Proceedings of the 12th RSI International Conference on Robotics and Mechatronics (ICRoM 2024), Dec. 17-19, 2024, Tehran, Iran

Control of Cable Driven Parallel Robots Through Deep Reinforcement Learning

D. A. Nejad*, A. Sharifi*, M. R. Dindarloo*, A. S. Mirjalili*, S. A. Khalilpour*, H. D. Taghirad*
* Applied Robotics and AI Solutions (ARAS), K. N. Toosi University of Technology, Tehran, Iran.
Email: *taghirad@kntu.ac.ir

Abstract—**Cable-driven parallel robots (CDPRs) pose significant challenges for precise control due to their complex cable dynamics and environmental uncertainties. This paper presents the implementation of Deep Reinforcement Learning (DRL) techniques to control a planar CDPR using the Deep Deterministic Policy Gradient (DDPG) algorithm. Leveraging a cable robot simulator for Reinforcement Learning (RL) allows for safe exploration, faster iterations, and cost-effective training by efficiently handling the robot's complex dynamics. Additionally, we utilized the MuJoCo physics engine to accurately simulate the nonlinear behavior of the cable robot, offering a robust platform for training and validating RL-based control strategies. The performance of this approach is evaluated using prescribed paths, and the results effectively demonstrate strong tracking accuracy and the overall effectiveness of the control strategy.**

Index Terms—**Cable-Driven Parallel Robot, Deep Deterministic Policy Gradient, Deep Reinforcement Learning, MuJoCo Simulator, Trajectory Tracking.**

I. Introduction

Cable-Driven Parallel Robots (CDPRs) utilize cables instead of rigid links for actuation, offering a lightweight and cost-effective solution for actuating moving platforms. Their parallel structure and flexible cables enable fast operation, the ability to handle heavy loads, and access to a large workspace. These characteristics make CDPRs ideal for applications such as virtual reality [1], art [2], material handling, and rehabilitation devices [3].

In addition to the numerous advantages offered by CDPRs, however, they come with their own set of challenges, primarily due to the flexibility of cables. This flexibility complicates the kinematic and dynamic modeling of CDPRs, with various uncertainties further impacting their overall performance and control. As a result, traditional control methods often struggle to maintain precision and reliability [4]. Achieving stable and accurate control in these conditions requires careful tuning of control parameters, which must often be adjusted as the reference input changes [5], [6].

To address these challenges, advanced control algorithms have been developed to enhance performance in complex scenarios. Recently, learning-based control methods have gained significant attention for their ability to manage dynamic systems and adapt to uncertain environments without relying on precise models [7], [8]. Among these methods, reinforcement learning (RL) has proven particularly effective, as it autonomously discovers optimal control strategies through interaction with the environment and reward-based feedback. This adaptive approach enables RL to continuously refine control inputs, effectively managing both explicit and implicit uncertainties.

Recent advances show that RL can surpass traditional methods in various robotic applications, providing a flexible, scalable, and model-free solution [9], [10]. For instance, [11] applied deep reinforcement learning (DRL) to robotic manipulation tasks, such as object pick-and-place, where RL-based controllers adapted to changing conditions and optimized precision in real-time, outperforming traditional PID controllers. Additionally, Kalashnikov et al. demonstrated that Q-learning significantly improved success rates in real-world robotic grasping tasks compared to classical control strategies in dynamic environments [12].

RL-based controllers have shown significant potential in robotic manipulation and autonomous navigation. For example, [13] applied RL to control robotic arms in industrial settings, enhancing their ability to perform precision tasks such as assembly and material handling in dynamic environments. Similarly, [14] employed RL techniques for mobile robots, achieving improved path planning and obstacle avoidance. Despite these advancements, the utilization of RL-based controllers for CDPRs has been largely overlooked.

