Robotica (2022), **40**, pp. 3378–3395
doi:10.1017/S0263574722000273

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Position control of a planar cable-driven parallel robot using reinforcement learning

Caner Sancak[1,*], Fatma Yamac[2] and Mehmet Itik[3]

[1]Department of Mechanical Engineering, Karadeniz Technical University, Trabzon, Turkey, [2]Department of Mechanical Engineering, Tarsus University, Mersin, Turkey, and [3]Department of Mechanical Engineering, Izmir Democracy University, Izmir, Turkey
*Corresponding author. E-mail: csancak@ktu.edu.tr

Received: 11 October 2021; **Revised:** 18 December 2021; **Accepted:** 6 February 2022; **First published online:** 17 March 2022

Keywords: cable-driven parallel robot, reinforcement learning, position control, torque distribution, MIMO system

Abstract

This study proposes a method based on reinforcement learning (RL) for point-to-point and dynamic reference position tracking control of a planar cable-driven parallel robots, which is a multi-input multi-output system (MIMO). The method eliminates the use of a tension distribution algorithm in controlling the system's dynamics and inherently optimizes the cable tensions based on the reward function during the learning process. The deep deterministic policy gradient algorithm is utilized for training the RL agents in point-to-point and dynamic reference tracking tasks. The performances of the two agents are tested on their specifically trained tasks. Moreover, we also implement the agent trained for point-to-point tasks on the dynamic reference tracking and vice versa. The performances of the RL agents are compared with a classical PD controller. The results show that RL can perform quite well without the requirement of designing different controllers for each task if the system's dynamics is learned well.

1. Introduction

Cable-driven parallel robots (CDPRs) are parallel mechanisms, which use cables instead of rigid actuation limbs. In a cable robot configuration, the end-effector's motion is provided by cables actuated by electric motors and pulley mechanisms. CDPRs have simple and light mechanical structures, high-speed moving ability, high acceleration rate, and large working spaces due to the advantageous properties of the cables. Therefore, CDPRs can be used in various industrial areas such as material handling and placement, carrier crane systems, rehabilitation devices, and assembly processes [1].

CDPRs are investigated on diverse topics such as kinematic and dynamic modeling [2–5], stability [6], vibration analysis [7], and out-of-plane vibration control [8]. In controlling the pose of CDPRs, various methods have been presented in the literature. In refs. [9, 10], an experimental control of a planar CDPR is performed with a robust PID controller, and impedance and PD controller are implemented to control cables on the rehabilitation robot. The sliding mode control method, which can handle the changes in system parameters, disturbing effects, and uncertainties, is employed in refs. [11–13]. More recently, synchronization control in the cable space [14] and coordinated dynamic control in the task space [15] to increase the position tracking accuracy of CDPRs have been proposed. Controller design for CDPRs using classical methods such as PID and model-based state-space methods is challenging, having limitations, trade-offs, and dynamic complexity due to several reasons. First, adjusting the controller parameters while maintaining the control input constraints is difficult by trial-error methods and generally requires off-line optimization for the desired control behavior. However, the controller parameters are tuned based on the reference to be tracked, and when the reference input characteristics change, a new control parameter set must be obtained. Second, model-based classical control methods utilize a tension distribution algorithm to ensure that the cable tensions are always positive and provide

© The Author(s), 2022. Published by Cambridge University Press.

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

a unique tension distribution solution. There should be no sagging in the cables during the operation of the CDPR since the cables can only carry pulling force. Therefore, optimization methods are used to determine the appropriate tension distributions to ensure the positive tension of the cables [16–20]. However, finding optimal tension distributions cannot be done on-line and must be performed off-line based on a designed controller. Moreover, when the actuation redundancy increases, the on-line solution of tension distribution algorithms becomes more tedious and time-consuming causing delays in the control process which may lead to instability.

Learning-based control methods have promising features to control the complex dynamics and handle uncertainties of the CDPRs [21, 22]. Reinforcement learning (RL) algorithms have recently been used to control robots in various application areas where complex operations such as path planning, obstacle avoidance, and complex assembling tasks are required [23, 24]. RL is a machine learning approach that deals with choosing actions to maximize a numerical reward signal. The agent, who carries out the learning process, performs its actions according to a policy and receives a reward from the environment. In this way, the agent is in continual interaction with the environment, and the main goal is to optimize the amount of reward it collects from the environment [25]. In controlling the CDPR by the RL method, there is no need for a tension distribution algorithm since the required control input can be generated by learning. Adjusting the rewards given for the actions provides the desired control behavior. RL method does a trial-and-error process by itself to learn the required control input and can handle complex tasks by learning. After the learning process, the RL method has formed an artificial neural network controller in which the computational cost is relatively low. RL method can overcome the problems of classical model-based controllers in controlling the CDPRs. The advantages of the RL over classical control methods are as follows. First, since different reference signal characteristics can be taught to the RL agents in the learning process, there is no need to tune controller parameters for different reference signals using off-line optimization methods. Moreover, control input constraints can be handled easily in the learning process of RL agents. Second, the RL method finds the required torques directly during the training of the RL agents and does not require any tension distribution algorithm. Therefore, it saves from the off-line tension distribution optimization problem and provides the tension distributions in real-time. These advantageous features of the RL method make it convenient to control the CDPR dynamics. There are very few works on controlling CDPRs using RL [26–28]. However, the RL control of planar CDPRs with two translational and one rotational degree of freedom has not been investigated yet.

This paper investigates the performance of RL to perform given point-to-point and dynamic reference tracking tasks for a planar CDPR which is driven with four cables and has three degrees of freedom for the first time, up to the knowledge of the authors. In the point-to-point task, reaching the reference point and, unlike [26], the response after reaching, and steady-state behavior of the control process is considered. For this purpose, training of the RL agent is not terminated immediately after reaching the reference point as generally did for the RL tasks. On the contrary, we continue the training for a certain period in one episode to cover the dynamics of the robot after reaching nearby the desired point. For the dynamic reference tracking task, another RL agent with the same design parameters as the point-to-point agent is trained to track sinusoidal reference signals. Thus, it is intended for the agent to learn dynamic reference tracking, where the change of the pose error does not only depend on the action, as is the case in the point-to-point control task. This paper also presents a novel reward construction to achieve desired control behavior which has three phases; reaching the desired pose, keeping the end-effector at the pose, and minimizing the cable tensions. Besides the control, the deep deterministic policy gradient (DDPG) algorithm is programed to find a minimum control action solution by adjusting the RL rewards. This provides to find the optimal tension distribution that ensures the positive tension in the cables. Both trained agents have successfully performed the given training tasks. The training signal used for one agent is used as a reference signal for the other agent to validate and compare the agents' performances. The agent trained for point-to-point control showed similar performance as dynamic tracking control, which is performed with the agent trained for the dynamic reference signal and vice-versa. In order to compare the performance of the RL agent with a classical controller, a PD controller is also designed and employed in the simulations.

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

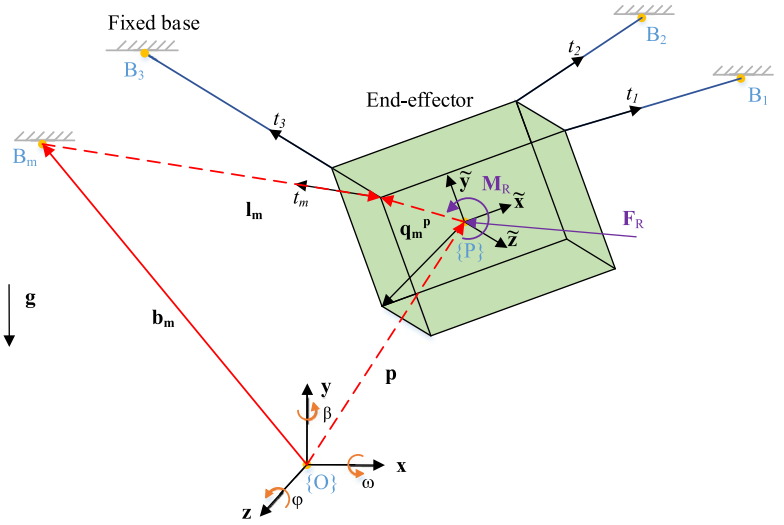


Figure 1. *Free-body diagram of a CDPR.*

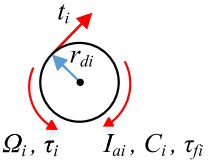


Figure 2. *Free-body diagram of the actuator.*

The rest of the paper is organized as follows. First, the CDPR discussed in this study is introduced, and the dynamical equations of the system are given. In Section 3, the RL method is explained, and the design parameters of the RL are provided. Then, simulation results are presented with detailed discussions. Finally, the conclusions are drawn.

