International Journal of Control, Automation, and Systems 21(11) (2023) 3507-3518
http://dx.doi.org/10.1007/s12555-023-0342-6

ISSN:1598-6446 eISSN:2005-4092
http://www.springer.com/12555

Compensated Motion and Position Estimation of a Cable-driven Parallel Robot Based on Deep Reinforcement Learning

Huaishu Chen, Min-Cheol Kim, Yeongoh Ko, and Chang-Sei Kim*

Abstract: Unlike conventional rigid-link parallel robots, cable-driven parallel robots (CDPRs) have distinct advantages, including lower inertia, higher payload-to-weight ratio, cost-efficiency, and larger workspaces. However, because of the complexity of the cable configuration and redundant actuation, model-based forward kinematics and motion control necessitate high effort and computation. This study overcomes these challenges by introducing deep reinforcement learning (DRL) into the cable robot and achieves compensated motion control by estimating the actual position of the end-effector. We used a random behavior strategy on a CDPR to explore the environment, collect data, and train neural networks. We then apply the trained network to the CDPR and verify its efficacy. We also addressed the problem of asynchronous state observation and action execution by delaying the action execution time in one cycle and adding this action to be executed to match the motion control command. Finally, we implemented the proposed control method to a high payload cable robot system and verified the feasibility through simulations and experiments. The results demonstrate that the end-effector position estimation accuracy can be improved compared with the numerical model-based forward kinematics solution and the position control error can be reduced compared with the conventional open-loop control and the open-loop control with tension distribution form.

Keywords: Cable-driven parallel robot, deep reinforcement learning, motion control.

1. INTRODUCTION

A cable-driven parallel robot (CDPR) is a type of parallel manipulator configured and actuated by several flexible cables different from the conventional rigid links parallel robots [1]. A CDPR is composed of multiple kinematic chains connecting the base to the end-effector. These chains are flexible and stretched cables whose length and tension are controlled by motor-winches. Compared with rigid-link parallel robots, these cables have less mass than rigid bodies, and it is easy to change the cable length by winding and unwinding the winch. Then, a cable robot can achieve low inertia, high payload-to-weight ratio, and larger workspaces [2,3].

Actuation redundancy is essential for the structure of parallel robots which improves the kinematic and dynamic properties of parallel robots and extends the workspace size for a CDPR by enhancing stiffness and avoiding singularity [4]. Interestingly, because the limitation of the cables is that cables must remain in tension; cables can only impose tensile forces, not compressive forces [5], more cables than the required degree of freedom are used and it causes a redundancy problem while computing forward kinematics.

In addition, conventional motion control methods such as the widely used proportional-integral-derivative (PID) controller [6], a position correction method using cable tension distribution by inverse dynamics [7,8], and an adaptive controller [9] have been incorporated for a CDPR control. Most conventional controllers are based on ideal inverse kinematics and designed to compensate for the cable length obtained by the inverse kinematics with encoder feedback at each motor [10]. However, because of the higher material-dependent properties of a cable, the flexibility of the cable and the uncertainties of the kinematic model also have to be considered to improve control performances. Especially, the tension distribution requires a precise dynamic model of a CDPR with uncertain model friction and cable elongation. It is almost impossible to render an accurate numerical model and parameters without any strong assumptions, even under well-established experimental conditions. Alternatively, an adaptive con-

Manuscript received June 5, 2023; revised August 23, 2023; accepted September 19, 2023. Recommended by Guest Editor Juhoon Back under the direction of Editor-in-Chief Hyo-Sung Ahn. This study was supported by “Regional Innovation Strategy (RIS)” through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(MOE) (2021RIS-002), Chonnam National University (Grant number: 2023-0095-01), and Korea Institute for Advancement of Technology(KIAT) grant funded by the Korea Government(MOTIE (P0008473, HRD Program for Industrial Innovation). Huaishu Chen and Min-Cheol Kim equally contributed to this work.

Huaishu Chen, Min-Cheol Kim, Yeongoh Ko, and Chang-Sei Kim are with the Department of Mechanical Engineering, Chonnam National University, 77 Yongbong-ro, Buk-gu, Gwangju 61186, Korea (e-mails: {h.s.chen8148, kmc100291}@gmail.com, kyo808@naver.com, ckim@jnu.ac.kr).
* Corresponding author.

troller could estimate several adapted parameters to reduce the effect of parametric uncertainties in the kinematic and dynamic models. However, state feedback from the forward kinematics requires additional sensors at an end-effector with a complex kinetic model. The kinematic calibration method for CDPRs considering the pulley kinematics is also used to improve the accuracy of pose estimation [11]. If the system is cost-independent, then, additional internal or external sensors such as IMU sensor [12] and optical sensor [13] can be used to provide more exact information of the end-effector to increase the accuracy of position estimation. Upon this background, a network algorithm along with model-based approaches was introduced to improve the control performances [14].