While RL-based controllers have demonstrated considerable potential in robotic manipulation and autonomous navigation, applying these techniques to real robots presents several challenges. The inherent uncertainties and safety risks involved in the training process complicate real-world implementation. A practical approach to mitigate these challenges is to utilize realistic simulators that closely mimic real-world conditions, enabling training in a controlled virtual environment. Subsequently, transfer learning can be employed to adapt the learned control strategies to physical robots, facilitating smoother integration with real-world dynamics [15]. However, the lack of robust simulators specifically designed for cable-driven systems remains a significant barrier to developing effective RL-based control solutions for CDPRs.

Several studies have explored RL techniques for controlling CDPRs using simulators. For example, Sancak et al. [16] applied a DDPG algorithm in a simulated environment to control the position of a planar CDPR, optimizing cable tensions and enhancing tracking accuracy. Similarly, Xie et al. [17] utilized deep Q-learning (DQN) in MATLAB Simulink for a 3-degree-of-freedom (DOF) CDPR designed for rehabilitation, achieving smoother trajectories and reduced cable tension. Nomanfar et al. [18] implemented a multi-agent DDPG framework in MATLAB Simulink to simulate agent collaboration and facilitate transfer learning for CDPR control. However, these studies relied on proprietary simulators like MATLAB Simulink, which limit flexibility due to difficulties in integrating with modern RL libraries in Python. Moreover, the use of robot URDF definitions in simulators has gained traction in the robotics field, as they facilitate easier integration and modeling.

In this paper, we address the limitations of existing simulators by presenting a planar CDPR model in MuJoCo (Multi-Joint Dynamics with Contact), an open-source simulator that supports URDF. MuJoCo enables seamless integration with Python APIs and facilitates the implementation of various RL algorithms, providing a flexible and accessible platform for developing and testing RL-based control strategies for CDPRs. To the best of our knowledge, this is the first time that a CDPR has been trained as a reinforcement learning agent within the MuJoCo simulator. By employing the DDPG algorithm, we eliminate the need for system identification and Jacobian matrix calculations, which are typically complex and sensitive to uncertainties, especially in CDPRs. DDPG allows direct learning of control policies without relying on precise system models, overcoming these challenges. We evaluated our approach on a circular trajectory, demonstrating strong tracking accuracy with minimal error, confirming the effectiveness of the proposed setup.

The remainder of this paper is organized as follows: Section II provides an overview of trajectory planning in robotic control and the reinforcement learning framework, with a focus on the DDPG algorithm for continuous control in CDPRs. Additionally, the MuJoCo simulation setup and the robot model are described. Section III presents the experimental setup, results, and analysis, including the evaluation of trajectory tracking performance. Section IV discusses potential future work. Finally, Section V concludes with a summary of the contributions and emphasizes the effectiveness of the proposed RL-based control strategy for CDPRs in dynamic environments.

II. Methodology

A. Trajectory Planning in Robotic Control

The choice of a suitable trajectory is fundamental in evaluating the performance of robotic systems, particularly for tracking tasks in dynamic environments. An appropriate trajectory ensures smooth, continuous movement, reducing deviations and enhancing precision. Both Bezier curves and circular trajectories are commonly used in such tasks due to their inherent flexibility and stability. Bezier curves, defined by control points, allow for the creation of adaptable, non-linear paths, making them effective for navigating complex environments. This has been demonstrated in applications like trajectory planning for industrial robots [19] and in reinforcement learning-based control for CDPRs [20]. Similarly, circular trajectories have been utilized to evaluate tracking performance in CDPRs, as shown by [21] and [22] providing smooth motion that is particularly suitable for assessing the performance of trajectory tracking in dynamic environments. In our study, we utilize a circular trajectory to guide the motion of the robot.

B. Reinforcement Learning

The goal of Deep Reinforcement Learning (DRL) in control systems is to derive a policy that maximizes a predefined objective function. Once trained, the policy can be deployed in real-time, providing an efficient solution to complex optimization problems that are typically computationally intensive and may not have analytical solutions. This problem can be formulated as a Markov Decision Process (MDP), where the system is represented by a tuple $M = (s, a, p, r, \gamma)$. In this framework, s denotes the set of possible states, a the set of available actions, p the transition dynamics between states, r the reward function, and γ the discount factor. At each time step t, the agent observes the current state $s_t \in s$, selects an action $a_t \in a$, and the system transitions to a new state s_{t+1} according to the transition probability p, receiving a reward r_t. Through repeated interactions with the environment, the agent refines its control policy to optimize long-term performance.