2. Cable-driven parallel robot

A CDPR is a kind of manipulator that is driven by a set of cables in parallel. CDPRs consist of four essential components: a fixed base platform, actuators, end-effector, and cables. The movement of the end-effector is provided by lengthening and shortening the cables with the rotation of the actuator. The dynamic structure of the CDPR comprises the mobile end-effector and the actuator dynamics. A kinematic diagram of a typical CDPR structure with cables and the free-body diagram of its dynamic components are shown in Figs. 1 and 2.

A closed-loop kinematic chain shown in Fig. 1 can be used for each of the cables as given in Eq. (1) to find the required cable lengths in a given position of the end-effector.

$$\boldsymbol{l}_i = \boldsymbol{p} + \boldsymbol{R}\boldsymbol{q}_i^P - \boldsymbol{b}_i, \quad \mathrm{i} = 1, 2 \ldots, \mathrm{m} \tag{1}$$

where $\boldsymbol{l}$ is the cable length vector, $\boldsymbol{p}$ is the position vector, $\boldsymbol{R}$ is the rotation matrix, $\boldsymbol{q}^P$ is a constant position vector in end-effector frame {P}, and $\boldsymbol{b}$ is a constant position vector in the base frame {O}.

In dynamic modeling, it is assumed that the dynamic model of the CDPR consists of the actuators (winches) and the end-effector dynamics. The dynamics of cables are ignored in order to simplify the dynamic model of the CDPR. Under these assumptions, the general dynamic model of a CDPR is

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

expressed [3] as

$$\boldsymbol{HT} = \boldsymbol{W} \tag{2}$$

where $\boldsymbol{H} = \begin{bmatrix} -\boldsymbol{l}_1/\,||\boldsymbol{l}_1|| & \dots & -\boldsymbol{l}_m/\,||\boldsymbol{l}_m|| \\ \boldsymbol{Rq}_1^P x\,(-\boldsymbol{l}_1/\,||\boldsymbol{l}_1||) & \dots & \boldsymbol{Rq}_m^P x\,(-\boldsymbol{l}_m/\,||\boldsymbol{l}_m||) \end{bmatrix}$ is the structure matrix and can be obtained as $\boldsymbol{H} = -\boldsymbol{J}^T$. Here, $\boldsymbol{J}$ is the jacobian matrix and can be determined by taking the time derivative of Eq. (1) and $\boldsymbol{T}$ is the cable tension vector. According to the Newton–Euler formulation, the dynamic wrench $\boldsymbol{W}$ acting on the end-effector can be represented as

$$\boldsymbol{W} = \begin{Bmatrix} m_e\ddot{\boldsymbol{p}} - m_e\boldsymbol{g} - \boldsymbol{F}_R \\ \boldsymbol{I}_e\ddot{\boldsymbol{\psi}} + \dot{\boldsymbol{\psi}} \times \boldsymbol{I}_e\dot{\boldsymbol{\psi}} - \boldsymbol{M}_R \end{Bmatrix} \tag{3}$$

where m_e is the mass of the end effector, $\boldsymbol{I}_e$ is the inertia tensor of the end-effector with respect to body axis, $\boldsymbol{g}$ is the acceleration of gravity, $\boldsymbol{\psi}$ is rotational position vector, and $\boldsymbol{F}_R$ and $\boldsymbol{M}_R$ are external force and moment vector, respectively. The dynamic model of the actuators that is considered with a motor shaft and cable winding drum is given as

$$\boldsymbol{T} = \boldsymbol{r}_d^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\tau}_f - \boldsymbol{I}_a\ddot{\boldsymbol{\Omega}} - \boldsymbol{C}\dot{\boldsymbol{\Omega}}\right) \tag{4}$$

where $\boldsymbol{\tau}$, $\boldsymbol{\tau}_f$, $\boldsymbol{\Omega}$, are the input torque, friction torque, and rotational position vectors of the actuators, respectively. $\boldsymbol{I}_a$, $\boldsymbol{C}$, and $\boldsymbol{r}_d$ is rotational inertia, viscous damping coefficient, and drum radius matrices which are constructed diagonally. By combining Eqs. (2) and (4), the dynamic model of the CDPR is given in following form:

$$\boldsymbol{M}_{\text{eq}}\ddot{\boldsymbol{X}} + \boldsymbol{N}\left(\boldsymbol{X}, \dot{\boldsymbol{X}}\right) = \boldsymbol{H}\boldsymbol{\tau} \tag{5}$$

$$\boldsymbol{M}_{\text{eq}} = \boldsymbol{H}\boldsymbol{I}_a\left(\frac{\partial\boldsymbol{\Omega}}{\partial\boldsymbol{X}}\right) + \boldsymbol{r}_d\begin{bmatrix} m_e\boldsymbol{I}_{3\times3} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_e \end{bmatrix} \tag{6}$$

$$\boldsymbol{N}\left(\boldsymbol{X}, \dot{\boldsymbol{X}}\right) = \boldsymbol{H}\left(\boldsymbol{I}_a\frac{d}{\text{dt}}\left(\frac{\partial\boldsymbol{\Omega}}{\partial\boldsymbol{X}}\right) + \boldsymbol{C}\left(\frac{\partial\boldsymbol{\Omega}}{\partial\boldsymbol{X}}\right)\right)\dot{\boldsymbol{X}} + \boldsymbol{H}\boldsymbol{\tau}_f + \boldsymbol{r}_d\begin{bmatrix} -m_e\boldsymbol{g} - \boldsymbol{F}_R \\ \dot{\boldsymbol{\psi}} \times \boldsymbol{I}_e\dot{\boldsymbol{\psi}} - \boldsymbol{M}_R \end{bmatrix} \tag{7}$$

where $\boldsymbol{M}_{\text{eq}}$ is the equivalent inertia matrix, $\boldsymbol{N}\left(\boldsymbol{X}, \dot{\boldsymbol{X}}\right)$is the nonlinear terms, $\mathbf{X} = \left[\boldsymbol{p}, \boldsymbol{\psi}\right]^T$ is the end-effector position vector, and $\boldsymbol{I}$ is the identity matrix.

3. Reinforcement learning

RL is a machine learning approach that learns what a decision-maker (agent) must do to maximize a numerical reward signal and how to map states to actions. The environment in which the agent is in RL has the Markov decision process (MDP) feature. In MDP, the agent's current state is assumed to include all past information that will affect future states of the process, and future states in the current state are independent of past states. In the MDP environment (Fig. 3), the agent interacts with the environment by choosing an action in its current state and later transfers a new state by receiving a reward signal in response to its action. In this way, the agent is in constant interaction with the environment.

The components of MDP are $\{S, A, P, R, \gamma\}$. At any time interval t, the state of the agent in the environment is S_t, the action chosen for this state is A_t, the reward from this action is R_{t+1}, and the new state is S_{t+1}. The transition probability function p that defines the system dynamics is Eq. (8), and for each given state s and action a, $p(s', r\,|s, a)$ brings the agent from state s to the next state s'. The discount factor $\gamma \in [0, 1]$ is a parameter that essentially determines how much the agent cares about rewards in the distant future relative to those in the immediate future. For larger values of γ, the agent cares about distant future rewards, and for smaller values, the agent mainly concerns rewards in near future [25].

$$p(s', r\,|s, a) = Pr\left\{S_t = s', R_t = r, |S_{t-1} = s, A_{t-1} = a\right\}, (s', s)\epsilon S, r\epsilon R, a\epsilon A(s) \tag{8}$$

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

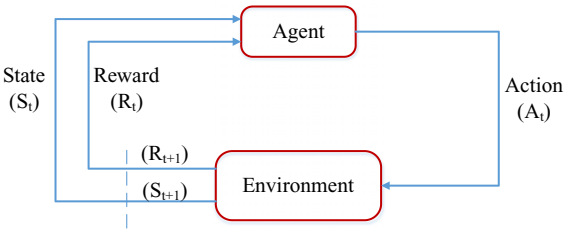


Figure 3. *Agent−environment interaction in the RL system.*

The main goal of the RL agent is to maximize the total amount of reward received in the long term, which is defined as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{9}$$

