3\,l_2}{400} + \frac{3\,l_3}{400}$$

$$\sigma_3 = \text{atan2}(\sigma_{14}, 3\,l_3 - 3\,l_2) - \text{atan2}\left(\sqrt{3}\,l_5 - 2\sqrt{3}\,l_4 + \sqrt{3}\,l_6, 3\,l_6 - 3\,l_5\right)$$

$$\sigma_4 = ({\sigma_8}^2 + 3\,(l_2 - l_3)^2)^{3/2}$$

$$\sigma_5 = 18\,\text{asin}\left(\frac{400\,\sqrt{\sigma_{11}}}{81}\right) \quad , \quad \sigma_6 = 18\,\text{asin}\left(\frac{400\,\sigma_9}{81}\right)$$

$$\sigma_7 = \sqrt{\sigma_{12}}\,\sqrt{\sigma_{11}}\,\sigma_9\,\sigma_{10} \quad , \quad \sigma_8 = l_2 - 2\,l_1 + l_3$$

$$\sigma_9 = \sqrt{{l_4}^2 - l_4\,l_5 - l_4\,l_6 + {l_5}^2 - l_5\,l_6 + {l_6}^2}$$

$$\sigma_{10} = \sqrt{-160000\,{l_1}^2 + 160000\,l_1\,l_2 + 160000\,l_1\,l_3 - 160000\,{l_2}^2 + 160000\,l_2\,l_3 - 160000\,{l_3}^2 + 6561}$$

$$\sigma_{11} = {l_1}^2 - l_1\,l_2 - l_1\,l_3 + {l_2}^2 - l_2\,l_3 + {l_3}^2 \quad , \quad \sigma_{12} = {\sigma_{14}}^2 + {\sigma_{13}}^2$$

$$\sigma_{13} = 3\,l_2 - 3\,l_3 \quad , \quad \sigma_{14} = \sqrt{3}\,l_2 - 2\sqrt{3}\,l_1 + \sqrt{3}\,l_3$$

After verifying the forward kinematics equations, the Jacobian matrix (which maps the rate of change of the effective length of the robot's tendons and its final executive speed) should be verified. To do this, from the harmonic inputs used in the previous part, the derivative is taken with respect to time and considered as the input for the Jacobian matrix, and its output is compared with the derivative of the camera output, which is equivalent to the instantaneous speed of the end-effector. Verification results are presented in figure A2.

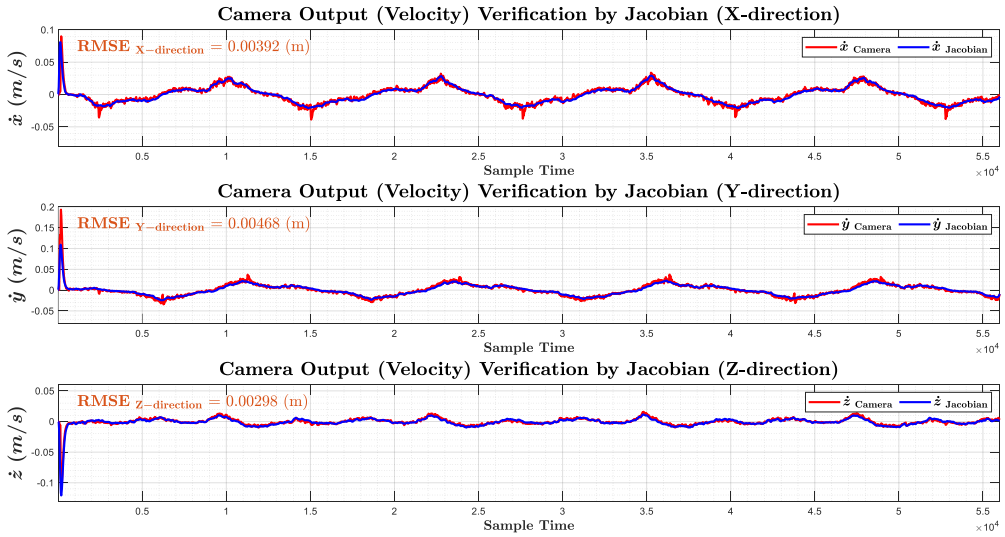


Figure A2. Camera output (Velocity) verification by Jacobian matrix

References

[1] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control Strategies for Soft Robotic Manipulators: A Survey," *Soft Robot*, vol. 5, no. 2, pp. 149–163, Apr. 2018, doi: 10.1089/SORO.2017.0007/ASSET/IMAGES/LARGE/FIGURE12.JPEG.

[2] X. Wang, Y. Li, and K. W. Kwok, "A Survey for Machine Learning-Based Control of Continuum Robots," *Front Robot AI*, vol. 8, p. 730330, Sep. 2021, doi: 10.3389/FROBT.2021.730330/BIBTEX.