Overall, the model-based control and estimation could perform moderate results for a CDPR if the incorporated kinematics and dynamic models are reasonable depending on the applications. However, it still needs to consider uncertainties and frictions at various operating conditions mainly related to cable elongations that hinder obtaining exact forward kinematics solutions under less number of sensors.

In contrast, reinforcement learning (RL) is a semi-supervised data-driven machine learning method that trains an agent to perform the desired control task without prior knowledge of the robot's kinematics and dynamics. The agent is trained to interact with uncertain or unknown parameters and learns to optimize its behavior by maximizing a predefined performance in sequential decision-making problems [15]. RL-based compensation methods use the actor-critic approach [16] to learn the correction signal added to the existing nominal input to compensate for aberrations [17].

Recent advances in deep RL (DRL) have emerged as a promising approach for autonomously acquiring complex behaviors from low-level sensor observation [18]. DRL uses expressive function approximations (deep neural networks) and has achieved great success in solving complex tasks such as games robotic control [19-23], and industrial control scenarios [24]. An offline RL was developed to advance the RL which collects experience iteratively by interacting with the environment. The offline RL addresses the problem of learning effective policies entirely from previously collected data without additional online data collection [25].

On these backgrounds, we present a novel forward kinematic computation and control approach for a CDPR by introducing DRL as shown in Fig. 1. Eventually, the DRL can produce the updated trajectory to overcome the aforementioned difficulties of CDPR control against inaccurate kinematic and kinetic model including cable properties. The contribution of this study is a compensated motion control method with forward kinematics estimation through the offline deep RL algorithm. We used a random behavior strategy on a CDPR to explore the environment,

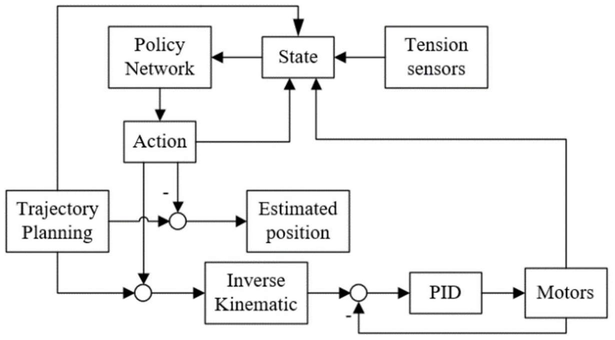


Fig. 1. Block diagram of proposed DRL-based CDPR control and position estimation.

collect data, and train neural networks. We then applied the trained neural network to the CDPR and verify its efficacy on the prototyped eight-cable CDPR. To encounter a practical application, we also propose an approach to avoid the latency between state measurement and action execution that is effective for dynamic simulation. Finally, we validated the proposed network to estimate the actual position of the CDPR that can be used as an alternative to forward kinematic computation for a CDPR.

2. SYSTEM OVERVIEW

2.1. The CDPR and measurements

The CDPR for data collection and policy evaluation is shown in Fig. 2. This CDPR is fully constrained with eight cables equipped with tension sensors (tecsis GmbH, Germany) at the end of each cable [26]. Motor-winches are connected to cables and controlled by servo drivers, where the TwinCAT3 PLC and TwinCAT3 Engineering are used [26]. This software provides an ST language programming environment that can implement the real-time control algorithm to the PLC.

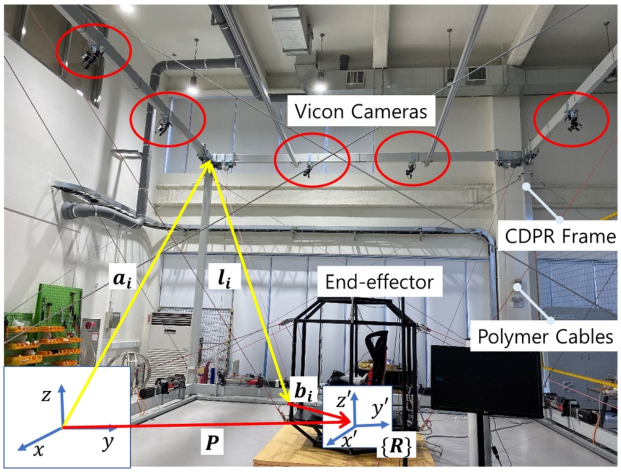


Fig. 2. Overview of a high-payload CDPR.