Therefore, the agent must choose actions that will maximize the total amount of rewards. The choice of action from the agent's action space is based on a policy that can be stochastic or deterministic. There are many policies that the agent can follow from its current state, but only the optimal policy π^* makes the total return the highest [25]. In order to find the optimal policy, value-based RL algorithms such as Q-learning and DQN (deep Q-network) determine the optimal value function and the corresponding policy [29, 30]. On the other hand, policy-based methods such as DDPG, A3C (Asynchronous Advantage Actor−Critic), TD3 (Twin Delayed DDPG) directly determine the optimal policy [31–34]. In the Q-learning method, a table is used in which the $Q(s, a)$ values for each state and action pair are kept. The Q values in the table are updated by Eq. (10) after each interaction of the agent with the environment with a learning rate α:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(r + \gamma \max Q(s', a') - Q(s, a)\right) \tag{10}$$

The Q-learning method is insufficient for problems where the state and action space are extensive due to a lack of memory and computational cost. The DQN method developed using ANNs instead of tables can also solve high-dimensional problems where the state space is continuous. An ANN called Q-Network which have θ^Q parameters that predict the $Q(s, a)$ values of actions against a given state. Q-Network training is carried out updating the error function given as follows:

$$L(s, a) = \left(y - Q\left(s, a \left|\theta^Q\right.\right)\right)^2 \tag{11}$$

In this method, the target value y is calculated by

$$y = r + \gamma \max Q'\left(s', a' \left|\theta^{Q'}\right.\right) \tag{12}$$

An ANN with $\theta^{Q'}$ parameters called Target Q-Network is used to calculate the target value, and it is updated periodically according to the Q-Network. The θ^Q parameters of the Q-network are updated by

$$\theta^Q = \theta^Q + \alpha \nabla_{\theta^Q} L\left(s, a \left|\theta^Q\right.\right) \tag{13}$$

While DQN can solve problems where the observation space is high dimensional, it can only handle discrete and low dimensional action space. However, many physical processes have large and continuous action space. Methods such as DDPG, A3C, and TD3, including Actor−Critic algorithms shown in Fig. 4, are used in the literature to solve high-dimensional problems with such continuous action structure.

In the DDPG method, there are four artificial neural networks: Critic-Network $Q\left(s, a \left|\theta^Q\right.\right)$, Actor-Network $\mu(s \left|\theta^\mu\right.)$, Target Critic-Network $Q'\left(s, a \left|\theta^{Q'}\right.\right)$, and Target Actor-Network $\mu'\left(s \left|\theta^{\mu'}\right.\right)$. The actor observes the current state s and tries to determine the action a that maximizes the long-term reward. On the other hand, the critic takes the observation s and action a as input and estimates the expectation

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

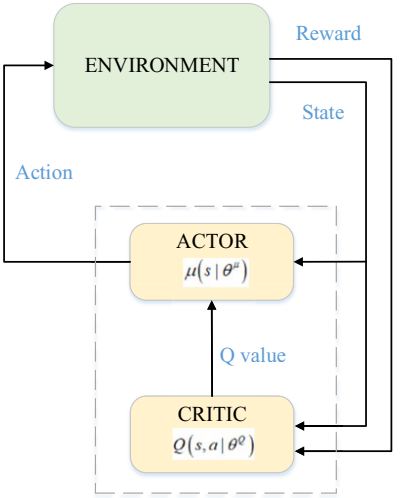


Figure 4. *Actor−Critic method.*

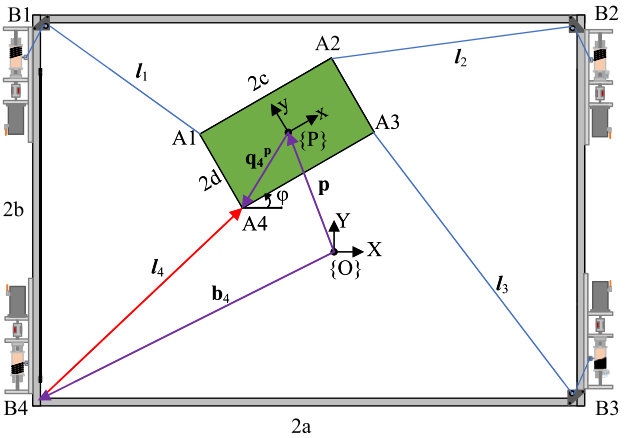


Figure 5. *Schematic of the planar CDPR.*

value corresponding to the long-term reward. Target Actor and Target Critic determine the target value required to calculate the error function used in updating Critic and Actor networks. The target value y is obtained as follows:

$$y = \left(r + \gamma Q'\left(s', \mu'\left(s' \left|\theta^{\mu'}\right.\right) \left|\theta^{Q'}\right.\right)\right) \tag{14}$$

In calculating the target value y, the reward from the environment and Target network are used. The error function for the Critic-Network, and the gradient of the Actor-Network are given, respectively, as

$$L\left(s, a \left|\theta^{Q}\right.\right) = \left(y - Q\left(s, a \left|\theta^{Q}\right.\right)\right)^2 \tag{15}$$

$$\nabla_{\theta^{\mu}} J = \nabla_a Q\left(s, a|\theta^{Q}\right)_{|a=\mu(s|\theta^{\mu})} \nabla_{\theta^{\mu}} \mu\left(s|\theta_k^{\mu}\right) \tag{16}$$

θ^{Q} and θ^{μ} network parameters of Critic and Actor-Network are updated as

$$\theta^{Q} = \theta^{Q} + \alpha \nabla_{\theta Q} L\left(\theta^{Q}\right) \tag{17}$$

$$\theta^{\mu} = \theta^{\mu} + \alpha \nabla_{\theta^{\mu}} J(\theta^{\mu}) \tag{18}$$

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

Table I. *Kinematic parameters of the planar CDPR.*

a (m)	0.683
b (m)	0.670
c (m)	0.143
d (m)	0.098

Table II. *Dynamic parameters of the planar CDPR.*

τ_f (Nm)	diag(*0.045, 0.045, 0.045, 0.045*)
I_a (kgm^2)	diag(*1.78, 1.78, 1.78, 1.78*)$^*10^{-4}$
C (Nms)	diag(*7.06, 7.06, 7.06, 7.06*)$^*10^{-5}$
r_d (m)	diag(*0.025, 0.025, 0.025, 0.025*)
m_e (kg)	0.517
I_e (kgm^2)	17.8×10^{-4}

$\theta^{Q'}$, $\theta^{\mu'}$ and λ are the network parameters and smooth factor of Target Critic and Target Actor-Networks and updated as

$$\theta^{Q'} = \lambda\theta^{Q} + (1-\lambda)\,\theta^{Q'} \tag{19}$$

$$\theta^{\mu'} = \lambda\theta^{\mu} + (1-\lambda)\,\theta^{\mu'} \tag{20}$$

4. Control of the CDPR using DDPG agent

4.1. Specifications of the CDPR

The CDPR considered in this study is assumed to be located in a plane. The end-effector has 3 degrees of freedom (rotation around the z-axis (φ) and translation in x and y axes), and it is driven by four cables (Fig. 5). The kinematic and dynamic parameters of the planar CDPR are given in Tables I and II. It is assumed that no external force and moment is applied to the end-effector. The mathematical model of the CDPR, whose dynamic equations are presented previously, and the design of the necessary elements for DDPG is created with the MATLAB/Simulink 2020.

4.2. Observations and reward function

In RL, the observed states in the environment vary according to the type of problem. Selection of the state observation is essential for the convergence of the RL algorithm. Here, since the CDPR must track a given reference, the state matrix (observations) is chosen as

$$\boldsymbol{S}_t = \begin{bmatrix} x & y & \varphi & \dot{x} & \dot{y} & \dot{\varphi} & e_x & e_y & e_\varphi \end{bmatrix} \tag{21}$$

The state vector includes the pose vector, its derivative, and tracking error. The pose vector is included in observation because the agent must learn to apply different actions that change nonlinearly with the pose. The derivative of the pose vector is included to learn the dynamic interaction between inputs and pose change. Therefore, the RL agent can learn the direction of the movement and smooth actions with the help of the reward. Lastly, the tracking errors are included to learn that minimizing these states brings more rewards. The RL agent tries to determine the action values that will maximize the reward function according to the observations from the environment. These action values represent the input torques (cable tensions) for the CDPR and are given in a vector form as

$$\boldsymbol{a}_t = \begin{bmatrix} \tau_1 & \tau_2 & \tau_3 & \tau_4 \end{bmatrix} \tag{22}$$