The problem of trajectory tracking in CDPRs can be effectively addressed using this DRL framework. In this context, the RL agent aims to learn an optimal policy that minimizes trajectory tracking errors while ensuring smooth and stable robot motion. At each time step, the state of the system is represented by a set of key features that describe the robot's condition. These features, obtained from the MuJoCo simulator, include the end-effector's position, velocity, position error, velocity error, and cable lengths. These are encapsulated in the state vector $\mathbf{s}_t$ as follows:

$$\mathbf{s}_t = \begin{bmatrix} \mathbf{x}_e(t) & \dot{\mathbf{x}}_e(t) & \mathbf{e}_x(t) & \dot{\mathbf{e}}_x(t) & \Delta l_i(t) \end{bmatrix} \tag{1}$$

where $\mathbf{x}_e(t)$ and $\dot{\mathbf{x}}_e(t)$ represent the position and velocity of the end-effector, which are crucial for the RL agent to understand the robot's spatial location and movement dynamics. In a planar CDPR, the y-axis remains constant, restricting the robot's movement to the x and z directions. The terms $\mathbf{e}_x(t)$ and $\dot{\mathbf{e}}_x(t)$ represent the position and velocity errors, providing real-time feedback on deviations from the desired state, which is key for accurate trajectory tracking and smooth motion. Lastly, $\Delta l_i(t)$ represents the incremental changes in cable lengths, ensuring the cables remain properly tensioned by the model itself to maintain control over the robot's dynamics.

The control actions taken by the RL agent are represented by the action vector $\mathbf{a}_t$, which defines the adjustments to the cable lengths at each time step:

$$\mathbf{a}_t = \begin{bmatrix} \Delta l_1(t) & \Delta l_2(t) & \Delta l_3(t) \end{bmatrix} \tag{2}$$

These actions directly influence the position and movement of the end-effector, allowing the agent to fine-tune the robot's trajectory in real-time. By continuously adjusting the cable lengths based on the state feedback, the RL agent ensures that the robot follows the desired trajectory while maintaining system stability and avoiding abrupt movements.

The RL agent's primary objective is to follow a predefined reference trajectory, which in this case is a circular path. The trajectory is defined by the radius of the circle, the angular frequency of the motion (denoted by ω), and a constant value that keeps the position along the y-axis fixed, confining movement to the x and z directions. The end-effector moves along this circular path continuously, allowing the RL agent to adjust and optimize its control policy to maintain precise trajectory tracking.

In addition to position tracking, the velocity of the end-effector plays a critical role in ensuring smooth movement. The velocity varies sinusoidally along the x and z directions, with the magnitude determined by the radius and angular frequency ω. Throughout the motion, the y-axis remains constant. By monitoring the velocity, the RL agent ensures that the robot maintains a consistent speed, avoiding sudden changes that could disrupt system stability.

The reward function is a vital component of the RL framework, designed to guide the agent in minimizing both position and velocity errors. It is structured as follows:

$$r(\mathbf{s}, \mathbf{a}) = -\alpha \|\mathbf{e}_x\|_2^2 - \beta \|\dot{\mathbf{e}}_x\|_2^2 \tag{3}$$

The first term penalizes the squared position error $\mathbf{e}_x$, ensuring that the agent minimizes deviations from the desired position. The second term penalizes the squared velocity error $\dot{\mathbf{e}}_x$, promoting smooth movement and discouraging abrupt speed changes. By tuning the parameters α and β, the reward function balances the trade-off between position accuracy and velocity smoothness, enabling the agent to optimize its control policy through repeated interactions with the environment. This leads to precise trajectory tracking and stable, continuous robot motion.