The Vicon camera measurement system is used to provide the position information of the CDPR, as depicted in Fig. 2. This system is composed of eight cameras evenly fixed on the base frame of the CDPR in pairs and measures the pose of the end-effector through six special reflective balls installed on the top of the end-effector. The camera system provides measured data with less than 0.2 mm error. The Vicon system is used to obtain the pose information of the end-effector that calculates the reward and position error only, not for feedback control.

2.2. Kinematic of CDPRs

The CDPR is a closed kinematic chain mechanism in that the end-effector is connected to the base frame by several cables [5]. The kinematics notation of a CDPR with eight cables is shown in Fig. 2. The large outer structure represents the base frame of the CDPR and the inner black frame represents the end-effector. The x, y, and z axes are the base coordinate system of the CDPR and the x', y', and z' axes are the relative coordinate system of the end-effector, $\boldsymbol{a}_i$ is the location vector of each endpoint of the base of the CDPR, $\boldsymbol{b}_i$ is the location vector of each endpoint of the end-effector of the CDPR, $\boldsymbol{l}_i$ is the vector of each cable connecting the base and the end-effector, and $\boldsymbol{P}$ is the position vector of the end-effector.

We can then express the inverse kinematic of the eight-cable CDPR as

$$l_i = |l_i| = |\boldsymbol{a}_i - \boldsymbol{P} - \boldsymbol{R}\boldsymbol{b}_i|, \tag{1}$$

where l_i is the length of each cable, and $\boldsymbol{R}$ is the rotation matrix of the end-effector.

Different from the inverse kinematics, there exists no unique forward kinematics solution for a CDPR. Therefore, various complicated computation algorithms are studied for the forward kinematics solver.

The dynamic model of a fully constrained CDPR with the cable tension distribution is derived as follows with respect to the general coordinate

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} = A^T\mathbf{f} \text{ and } \mathbf{f} = \mathbf{f_M} + \mathbf{f_V}, \tag{2}$$

$$\mathrm{f_{M,i}} = (\mathrm{f_{min}} + \mathrm{f_{max}})/2, \tag{3}$$

$$A^T\mathbf{f_V} = -\mathbf{w_p} - A^T\mathbf{f_M},$$

$$A^T = \begin{bmatrix} u_1 & \dots & u_m \\ b_1 \times u_1 & \dots & b_m \times u_m \end{bmatrix}, \tag{4}$$

$$u_i = \frac{l_i}{\|l_i\|_2} \text{ and } \mathbf{w_p} = \begin{bmatrix} \boldsymbol{f}_p \\ \boldsymbol{\tau}_p \end{bmatrix}, \tag{5}$$

where $\boldsymbol{q}$ is a generalized coordinate in wrench space, $\mathbf{f}$ is a cable tension vector designed by a desired mean cable tension, $\mathbf{f_M}$, and a control tension to cables, $\mathbf{f_V}$. The external wrench force vector is defined as $\mathbf{w_p}$. $\mathrm{f_{min}}$ and $\mathrm{f_{max}}$ are a minimum and a maximum tension depending on the motor-winch specification, respectively, and A^T is a transformation matrix of a parallel manipulator composed of cable direction vectors and rotation transformations.

2.3. Reinforcement learning

The designed workflow of the network considering physical hardware setups is depicted in Fig. 3. The goal of RL in the proposed method is to address the problem of learning to control an agent via policy $\pi(act_t \mid s_t)$ attempting to maximize the expected sum of rewards [11].

We consider this problem as a Markov decision process (MDP). The fully observed MDP can be defined as a tuple $M = (S, A, T, d_0, r, \gamma)$, where S is the state space, A is the action space, T is the dynamics of the system formatted as $T = (\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{act}_t)$ where $\boldsymbol{act}$ is the action vector and $\boldsymbol{s}$ is a state vector, d_0 is the initial state distribution, r is the reward, and γ is the discount factor. The MDP can be expressed with the following steps at every time step t

Step 1: Observe the current state $\boldsymbol{s}_t$ and sample an action $\boldsymbol{act}_t$ from its policy distribution $\pi(\boldsymbol{act}_t \mid \boldsymbol{s}_t)$.

Step 2: The agent performs the action, $\boldsymbol{act}_t$, and the product of the state $\boldsymbol{s}_{t+1}$ is calculated using the environment transition function T.