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

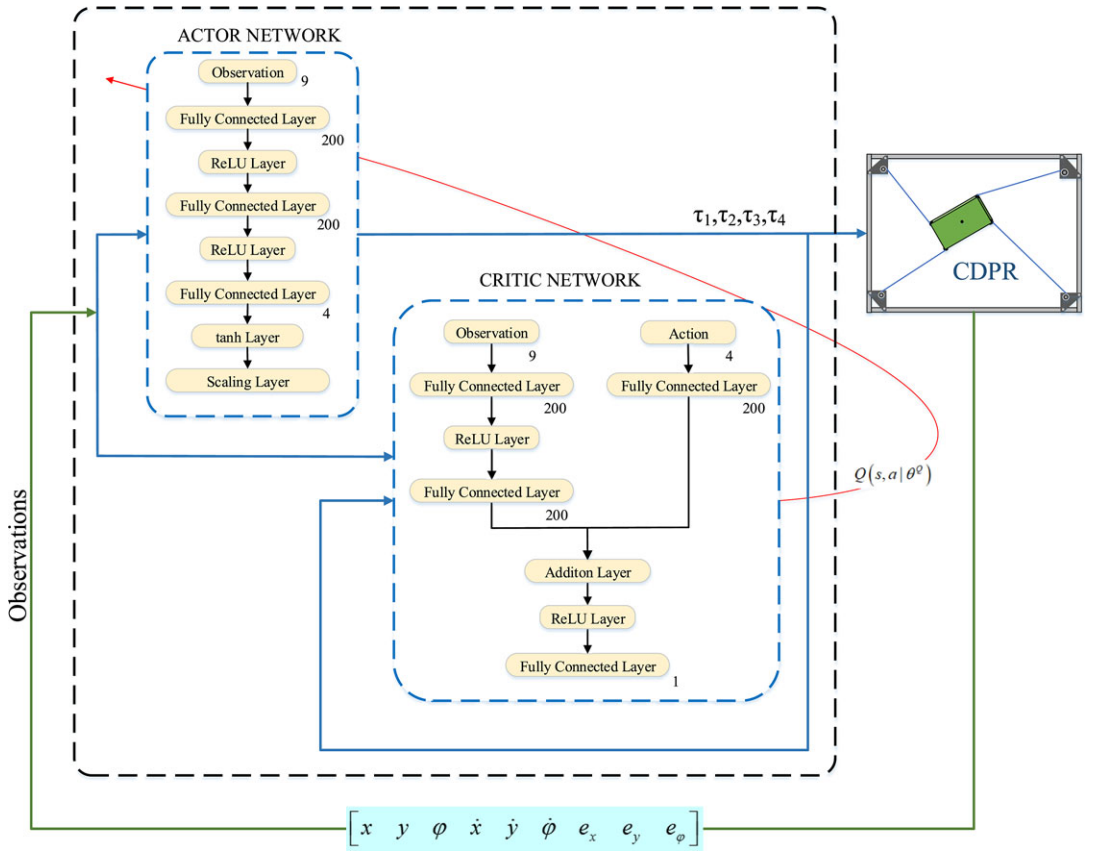


Figure 6. *Actor and Critic network structure.*

The reward function determined for the control of the CDPR consists of three separate parts. The first reward R_1 given by Eq. (23) punishes the agent exponentially according to the tracking errors. If the tracking error is too high, the calculated penalty becomes high, and the penalty is reduced depending on the decrease in the error. The second reward R_2, given in Eq. (24), punishes the input action. Therefore, the RL agent will avoid producing high action values and reach the minimum optimal actuation values to drive the CDPR in a given task. The reward R_3, given by Eq. (25) uses derivatives of absolute errors and works depending on the sign change of the error. If the error increases, it is penalized, and if the error decreases, it is rewarded. The penalty given for when the agent diverges from the target depends on the divergence rate, but when the agent converges to the target, the received reward has limits. These limits are used to prevent generating highly aggressive actions in order to collect more reward. Diverging from the desired pose penalty is unlimited, thus informing the agent that it is undesired behavior in error change rate.

$$R_1 = \exp\left(-3\left(||e_x|| + ||e_y|| + ||e_\varphi||\right)\right) \tag{23}$$

$$R_2 = -0.05 \sum \tau_i^2 \tag{24}$$

$$R_3 = \left\{ \begin{array}{r} ;||\dot{e_x}|| = -10 \text{ if } ||\dot{e_x}|| \leq -10 \\ -\left(||\dot{e_x}|| + ||\dot{e_y}|| + ||\dot{e_\varphi}||\right)_{t-1} \quad ;||\dot{e_y}|| = -10 \text{ if } ||\dot{e_y}|| \leq -10 \\ ;||\dot{e_\varphi}|| = -10 \text{ if } ||\dot{e_\varphi}|| \leq -10 \end{array} \right\} \tag{25}$$

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

Table III. *Hyper-parameters.*

Parameters	**Values**
Learning rate α (Actor network)	0.001
Learning rate α (Critic network)	0.0001
Discount factor γ	0.99
Experience replay	10^6
Mini-batch	1024
Sample time	0.01
Target smooth factor λ	0.001
Target update frequency	100
Noise variance	0.4
Noise variance decay rate	0.01
Maximum number of time steps for each episode	5000

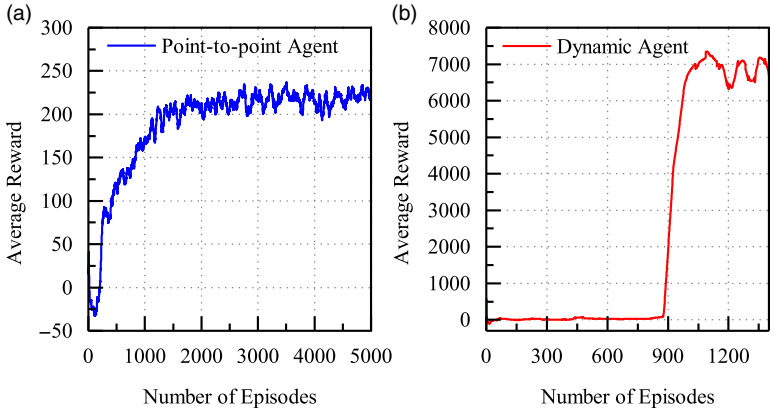


Figure 7. *Average rewards of the DDPG agents in the point-to-point control training (a) and the dynamic reference tracking training (b).*

Consequently, the sum of these three rewards forms the final reward function. The coefficients of the rewards are chosen by considering their effects on the learning process. At the beginning of the learning, it is desired that R_3 has more effect on the total reward to provide that all states start to converge to the target. Then, R_1 dominates the total reward so that the agent keeps the robot's position close to the target. Finally, the R_2 reward affects the learning process after the R_3 and R_1 completed their duty. Thus, the control action converges to minimum control action to perform the task.

4.3. DDPG network structure and hyper parameters

The number of neurons and activation functions in the layers of the Actor and Critic network structures are shown in Fig. 6. Critic estimates the Q value based on the observations from the environment and the action it receives from the Actor. The Actor estimates the torque values, which are the input of the CDPR system, and sends them to the actuators to move the cables on the CDPR system. As a result, the pose of the end-effector changes with the movement of the cables, and error between the pose and reference signal is calculated to obtain the observation matrix. To design the network and chose hyper-parameters, there are no known rules. It depends on the problem's nature. Therefore, these parameters are determined by trial and error. Simulations are performed until an acceptable result to control the CDPR is

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

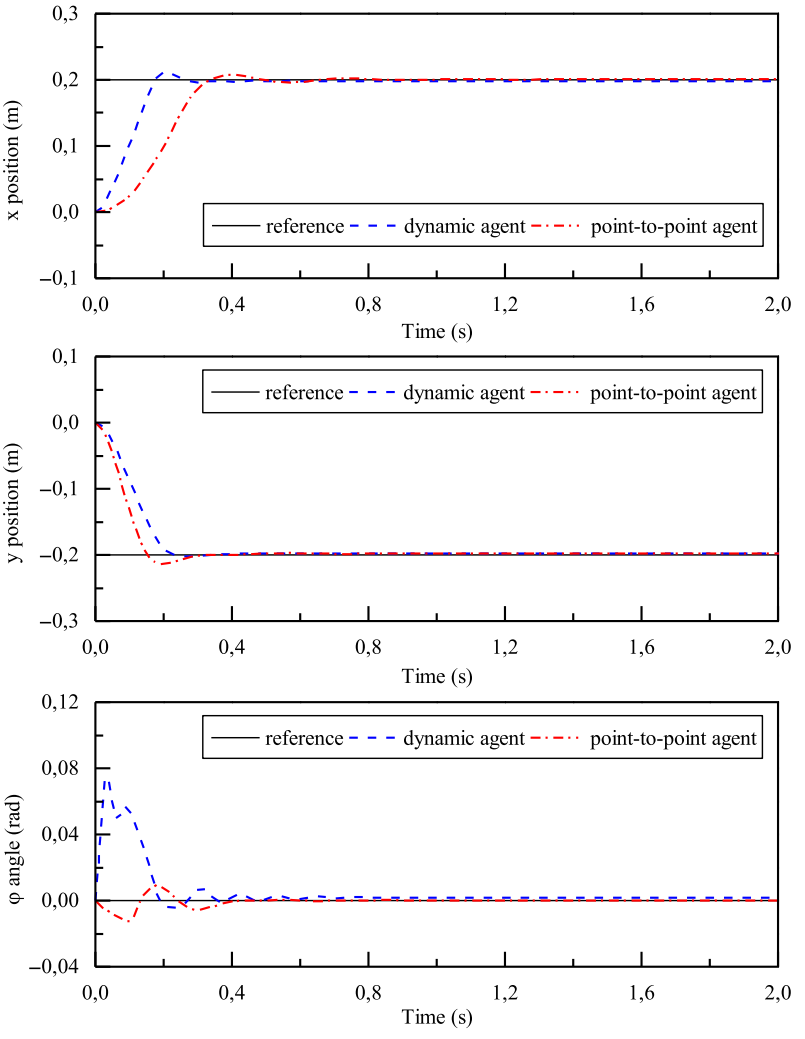


Figure 8. *Control response of the dynamic and point-to-point agent for the point-to-point (step) task.*