Deep Deterministic Policy Gradient (DDPG): Due to the continuous nature of both the action space (cable length adjustments) and the state space (robot position, velocity, and errors), the DDPG algorithm is particularly well-suited for this task. DDPG, introduced by Lillicrap et al. [23], is a model-free, off-policy RL algorithm that excels in environments with continuous action spaces, making it highly effective for trajectory tracking in CDPRs. By combining deterministic policy gradients with Q-learning, DDPG enables the agent to learn and optimize control policies in continuous spaces.

DDPG employs an actor-critic architecture, where the actor network $\mu(s|\theta^\mu)$ generates actions based on the current state s, and the critic network $Q(s, a|\theta^Q)$ estimates the Q-value, which evaluates the effectiveness of those actions. The actor network is updated using deterministic policy gradients, calculated as:

$$\nabla_{\theta^\mu} J \approx \mathbb{E}_{s\sim\mathcal{D}} \left[\nabla_a Q(s, a|\theta^Q)|_{a=\mu(s)} \nabla_{\theta^\mu} \mu(s|\theta^\mu) \right] \tag{4}$$

The critic network is trained by minimizing the loss function:

$$L(\theta^Q) = \mathbb{E}_{(s,a,r,s')} \left[\left(Q(s, a|\theta^Q) - y \right)^2 \right] \tag{5}$$

$$y = r + \gamma Q'(s', \mu'(s'|\theta^{\mu'})|\theta^{Q'}) \tag{6}$$

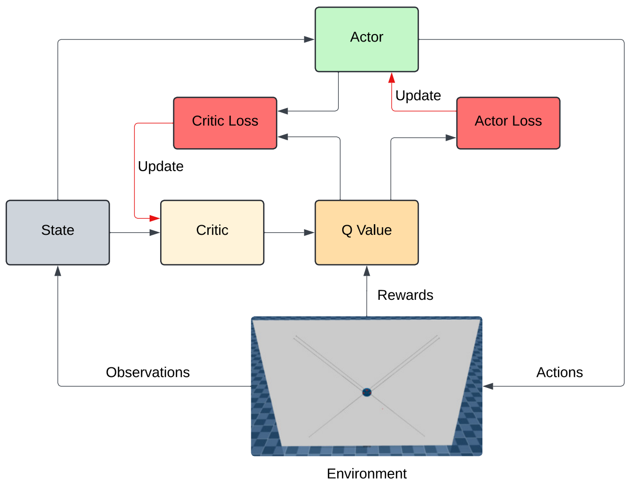

Fig. 1. The actor-critic architecture in DDPG for continuous control.

where Equation 6 is the target Q-value used to train the critic network. In addition, DDPG incorporates experience replay, where past experiences are stored in a replay buffer $\mathcal{D}$ and sampled in mini-batches. This helps break the correlation between consecutive experiences, which enhances learning stability and accelerates convergence.

Our actor-network consists of 6 layers, including 3 fully connected linear layers with ReLU activations between the first two layers and a Tanh activation at the output. The critic network comprises 5 layers, with 3 fully connected linear layers and ReLU activations between each layer. This architecture is specifically designed to handle the complexities of the continuous control problem posed by CDPRs, enabling effective policy learning and action optimization.

The diagram in Figure 1 illustrates the DDPG algorithm's actor-critic framework, highlighting the interaction between the various components involved in the learning process. The agent observes the current state from the environment, which is passed to both the actor and critic networks. The actor generates continuous actions based on the current state, which are executed in the environment, producing rewards and new observations. The critic evaluates the actions chosen by the actor by estimating the Q-value, which represents the expected return from the current state-action pair. The critic loss, computed as the difference between the predicted Q-value and the actual reward, updates the critic network, improving its ability to assess actions. Meanwhile, the actor loss is derived from the critic's feedback, updating the actor network to refine its policy over time. These updates are iterative, allowing the actor to learn better control strategies. Experience replay, though not shown in the diagram, plays a key role by storing past experiences and enabling the agent to sample mini-batches, breaking the correlation between consecutive steps and stabilizing the learning process. This combination of actor and critic updates drives the improvement of the control policy in continuous action environments like CDPRs.