Step 3: Observe the resulting next state $\boldsymbol{s}_{t+1}$ and reward value $r_t = r(\boldsymbol{s}_t, \boldsymbol{act}_t)$.

Step 4: Use the reward value to evaluate and update the policy.

Step 5: Obtain the sequence of states and actions of length H given by $\tau = (\boldsymbol{s}_0, \boldsymbol{act}_0, \cdots, \boldsymbol{s}_H, \boldsymbol{act}_H)$, and its distribution as

$$p_\pi(\tau) = d_0(\boldsymbol{s}_0)\sum_{t=0}^{H}\pi(\boldsymbol{act}_t \mid \boldsymbol{s}_t)T(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{act}_t), \tag{6}$$

and the RL objective can then be written as

$$L(\pi) = E_{\tau \sim p_\pi(\tau)}\left[\sum_{t=0}^{H}\gamma^t r(\boldsymbol{s}_t, \boldsymbol{act}_t)\right]. \tag{7}$$

There are two stages to optimize this objective function. The first is policy gradients. Assume that the policy is pa-

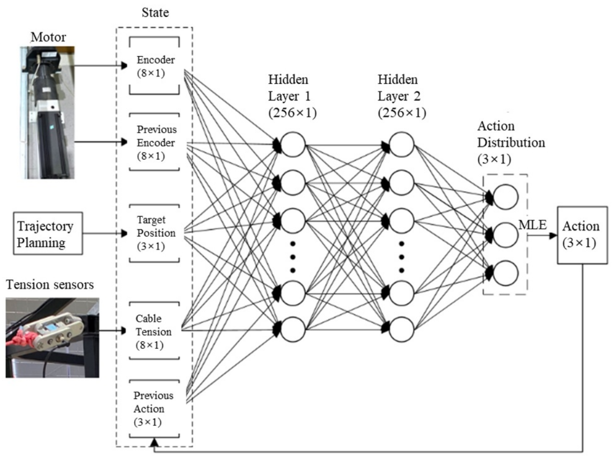


Fig. 3. Two-layer MLP and physical workflow of designed network.

rameterized by a parameters vector $\boldsymbol{\theta}$. We can then optimize parameters as $\boldsymbol{\theta} = \arg max_{\boldsymbol{\theta}} L(\pi)$. The second is the approximated dynamic programming to define the value function which estimates the expected cumulative reward. If the sequence τ starts from a state-action tuple $(\boldsymbol{s}_t, \boldsymbol{act}_t)$, the value function can be written as

$$Q_{\boldsymbol{\phi}}(\boldsymbol{s}_t, \boldsymbol{act}_t) = E_{\tau \sim p_\pi(\tau|\boldsymbol{s}_t, \boldsymbol{act}_t)} \left[\sum_{t'=t}^{H} \gamma^{t'-t} r(\boldsymbol{s}_t, \boldsymbol{act}_t) \right]. \quad (8)$$

We then obtain the transformed expectile regression objective which is called the temporal different (TD) target as follows:

$$L(Q_{\boldsymbol{\phi}}) = (Q_{\boldsymbol{\phi}}(\boldsymbol{s}_t, \boldsymbol{act}_t) - r(\boldsymbol{s}_t, \boldsymbol{act}_t) - \gamma \max_{\boldsymbol{act}_{t+1}} Q_{\boldsymbol{\phi}}(\boldsymbol{s}_{t+1}, \boldsymbol{act}_{t+1}))^2, \quad (9)$$

where $\boldsymbol{\phi}$ is the parameter vector of $Q_{\boldsymbol{\phi}}$.

And then, the parameters can be optimized by $\boldsymbol{\phi} = \arg\max_{\boldsymbol{\phi}} L(Q_{\boldsymbol{\phi}})$. Finally, the policy is transformed into a specific action $\boldsymbol{act}_t = \arg\max_{a_t} Q_{\boldsymbol{\phi}}(\boldsymbol{s}_t, \boldsymbol{act}_t)$. In the offline RL problem, the policy can be learned by affixed dataset $D = \{(\boldsymbol{s}_t^i, \boldsymbol{act}_t^i, \boldsymbol{s}_{t+1}^i, r_t^i)\}_{i=0}^n$, which is collected using a behavior policy, $\pi(\boldsymbol{act}_t \mid \boldsymbol{s}_t)$.