obtained. The RL agents used in this paper are trained using the same network structure and hyper-parameters given in Table III. The minimum (τ_{min}) and maximum (τ_{max}) actuator torques are bounded to 0.06 and 1.4 Nm by the scaling layer, respectively. Thus, generating large, and negative cable tensions are prevented.

5. Simulation results

5.1. Reinforcement learning results

We used two tasks for the CDPR control: the point-to-point control task and the dynamic reference tracking task. The CDPR should move from a starting point to a target point in the workspace with an acceptable transient response and stay at the target point with a tolerable steady-state error. Hence, the first RL agent was trained to perform a point-to-point control task, and it was named as point-to-point agent. For this purpose, initial positions and target positions were randomly determined in the training process for every episode. Initial positions and target positions were changed between $(-0.5, 0.5)$ m for the x, y, and $(-0.1, 0.1)$ rad for the φ state, respectively. We aimed that the agent learned to control the

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

Table IV. *Step response characteristics for the dynamic and point-to-point agent.*

Step information	**Dynamic agent**		**Point-to-point agent**	
	x	***y***	***x***	***y***
Rise time (s)	0.1234	0.1426	0.1994	0.1026
Settling time (s)	0.3040	0.2115	0.5803	0.2737
Overshoot (%)	5.2679	1.2046	3.7190	6.8724

Table V. *RMSE in the control tasks.*

Control type		**RMSE (m)**		
Reference signal	**Agent type**	***x* position**	***y* position**	**φ angle**
Dynamic (sinusoidal)	Dynamic	0.0053	0.0036	0.0056
	Point-to-point	0.0102	0.0034	0.0020
Point (Step)	Dynamic	0.0389	0.0415	0.0144
	Point-to-point	0.0558	0.0360	0.0028

CDPR for different conditions. One episode simulation time was set to 2 s, which was quite adequate to conclude the point-to-point task in the given control range of the CDPR. The termination criteria, which ended the episode when the end-effector was out of the workspace, was also included in the simulations. The limits were selected as (0.5, −0.5) m for x and y axes, and (0.3, −0.3) rad for φ. Thus, improper learning in the first episodes was prevented by terminating the episode when the end-effector exceeds the workspace limits. Figure 7(a) shows the average rewards of the point-to-point agent per episode. As shown in the figure, the collected reward per episode tends to increase at an exponential rate. After the 1700th episode, the collected reward becomes relatively stable for the following episodes. It means that the agents between 1700 and 5000 episodes can be chosen for the point-to-point control purpose. According to the reward values, there are minor differences among the agents. The agent at the end of the 1757th episode was chosen as the best agent in this work. Although there were agents that gained more reward than the selected agent, they showed aggressive changes in the control torques (cable tensions), which is an undesired way to gain reward. Another reason is, choosing an agent closer to the point that the collected reward becomes stable.

The second RL agent was trained to track dynamic reference signals, and it was named as dynamic agent. The same observations and reward function were used in both point-to-point and the dynamic agent learning processes. Sinusoidal references were used as the test signals for the x, y, and φ positions in the dynamic learning case whose amplitudes and frequencies were 0.3–0.5, 0.1–0.3, and 0.03–0.1 the x, y, and φ, respectively. Initial positions were determined randomly between (−0.5, 0.5) m for x, y, and (−0.1, 0.1) rad for φ in every episode, respectively. For the reference signal generation, the sine function was used for the x state while cosine was used for y and φ states to provide phase difference, which created a homogenous reference structure in terms of positive and negative directions in the learning process. With different initial conditions, it was aimed that the agent learned to reach and track the reference from different positions. One episode simulation time was set to 80 s for the agent to learn possible combinations of reference signals. The termination criteria were set for out of workspace positions. The behavior of the learning process for average rewards is given in Fig. 7(b). The collected rewards for the first 900 episodes are too low since the episodes have been terminated due to workspace restriction criteria. After the 900th episode, the agent keeps the end-effector in the workspace, and therefore, the collected reward increases. In the 1150th episode, the collected reward takes the highest value, and the following episodes collect similar rewards. The learning process was stopped at the 1400th episode due to the observed stable behavior of the collected rewards and to save time. The agent at the end of the 1159th episode was chosen as the best agent for the previously mentioned reasons.

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

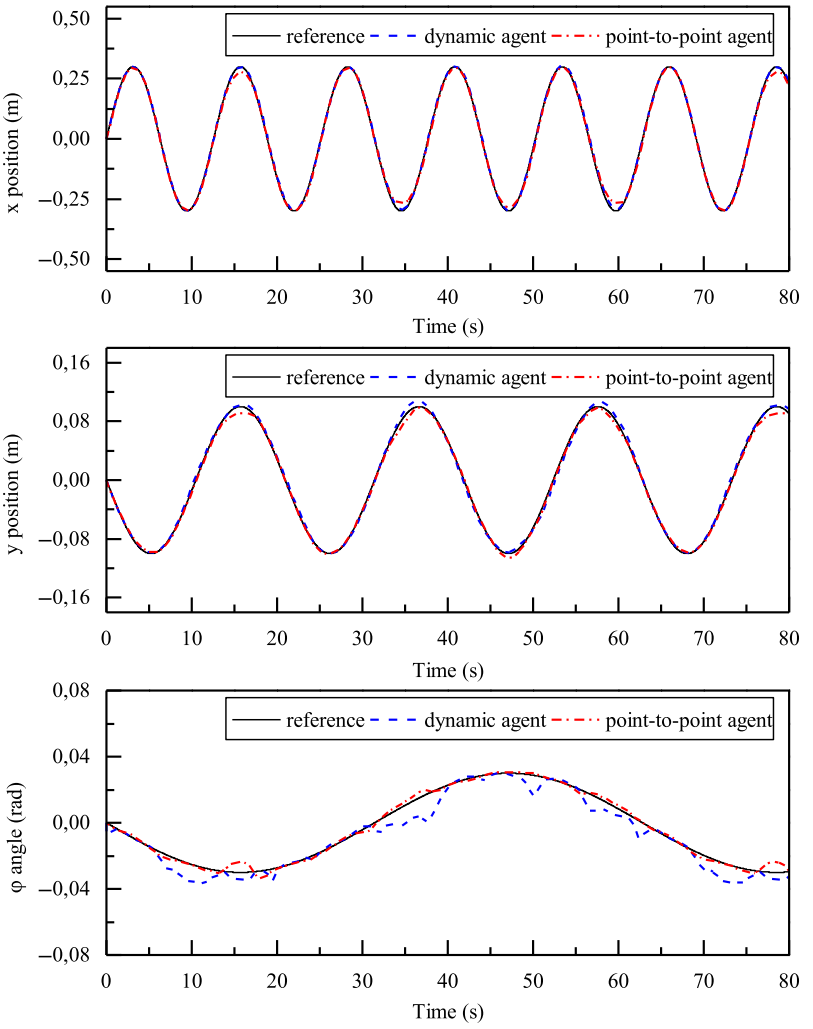


Figure 9. *Control response of the dynamic and point-to-point agent for the dynamic reference tracking task.*

Although the point-to-point agent started to learn faster than the dynamic agent, its maximum reward average is lower than the dynamic agent, mainly due to the simulation time. As the reward R_3 changes depending on the error difference, the dynamic agent tracking the sinusoidal input gains more rewards than the point-to-point agent. Learning to keep the end-effector in the workspace for the dynamic agent took quite longer than for the point-to-point agent. This is mainly due to the use of dynamic reference signals. It requires much experience to learn what to do in changing the last three observation states.