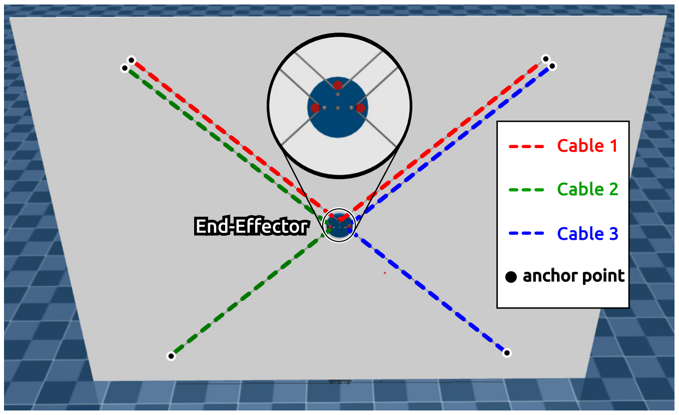

Fig. 2. The structure of the cable robot in MuJoCo environment.

C. MuJoCo Simulation Setup

We utilized the MuJoCo (Multi-Joint dynamics with Contact) simulator [24], which is widely used for physics-based modeling and dynamic simulations of robotic systems. MuJoCo offers precise physical modeling, including friction, collision detection, and joint dynamics, making it ideal for simulating the intended CDPR. Additionally, leveraging the MuJoCo API allowed us to efficiently simulate the dynamic behavior of our robotic system, implement and test control strategies and gather data for analysis.

All these advantages made MuJoCo an ideal choice for modeling our intended planar CDPR. The structure of the robot is depicted in Fig. 2. It is a medium-sized planar CDPR with a working area of $1.2m \times 1.9m$. This over-constrained robot consists of three cables, each actuated by its own actuator, along with parallelogram linkages that eliminate rotational motion of the end-effector, enabling it to operate with only two degrees of freedom (DoFs). In this setup, the cables extend from anchor points, wrap around pulleys on the end-effector, and terminate at other anchor points. By adjusting the cable lengths, the robot is capable of following predefined trajectories, such as circular and figure-eight paths.

III. Implementation Results

This section presents the results of trajectory tracking experiments on the planar CDPR using the MuJoCo simulator. The DDPG algorithm, implemented with Stable Baselines 3 [25], a PyTorch-based library known for its reliable, modular, and user-friendly implementations of popular RL algorithms, was used for training. The analysis begins with the RL agent's reward progression during training, followed by an evaluation of its accuracy in tracking predefined circular trajectories.

A. RL Agent Training

The training process for the RL agent involved completing 2000 episodes, with each episode capped at 200 timesteps to ensure efficient learning and resource management. The goal was to refine the agent's ability to follow predefined trajectories while balancing exploration and exploitation throughout the training.

The entire training took approximately 44 minutes on a system equipped with an Intel Core i9-9900K processor, 32 GB of RAM, and an NVIDIA GeForce GTX 1610 Ti GPU, leveraging the GPU's parallel processing capabilities to efficiently handle both the complex simulations and the deep learning model updates without performance bottlenecks. This implementation is made open-source for further research and development, it can be accessed at the GitHub repository linked on the page[1].