[3] M. T. Chikhaoui and J. Burgner-Kahrs, "Control of Continuum Robots for Medical Applications: State of the Art," in *ACTUATOR 2018; 16th International Conference on New Actuators*, 2018, pp. 1–11.

[4] K. Haiya, S. Komada, and J. Hirai, "Tension control for tendon mechanisms by compensation of nonlinear spring characteristic equation error," *International Workshop on Advanced Motion Control, AMC*, pp. 42–47, 2010, doi: 10.1109/AMC.2010.5464047.

[5] H. In, H. Lee, U. Jeong, B. B. Kang, and K. J. Cho, "Feasibility study of a slack enabling actuator for actuating tendon-driven soft wearable robot without pretension," *Proc IEEE Int Conf Robot Autom*, vol. 2015-June, no. June, pp. 1229–1234, Jun. 2015, doi: 10.1109/ICRA.2015.7139348.

[6] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, Sep. 2015, Accessed: Aug. 19, 2023. [Online]. Available: https://arxiv.org/abs/1509.02971v6

[7] J. J. Craig, "Introduction to robotics : mechanics and control," p. 438.

[8] S. A. A. Moosavian and E. Papadopoulos, "Modified transpose Jacobian control of robotic systems," *Automatica*, vol. 43, no. 7, pp. 1226–1233, Jul. 2007, doi: 10.1016/J.AUTOMATICA.2006.12.029.

[9] S. A. A. Moosavian and E. Papadopoulos, "Control of space free-flyers using the modified transpose Jacobian algorithm," *IEEE International Conference on Intelligent Robots and Systems*, vol. 3, pp. 1500–1505, 1997, doi: 10.1109/IROS.1997.656557.

[10] H. Yuan, L. Zhou, and W. Xu, "A comprehensive static model of cable-driven multi-section continuum robots considering friction effect," *Mech Mach Theory*, vol. 135, pp. 130–149, May 2019, doi: 10.1016/J.MECHMACHTHEORY.2019.02.005.

[11] I. S. Godage, D. T. Branson, E. Guglielmino, and D. G. Caldwell, "Path planning for multisection continuum arms," *2012 IEEE International Conference on Mechatronics and Automation, ICMA 2012*, pp. 1208–1213, 2012, doi: 10.1109/ICMA.2012.6283423.

[12] A. Amouri, A. Zaatri, and C. Mahfoudi, "Dynamic Modeling of a Class of Continuum Manipulators in Fixed Orientation," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 91, no. 3–4, pp. 413–424, Sep. 2018, doi: 10.1007/S10846-017-0734-Z/METRICS.

[13] M. Karimi and S. A. A. Moosavian, "Modified transpose effective Jacobian law for control of underactuated manipulators," *Advanced Robotics*, vol. 24, no. 4, pp. 605–626, Mar. 2010, doi: 10.1163/016918610X487135.

[14] S. A. A. Moosavian, M. Alghooneh, and A. Takhmar, "Modified transpose jacobian control of a biped robot," *Proceedings of the 2007 7th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS 2007*, pp. 282–287, 2007, doi: 10.1109/ICHR.2007.4813881.

[15] M. Khorram and S. A. A. Moosavian, "Modified Jacobian transpose control of a quadruped robot," *International Conference on Robotics and Mechatronics, ICROM 2015*, pp. 67–72, Dec. 2015, doi: 10.1109/ICROM.2015.7367762.

[16] A. K. Khalaji and S. A. A. Moosavian, "Modified transpose Jacobian control of a tractor-trailer wheeled robot," *Journal of Mechanical Science and Technology*, vol. 29, no. 9, pp. 3961–3969, Sep. 2015, doi: 10.1007/S12206-015-0841-3/METRICS.

[17] R. Khajepour, H. Khajvand, H. Daniali, M. Rastin, and S. A. A. Moosavian, "Explicit dynamic and MTJ Control of a 3-RRS Parallel Manipulator with mass Compensator," *5th RSI International Conference on Robotics and Mechatronics, IcRoM 2017*, pp. 552–557, Sep. 2018, doi: 10.1109/ICROM.2017.8466192.

[18] S. A. A. Moosavian, A. Janati, and M. H. Ghazikhani, "Object manipulation by two humanoid robots using MTJ control," *2011 IEEE International Conference on Mechatronics and Automation, ICMA 2011*, pp. 1286–1290, 2011, doi: 10.1109/ICMA.2011.5985847.

[19] S. A. A. Moosavian and M. Karimi, "Tuned Modified Transpose Jacobian Control of Robotic Systems," *Interdisciplinary Mechatronics: Engineering Science and Research Development*, pp. 255–275, May 2013, doi: 10.1002/9781118577516.CH11.

[20] M. A. Davari, S. A. A. Moosavian, A. Ghaffari, and M. T. Masouleh, "Adaptive Modified Transpose Jacobian Control of a Two-fingered Robotic Hand," *9th RSI International Conference on Robotics and Mechatronics, ICRoM 2021*, pp. 210–216, 2021, doi: 10.1109/ICROM54204.2021.9663452.