We also test the point-to-point agent on the dynamic (sinusoidal) reference tracking and the dynamic agent on the point-to-point (step) reference to see their performance on different tasks. Simulation results are comparatively given for the point-to-point agent and dynamic agent. First, in Fig. 8, the control results for reaching a point ($x = 0.2, y = -0.2, \varphi = 0$) reference are illustrated using the point-to-point and dynamic agent. The step response characteristics of the control processes are given in Table IV. Regarding x and y positions, the rise time is quite fast, but a tradeoff between x and y positions is observed in the control response. The dynamic agent is better in the rise time and overshoot for the x state and in settling time for the y state. This shows that being better in one state control costs other states control performance deterioration for both agents. A tradeoff is also observed in steady-state errors (Table V).

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

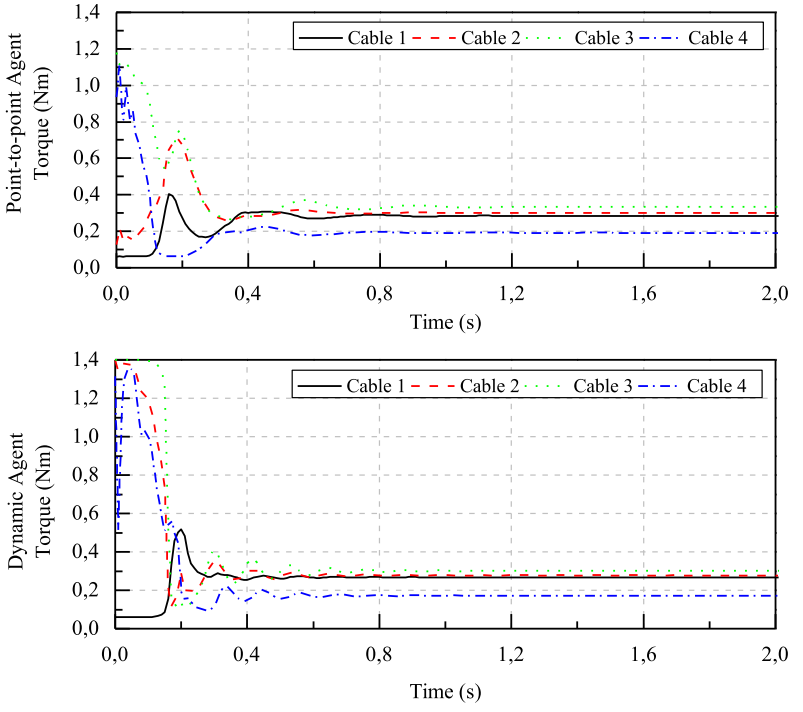


Figure 10. *Control input generated by the point-to-point and dynamic agent for the point-to-point (step) task.*

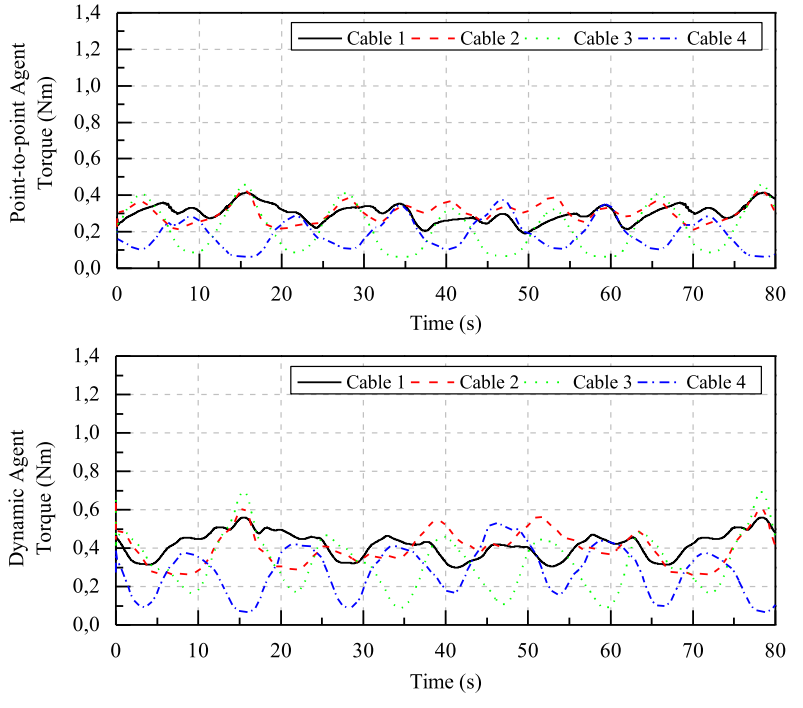


Figure 11. *Control input generated by the point-to-point and dynamic agent for the dynamic reference tracking task.*

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

Table VI. *Control parameters of the PD controller.*

K_P	diag(0.15, 0.15, 50)	τ_{max} (Nm)	1.4
K_D	diag(1.8, 1.8, 2)	τ_{min} (Nm)	0.06

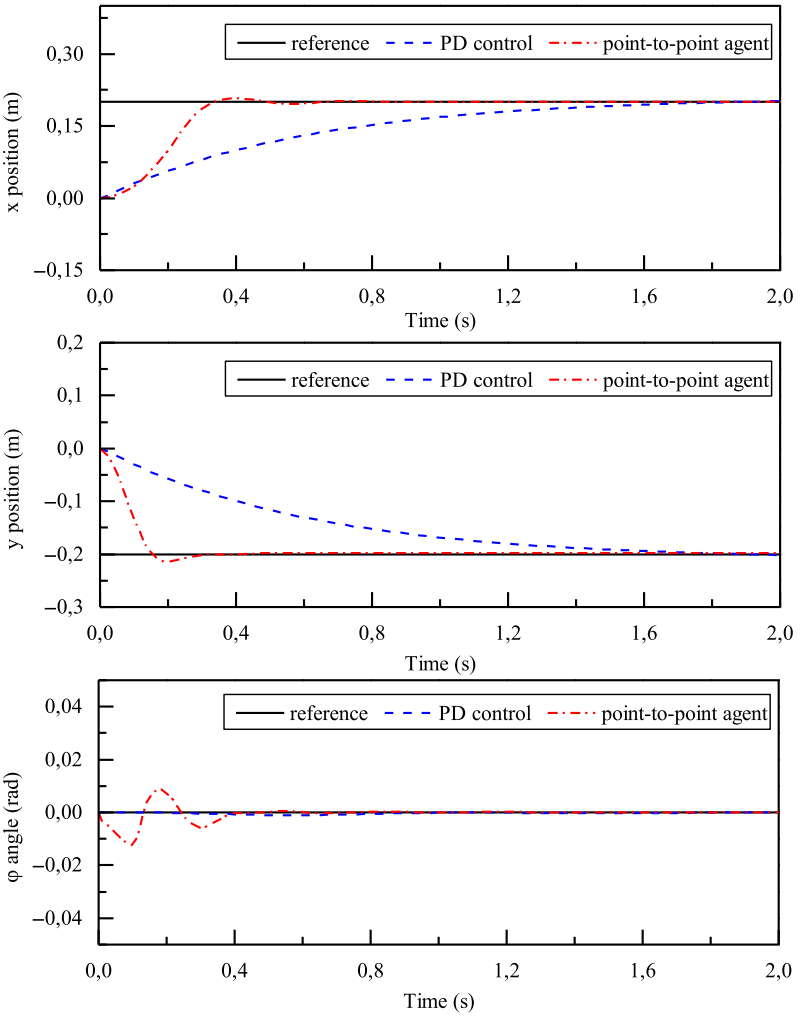


Figure 12. *Control response of the PD controller and the point-to-point agent for the point-to-point (step) task.*

The dynamic agent's performance is better in the x state control, while the point-to-point agent is better in the y state control. For the φ angle, the performance of keeping the state in the zero position when other states are moved to the target position is investigated. At the beginning of the control, a corruption in φ angle control appears then immediately decreases. The corruption is up to 0.08 rad in the case of using the dynamic agent for control, and steady-state error occurred, but it can be seen that the point-to-point agent control is successful, and the corruption is up to 0.01 rad. This situation can be considered as an advantage of training the agent for the point-to-point task. In general, the point-to-point agent is better than the dynamic agent in controlling the point reference. That is an expected outcome since the point-to-point agent is trained specifically for that task. However, the dynamic agent also performed quite well on the point-to-point task nevertheless it was trained for the dynamic references.