TABLE I
DDPG Model Parameters

Parameter	Value
Batch Size	256
Gamma (γ)	0.99
Tau (τ)	0.005
Buffer Size	1,000,000
Actor Learning Rate	0.001
Critic Learning Rate	0.001

The parameters chosen for the DDPG algorithm in this study as shown in Table I were carefully selected to ensure stable learning and effective performance in the continuous control task of trajectory tracking. A batch size of 256 was employed to balance between learning stability and computational efficiency, ensuring accurate gradient estimation over larger sample sizes. The discount factor ($\gamma = 0.99$) prioritizes long-term rewards without disregarding immediate gains, which is essential for tasks requiring sustained precision over time. A soft target update rate ($\tau = 0.005$) was chosen to ensure gradual and stable updates to the target networks, preventing abrupt policy shifts that could destabilize the learning process. A buffer size of 1,000,000 ensures that a large set of experiences is available for replay, improving the agent's ability to generalize across different states. Additionally, the learning rates for both the actor and critic networks were set at 0.001 to ensure steady convergence, allowing the networks to learn without overshooting or becoming stuck in suboptimal policies. These parameter choices, derived from experimentation, provide a robust foundation for the DDPG algorithm to achieve precise and smooth control of the CDPR.

Figure 3 illustrates the reward trends over the training steps, showcasing the agent's learning progression. Initially, the

[1]https://github.com/DanialNejad/CDPR_RL

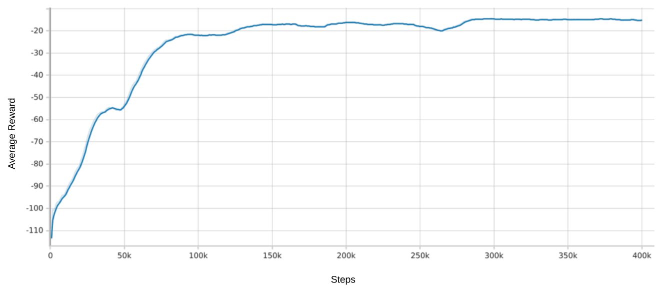

Fig. 3. Average reward progression of the RL agent over the training steps.

agent's rewards were low, reflecting a lack of understanding of the task. However, as the training progressed, the average reward steadily increased, indicating the agent's improvement in tracking the circular trajectories. The stabilization of rewards towards the later stages of training suggests that the agent successfully converged to an effective policy.

B. Trajectory Tracking Results

To evaluate the performance of the learned control policy, as described in Section II, we tested the robot on a circular trajectory with a radius R and angular frequency ω. The starting point for the end-effector is randomly set within the workspace at the beginning of each episode. The robot is required to reach the initial point of the trajectory at an angle of 0 radians and proceed to track the full trajectory using the learned policy. The trajectory consists of two complete revolutions (i.e., 4π radians), after which the end-effector should maintain its position at the final point.

As shown in Figures 4 and 5 , the end-effector successfully reaches the desired trajectory from a random starting point and then tracks the circular trajectory with high precision. Figure 4 illustrates the desired trajectory in red, while the actual tracking trajectory is depicted in blue. Figure 5 further highlights the position tracking of each axis, showing the desired and actual positions with greater accuracy for both axes. The results demonstrate excellent performance in terms of precise tracking along the prescribed path and positioning accuracy. We also report the root mean square error (RMSE) and mean absolute error (MAE) for the entire trajectory in Table II. The mean error values are computed based on 100 evaluation runs within our simulation environment, providing a robust assessment of the policy's performance.

TABLE II
EVALUATION METRICS FOR TRAJECTORY TRACKING

Error Metric	Value(m)
RMSE	0.018
MAE	0.016
Variance	0.000086
Standard Deviation	0.008

The robot's tracking accuracy is evidenced by a position RMSE of 0.018m, demonstrating minimal deviation between the desired and actual positions of the end-effector. This low error value highlights the robot's ability to closely follow the planned trajectory. Additionally, the position MAE, averaging 0.016m, underscores the high level of accuracy maintained throughout the experiment.