[21] M. Karimi and S. A. A. Moosavian, "Learning-based modified transpose jacobian control of robotic manipulators," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, pp. 1290–1295, 2008, doi: 10.1109/AIM.2008.4601848.

[22] M. Bajelani, S. A. Khalilpour, M. I. Hosseini, S. A. A. Moosavian, and H. D. Taghirad, "Time-Delay Learning-Based Controller for Fully-Constrained Cable-Driven Parallel Robots," *2022 8th International Conference on Control, Instrumentation and Automation, ICCIA 2022*, 2022, doi: 10.1109/ICCIA54998.2022.9737194.

[23] R. S. Sutton, A. G. Barto, and R. J. Williams, "Reinforcement Learning is Direct Adaptive Optimal Control," *IEEE Control Syst*, vol. 12, no. 2, pp. 19–22, 1992, doi: 10.1109/37.126844.

[24] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int J Rob Res*, vol. 32, no. 11, pp. 1238–1274, Aug. 2013, doi: 10.1177/0278364913495721.

[25] S. Satheeshbabu, N. K. Uppalapati, T. Fu, and G. Krishnan, "Continuous Control of a Soft Continuum Arm using Deep Reinforcement Learning," *2020 3rd IEEE International Conference on Soft Robotics, RoboSoft 2020*, pp. 497–503, May 2020, doi: 10.1109/ROBOSOFT48309.2020.9116003.

[26] A. Centurelli, L. Arleo, A. Rizzo, S. Tolu, C. Laschi, and E. Falotico, "Closed-Loop Dynamic Control of a Soft Manipulator Using Deep Reinforcement Learning," *IEEE Robot Autom Lett*, vol. 7, no. 2, pp. 4741–4748, Apr. 2022, doi: 10.1109/LRA.2022.3146903.

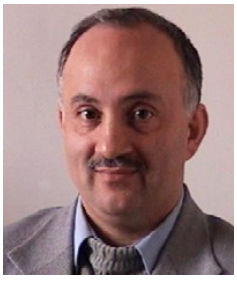
[27] P. Rao, Q. Peyron, S. Lilge, and J. Burgner-Kahrs, "How to Model Tendon-Driven Continuum Robots and Benchmark Modelling Performance," *Front Robot AI*, vol. 7, p. 630245, Feb. 2021, doi: 10.3389/FROBT.2020.630245/BIBTEX.

[28] R. J. Webster and B. A. Jones, "Design and Kinematic Modeling of Constant Curvature Continuum Robots: A Review," *Int J Rob Res*, vol. 29, no. 13, pp. 1661–1683, Jun. 2010, doi: 10.1177/0278364910368147.

Nima Maghooli received his B.Sc. degree in Mechanical Engineering from the University of Tehran, Tehran, Iran, in 2020. He received his M.Sc. degree in Mechanical Engineering (majoring in Dynamics and Control) at the K. N. Toosi University of Technology, Tehran, Iran, in 2023. He is currently a research assistant at the Advanced Robotics and Automated Systems (ARAS). His research interests are data-driven modeling and intelligent control applied to robotic systems.

Omid Mahdizadeh received his B.Sc. degree in Electrical Engineering - Control from Shahed University, Tehran, Iran, in 2015. He received his M.Sc. degree in Mechatronic Engineering from the Electrical Department at K. N. Toosi University of Technology, Tehran, Iran, in 2019. Omid is currently a Ph.D. student researcher in the IDEA Lab as a part of the Mechanical Engineering Department at the University of Alberta. His current research interests include Rehabilitation, Robotics, Mechatronics, Data fusion, model predictive control systems, and Artificial Intelligence.

Mohammad Bajelani received his B.Sc. and M.Sc. degrees in Aerospace and Mechatronics Engineering from K. N. Toosi University of Technology, Tehran, Iran, in 2020 and 2022, respectively. He is currently a Ph.D. student in the Data-Driven Modeling and Control group at the University of British Columbia (UBC). He is currently working on developing data-driven and learning-based control methods for safety-critical applications.



S. Ali. A. Moosavian received his B.Sc. degree in 1986 from the Sharif University of Technology and his M.Sc. degree in 1990 from Tarbiat Modares University (both in Tehran), and his Ph.D. degree in 1996 from McGill University (Montreal, Canada), all in Mechanical Engineering. He has been a Professor in the Mechanical Engineering Department at K. N. Toosi University of Technology (KNTU) in Tehran since 1997. His research interests are in the areas of dynamics modeling and motion/impedance control of terrestrial, legged, continuum and space robotic systems. He has published more than 350 articles in peer-reviewed journals and conference proceedings. He is one of the Founders of the ARAS Research Group and director of the Center of Excellence in Robotics and Control at KNTU.