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

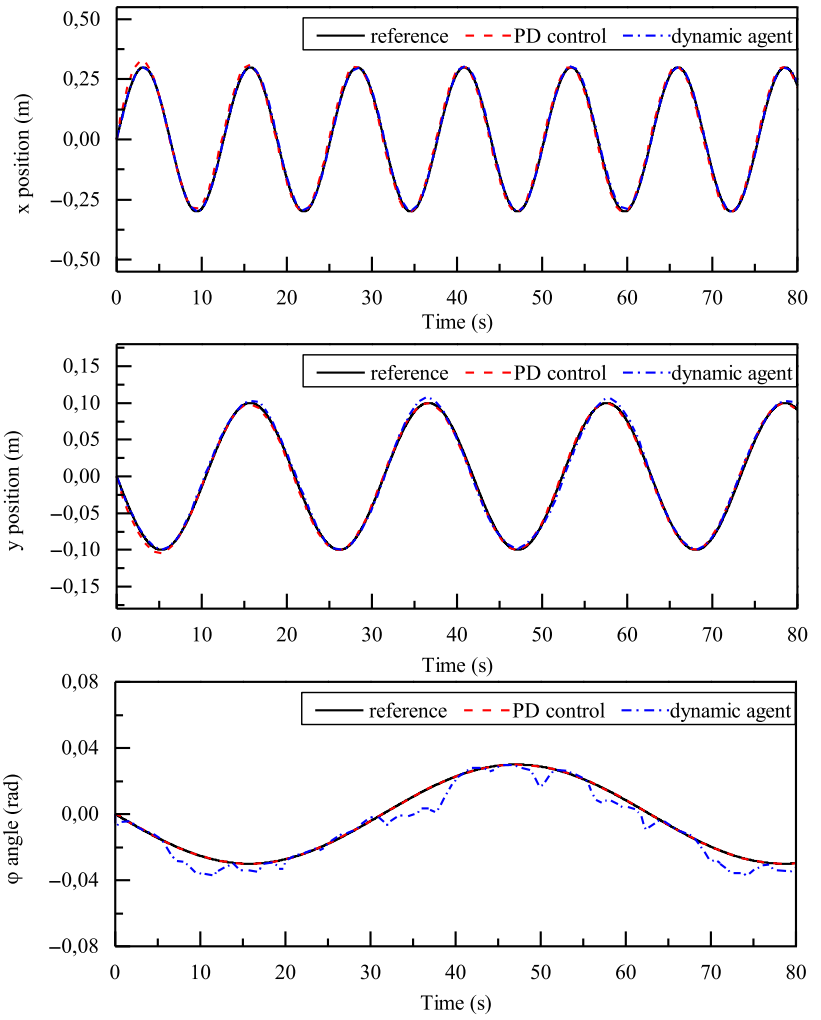


Figure 13. *Control response of the PD controller and dynamic agent for the dynamic reference tracking task.*

Second, the dynamic reference tracking results of the CDPR are given in Fig. 9 for both agents. The RMSE values are shown in Table V. The point-to-point agent's performance is as good as the dynamic agent's. This shows that the designed RL algorithm is capable of learning the required action for the control regardless of training task type. However, some deteriorations in tracking the peek points of the sinus reference were observed. This should be due to insufficient learning of the control signal at the corresponding pose of the end-effector.

Regarding control input for the point-to-point control, input torques (cable tensions) on the CDPR show the same behavior for dynamic and point-to-point agents in sequencing (Fig. 10). The generated control inputs by both agents are close to the minimum torque value of 0.06 Nm as expected because of the penalizing of the high torques in the learning process as a result of the proposed reward functions. In Fig. 11, the control torques for the dynamic reference tracking case are given for both agents. The figure shows that the dynamic agent's average control input is 0.3 Nm while the point-to-point agent's is 0.2 Nm. This implies that the dynamic agent applies more aggressive control, and the reward R_2, which penalizes the torques, is less effective in the dynamic agent training process. The control signals

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

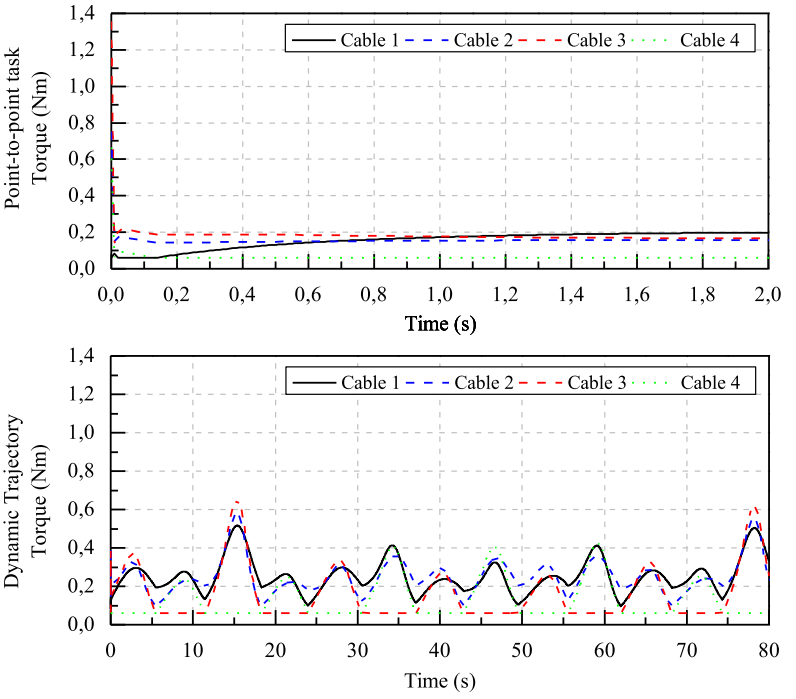


Figure 14. *Control input generated by the PD controller for the point-to-point (step) and dynamic reference tracking task.*

generated by the agents in both tasks are the same in terms of sequencing shows that the designed training algorithm produces stable results.

5.2. Comparison with a classical PD controller

In this section, the performance of the RL method is compared with a classical dynamic controller. For this purpose, a PD controller is designed end employed to control the CDPR dynamics for the same tracking tasks with the RL agents. For the sake of the brief representation, the design process of the PD controller is given briefly. One can refer to works [3, 8, 9] to have more information about the design process. The control input can be given as

$$\boldsymbol{\tau} = \boldsymbol{H}^{\dagger}\left(\boldsymbol{M}_{eq}\left(\ddot{\boldsymbol{X}}_d + \boldsymbol{K}_P\boldsymbol{e} + \boldsymbol{K}_D\dot{\boldsymbol{e}}\right) + \boldsymbol{N}\left(\boldsymbol{X}, \dot{\boldsymbol{X}}\right)\right) \tag{26}$$

where $\boldsymbol{H}^{\dagger} = \boldsymbol{H}^{T}(\boldsymbol{H}\boldsymbol{H}^{T})^{-1}$ is the pseudo-inverse of the matrix $\boldsymbol{H}$. $\boldsymbol{K}_D$, and $\boldsymbol{K}_P$ are PD controller gain parameters. The tracking error is defined as $\boldsymbol{e} = \boldsymbol{X}_d - \boldsymbol{X}$, where $\boldsymbol{X}_d$ is the desired state. In the construction of the control input, feedback linearization method is used. The control input in Eq. (26) may generate negative torques which is not desired for the cable robot. Therefore, a torque distribution algorithm [3] is used to keep cables always in tension. The PD controller parameters are given in Table VI. The parameters are chosen by trial and error until a proper response which satisfies the control input bounds is achieved. In the Figs. 12 and 13, the PD controller response is compared with the point-to-point agent and dynamic agent responses, respectively. The control inputs generated by the PD controller for the point-to-point (step) and dynamic reference tracking tasks are compared in Fig. 14.

The PD control responses' RMSE for dynamic reference tracking task are 0.016 m, 0.0031 m, and 0.0001 rad for x, y, and φ axes, respectively. The results are slightly better than the dynamic agent's response. However, for the point-to-point task, the RL agents' performance are much better than the PD controller. This shows that the agents learned well the control action for different cases, but the

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

PD controller performance depends on the task. Also, designing the PD control requires using a torque distribution algorithm, feedback linearization to analyze the stability, tuning the controller parameters with optimization algorithms in order to maintain the control constraints, which are time-consuming and high computational cost processes.