To further assess consistency, the variance of the position error was calculated at 0.000086m, reflecting minimal fluctuations in position accuracy across different time steps. The standard deviation of the position error, measured at 0.008m, indicates a narrow spread around the mean error, signifying remarkable tracking performance across multiple runs. These metrics collectively demonstrate the stability of the model, with consistently low error values observed across

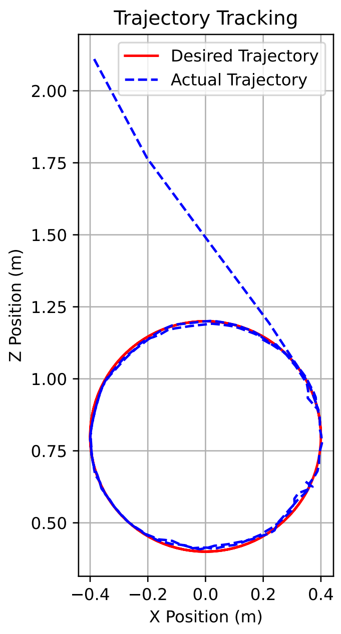

Fig. 4. Trajectory tracking

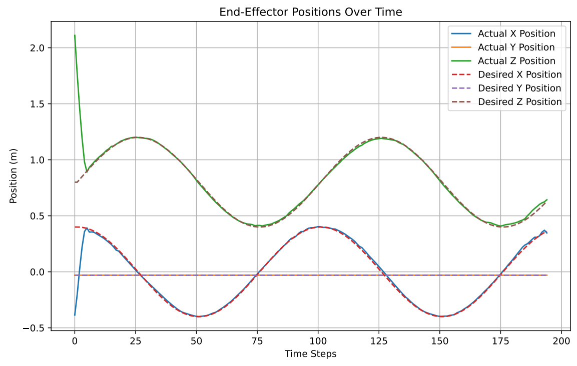

Fig. 5. End-effector trajectory tracking over time.

100 trials, affirming its robustness and reliability under various test conditions.

IV. DISCUSSION

One of the primary challenges in cable-driven parallel robots is maintaining consistent cable tension to ensure stability and precise control during operation. This issue is typically addressed through various methods, such as adjusting cable tension using motors based on feedback from tension sensors. In this work, the tension issue was handled through the simulation model, which operates independently of the control system. For future work, our goal is to integrate cable tension management alongside the trajectory tracking process within our RL-based controller. This approach will enhance the realism of the simulation, allowing the proposed controller to be tested under conditions that closely replicate real-world challenges.

In addition to integrating cable tension management, there are other areas that merit attention in future studies. For instance, safety in reinforcement learning (safety-RL) is a growing area of research that focuses on ensuring safe exploration and operation of RL-based systems, especially in real-world applications where failures could have severe consequences. This is especially relevant in the context of cable-driven robots, which operate in dynamic environments and must avoid hazardous conditions, such as excessive cable slack or tension overloads. Incorporating safety-RL could help improve the system's ability to handle unexpected situations while ensuring safe operation. Together, these considerations will help bridge the gap between simulation and real-world performance, ultimately leading to more effective and reliable control solutions for cable-driven parallel robots.

V. Conclusion

In this study, we demonstrated the significant trajectory tracking performance of Deep Reinforcement Learning (DRL) using the Deep Deterministic Policy Gradient (DDPG) algorithm, implemented on a planar Cable-Driven Parallel Robot (CDPR) in the MuJoCo simulator for the first time. The results show the potential of DRL for precise control of CDPRs, with strong tracking accuracy and minimal errors in trajectory tracking. This work highlights the effectiveness of DRL in handling the challenges of nonlinear systems like CDPRs, providing a stable and reliable control solution. Moreover, this study could pave the way for further improvements in implementing cable-driven robots in a well-known open-source simulator like MuJoCo, which is suited for developing and testing learning-based controllers.

References

1 P. Miermeister, M. Lachele, R. Boss, C. Masone, C. Schenk, J. Tesch, M. Kerger, H. Teufel, A. Pott, and H. H. Bulthoff, "The cablerobot simulator large scale motion platform based on cable robot technology," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3024–3029.

2 G. Chen, S. Baek, J.-D. Florez, W. Qian, S.-w. Leigh, S. Hutchinson, and F. Dellaert, "Gtgraffiti: Spray painting graffiti art from human painting motions with a cable driven parallel robot," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4065–4072.

3 S. Qian, B. Zi, W. W. Shang, and Q. S. Xu, "A review on cable-driven parallel robots," *Chinese Journal of Mechanical Engineering*, vol. 31, p. 66, 2018.

4 J. Merlet, "Jacobian, manipulability, condition number, and accuracy of parallel robots," in *ISRR*, 2005, pp. 199–206.

5 E. L. Oyman, M. Y. Korkut, C. Yılmaz, Z. Y. Bayraktaroglu, and M. S. Arslan, "Design and control of a cable-driven rehabilitation robot for upper and lower limbs," *Robotica*, vol. 40, no. 1, pp. 1–37, 2021.

6 A. Ghasemi, "Application of linear model predictive control and input-output linearization to constrained control of 3d cable robots," *Modern Mechanical Engineering*, vol. 1, pp. 69–76, 2011.

7 C. D. Li, J.-Q. Yi, Y. Yu, and D.-B. Zhao, "Inverse control of cable-driven parallel mechanism using type-2 fuzzy neural network," *Acta Automatica Sinica*, vol. 36, pp. 459–464, 2010.

8 H. Xiong, L. Zhang, and X. Diao, "A learning-based control framework for cable-driven parallel robots with unknown jacobians," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 234, pp. 1024–1036, 2020.

9 C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone, "Deep reinforcement learning for robotics: A survey of real-world successes," *arXiv preprint arXiv:2408.03539*, 2024.

10 A. Kumar and R. Sharma, "Linguistic lyapunov reinforcement learning control for robotic manipulators," *Neurocomputing*, vol. 272, pp. 84–95, 2018.

11 S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3389–3396, 2017.

12 D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv preprint arXiv:1806.10293*, 2018.

13 Z. Lu, T. Zhao, X. Jiang *et al.*, "Reinforcement learning for robotic manipulation in industrial assembly tasks," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 5, pp. 4532–4540, 2022.

14 S. Yuan, Y. Pan, X. Cheng *et al.*, "Autonomous navigation using reinforcement learning and deep learning: A review," *IEEE Access*, vol. 10, pp. 23 401–23 419, 2022.

15 M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

16 C. Sancak, F. Yamac, and M. Itik, "Position control of a planar cable-driven parallel robot using reinforcement learning," *Robotica*, vol. 40, no. 10, pp. 3378–3395, 2022.

17 C. Xie, J. Zhou, R. Song, and T. Xu, "Deep reinforcement learning-based cable tension distribution optimization for cable-driven rehabilitation robot," in *2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE, 2021, pp. 318–323.

18 P. Nomanfar and L. Notash, "Motion control of a cable-driven parallel robot using reinforcement learning deep deterministic policy gradient multi-agents," in *IEEE Conference on Robotics and Automation*. IEEE, 2024.

19 Z. Xu, S. Wei, N. Wang, and X. Zhang, "Trajectory planning with bezier curve in cartesian space for industrial gluing robot," in *Intelligent Robotics and Applications. ICIRA 2014. Lecture Notes in Computer Science*, X. Zhang, H. Liu, Z. Chen, and N. Wang, Eds. Springer, Cham, 2014, vol. 8918.

20 A. Raman, A. Salvi, M. J. Schmid, and V. Krovi, "Reinforcement learning control of a reconfigurable planar cable driven parallel manipulator," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9644–9650.

21 M. Bajelani, S. A. Khalilpour, M. I. Hosseini, H. Taghirad, and P. Cardou, "Model-free joint space controller for fully-constrained cable-driven parallel robots: A bio-inspired algorithm," *International Journal of Robotics, Theory and Applications*, vol. 9, no. 1, pp. 11–19, 2023.

22 J. Yoon, S. W. Hwang, J.-H. Bak, and J. H. Park, "Adaptive control for cable-driven parallel robots," in *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, 2017, pp. 416–419.

23 T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

24 E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

25 A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 1, pp. 1–8, January 2021.