6. Conclusions

This paper proposes an RL controller that employs the DDPG algorithm to perform learning to control a planar CDPR. The controller does not require a tension distribution algorithm, as is the case in the design of classical controllers for CDPRs. As the CDPR used in this study has three degrees of freedom (two translations and one rotation) and four actuators, which causes an over actuation problem, an infinite number of tension distribution solutions are found to control the CDPR in a specified pose. To overcome this problem, the RL algorithm is used to find the optimal control input and tensions for a given task based on the reward functions concurrently. This also saved us from off-line optimization of tension distribution and controller parameters separately. The RL agents trained for the point-to-point and dynamic reference tracking tasks performed successfully for various sinusoidal signals; however, slight distortions and tracking errors are observed for the point-to-point agent when the frequencies of the signals are increased. The achieved continuous-time control results with the RL method are acceptable, and the RL agent can be used for real-time applications since the required computation time is low. It may be possible to obtain more precise results by using classical control methods for a specified task. However, a controller tuned for a specific task may not provide the desired performance, even for a slightly changed reference in amplitude or frequency. This situation is investigated for a PD controller, and the controller performed well for the dynamic reference tracking while its point-to-point task response is quite worse than the RL agents. We also propose a novel reward function that has an immense effect on the controller performance. The reward function enables the optimization of tension distribution and control effort properly for different control tasks.

Since the implementation of RL on the CDPRs is very new, there are many issues to be investigated, such as vibration and kinematic issues of the CDPRs, experimental performance of the RL, and implementation of RL on different CDPR structures. As future work, we shall investigate the performance of RL on controlling an under constrained, cable suspended CDPR.

Author contributions. CS and MI conceived and designed the study. CS designed and performed simulations. FY performed simulations and edited figures. CS, FY, and MI wrote the article. MI edited the article.

Funding. This research received no specific grant from any funding agency, commercial, or not-for-profit sectors.

Competing interests. The authors declare none.

References

[1] S. Qian, B. Zi, W. W. Shang and Q. S. Xu, "A review on cable-driven parallel robots," *Chin. J. Mech. Eng.* **31**, 66 (2018).
[2] R. Kumar and S. Mukherjee, "Enhanced dynamic capability of cable-driven parallel manipulators by reconfiguration," *Robotica* **39**(12), 2153–2171 (2021).
[3] R. L. Williams, P. Gallina and J. Vadia, "Planar translational cable-direct-driven robots," *J. Robotic. Syst.* **20**, 107–120 (2003).
[4] C. Sancak, F. Yamaç and M. Itik, "Forward kinematics and control of a planar cable driven parallel robot," *Konya J. Eng. Sci.* **7**, 862–874 (2019). doi: 10.36306/konjes.622283.
[5] P. Miermeister and A. Pott, "Modelling and Real-Time Dynamic Simulation of the Cable-Driven Parallel Robot IPAnema," **In:** *New Trends in Mechanism Science* (D. Pisla, M. Ceccarelli, M. Husty and B. Corves, eds). (Springer Netherlands, Dordrecht, 2010) pp. 353–360.
[6] M. Carricato and J. P. Merlet, "Stability analysis of underconstrained cable-driven parallel robots," *IEEE Trans. Robot.* **29**(1), 288–296 (2013).
[7] X. Diao and O. Ma, "Vibration analysis of cable-driven parallel manipulators," *Multibody Syst. Dyn.* **21**, 347–360 (2009).

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press

[8] C. Sancak and M. Itik, "Out-of-plane vibration suppression and position control of a planar cable-driven robot," *IEEE/ASME Trans. Mechatron.* 1 (2021). doi: 10.1109/TMECH.2021.3089588.
[9] M. A. Khosravi and H. D. Taghirad, "Robust PID control of fully-constrained cable driven parallel robots," *Mechatronics* **24**(2), 87–97 (2014).
[10] 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* **40**(1), 1–37 (2021).
[11] M. H. Korayem, H. Tourajizadeh, M. Jalali and E. Omidi, "Optimal path planning of spatial cable robot using optimal sliding mode control," *Int. J. Adv. Robot. Syst.* **9**, 168 (2012).
[12] J.-H. Bak, J. H. Yoon, S. W. Hwang and J. H. Park, "Sliding-Mode Control Of Cable-Driven Parallel Robots with Elastic Cables, **In:** 2016 *16th International Conference on Control, Automation and Systems (ICCAS)*, (IEEE, Gyeongju, South Korea, 2016) pp. 1057–1060.
[13] H. Bayani, M. T. Masouleh and A. Kalhor, "An experimental study on the vision-based control and identification of planar cable-driven parallel robots," *Robot. Auton. Syst.* **75**(4), 187–202 (2016).
[14] W. Shang, B. Zhang, B. Zhang, F. Zhang and S. Cong, "Synchronization control in the cable space for cable-driven parallel robots," *IEEE Trans. Ind. Electron.* **66**(6), 4544–4554 (2018).
[15] B. Zhang, W. Shang, S. Cong and Z. Li, "Coordinated dynamic control in the task space for redundantly actuated cable-driven parallel robots," *IEEE/ASME Trans. Mechatron.* **26**(5), 2396–2407 (2020).
[16] Y. L. Wang, K. Y. Wang, Y. J. Chai, Z. J. Mo and K. C. Wang, "Research on mechanical optimization methods of cable-driven lower limb rehabilitation robot," *Robotica* **40**(1), 154–169 (2021).
[17] A. Pott, "An Improved Force Distribution Algorithm for Over-Constrained Cable-Driven Parallel Robots," **In:** *Computational Kinematics* (F. Thomas and A. P. Gracia, eds.) (Springer Netherlands, Dordrecht, 2014) pp. 139–146.
[18] M. Gouttefarde, J. Lamaury, C. Reichert and T. Bruckmann, "A versatile tension distribution algorithm for n -DOF parallel robots driven by $n + 2$ cables," *IEEE Trans. Robot.* **31**(6), 1444–1457 (2015).
[19] B. Ouyang and W. Shang, "Rapid optimization of tension distribution for cable-driven parallel manipulators with redundant cables," *Chin J. Mech. Eng.* **29**, 231–238 (2016).
[20] A. Ghasemi, "Application of linear model predictive control and input-output linearization to constrained control of 3D cable robots," *Modern Mech. Eng.* **1**, 69–76 (2011).
[21] 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 Autom. Sin.* **36**, 459–464 (2010).
[22] H. Xiong, L. Zhang and X. Diao, "A learning-based control framework for cable-driven parallel robots with unknown Jacobians," *Proc Inst. Mech. Eng. I J. Syst. Control Eng.* **234**, 1024–1036 (2020).
[23] N. Akalin and A. Loutfi, "Reinforcement learning approaches in social robotics," *Sensors* **21**, 1292 (2021).
[24] T. Zhang and H. Mo, "Reinforcement learning for robot research: A comprehensive review and open issues," *Int. J. Adv. Robot. Syst.* **18**(3), 172988142110073 (2021).
[25] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. 2nd edition (The MIT Press, Cambridge, MA, 2018).
[26] T. Ma, H. Xiong, L. Zhang and X. Diao, "Control of a Cable-Driven Parallel Robot via Deep Reinforcement Learning," **In:** *2019 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*, IEEE, Beijing, China, IEEE, Beijing, China, (2019) pp. 275–280.
[27] H. Xiong, T. Ma, L. Zhang and X. Diao, "Comparison of end-to-end and hybrid deep reinforcement learning strategies for controlling cable-driven parallel robots," *Neurocomputing* **377**, 73–84 (2020).
[28] J. Oyekan and A. Grimshaw, "Applying deep reinforcement learning to cable driven parallel robots for balancing unstable loads: a ball case study," *Front. Robot. AI* **7**, 195 (2021).
[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, Playing Atari with deep reinforcement learning, 2013. arXiv: 1312.5602 [cs].
[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, "Human-level control through deep reinforcement learning," *Nature* **518**(7540), 529–533 (2015).
[31] 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, 2019. arXiv: 1509.02971 [cs, stat].
[32] S. Fujimoto, H. Hoof and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods, **In:** *Proceedings of the International Conference on Machine Learning*, PMLR (2018) pp. 1587–1596.
[33] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," **In:** *Proceedings of the International Conference on Machine Learning*, PMLR (2016) pp. 1928–1937.
[34] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra and M. Riedmiller, "Deterministic Policy Gradient Algorithms," **In:** *Proceedings of the International Conference on Machine Learning*, PMLR (2014) pp. 387–395.

Cite this article: C. Sancak, F. Yamac and M. Itik (2022). "Position control of a planar cable-driven parallel robot using reinforcement learning", Robotica **40**, 3378–3395. https://doi.org/10.1017/S0263574722000273

https://doi.org/10.1017/S0263574722000273 Published online by Cambridge University Press