2.4. Network determination

We incorporated the implicit Q-learning (IQL) which is an offline RL algorithm that archives the state-of-the-art performance on D4RL, a standard benchmark for offline RL [27]. The IQL algorithm comprises expertise regression and advantage-weighted behavioral cloning to focus on in-sample actions and avoid querying out-of-sample actions. This approach is in the right place to address the issue of distribution shift [28,29], one of the primary challenges in offline RL. Based on the structure of the IQL, we defined values to determine the structure of our networks as follows:

State space $\boldsymbol{S}$: We divide the state into five parts: 1) the encoder value of each motor when the previous action was executed, with a dimension of 8 in our experiment, 2) the current encoder value of each motor, also with a dimension of 8, 3) the desired position of the end-effector when this action is completed with a position vector dimension of 3, 4) tension feedback of each cable with dimension 8, and 5) action to be performed with dimension 3. Thus, the dimension of the state space is 30.

Action space $\boldsymbol{A}$: As the output of the policy network, actions are represented as motion compensation of the CDPR based on the current state, with a dimension of 3. We set the output action as a three-dimensional Gaussian distribution in the interval $[-1, 1]$ and map all action values collected into this interval with $\max(\boldsymbol{act}_t) < 0.3$ mm. The Gaussian distribution is used to represent the action to increase the exploration of the action space during training, and the expected value of the distribution will be used as the action during validation. The action is sampled from this Gaussian distribution when training the network. After training, the action is the estimated mean of the Gaussian distribution. Thus, at any time, the compensation value output by the policy network in any coordinate axis direction is less than 0.3 mm.

Reward function $\boldsymbol{r}$: The reward is a critical value in network training that determines how good or bad the current policy is. We define the reward function as

$$r(\boldsymbol{s}_t, \boldsymbol{act}_t) = -\|\boldsymbol{p} - \boldsymbol{p}_{ref}\|_2 - c\boldsymbol{act}_t^T \boldsymbol{act}_t, \quad (10)$$

where $\boldsymbol{p}$ is the actual position of the end-effector, $\boldsymbol{p}_{ref}$ is the reference position of the end-effector, $\boldsymbol{act}_t$ is the currently executing action, and c is a constant determining the effect of action size on policy. Larger values of c prompt the policy to learn to make an action smaller. We set $c = 10$ in our experiment.

3. DRL-CONTROLLER DESIGN

3.1. Action loop time determination

In theory, a shorter action loop time can produce high performance. However, hardware-dependent conditions such as computation time and sensor delay need to be considered, so it is crucial to choose an appropriate loop time according to the given experimental platform. In this study, the TwinCAT3 software provides an integer time greater than or equal to 1 msec as the control loop time. We selected a 2 msec control loop time to allow the processor sufficient computation time. For the Vicon camera measurement system, we choose the fastest measurement frequency of 100 Hz.

As a communication protocol of TwinCAT3, an automation device specification (ADS) API enabling users to communicate with programs outside the TwinCAT3 is available, in which an external program reading or writing takes time; TwinCAT3 consumes approximately 13-17 msec. The TwinCAT3 measures an observation (state), sends it to the external running policy network program, and receives the result of the policy, which process requires approximately 26-34 msec. Combining the above timeline, we set the action loop time to 40 msec. The time sequence of the CDPR running is depicted in Fig. 4, where T_n is an action applied time sequence, and t_n is the motor's command applied time sequence in one action cycle.

3.2. Action execution time adjustment

The MDP formulation assumes synchronous execution; the observed state remains unchanged until the action is applied [18]. Because the robot's motion is continuous, the observation must be performed simultaneously as the action is to ensure the assumption of MDP formulation. The specific observation and action time are depicted in Fig. 5(a). Because of delays caused by factors such as

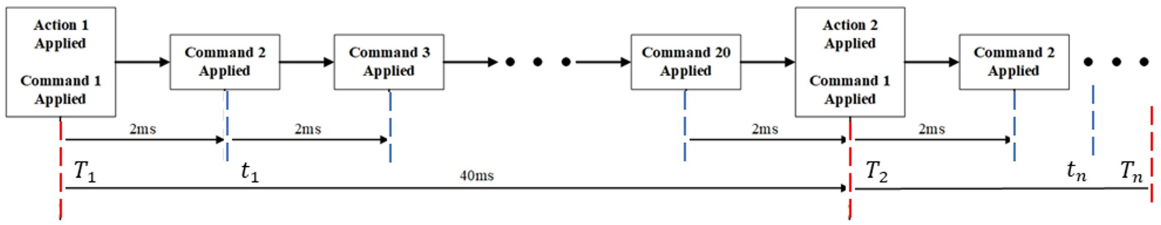
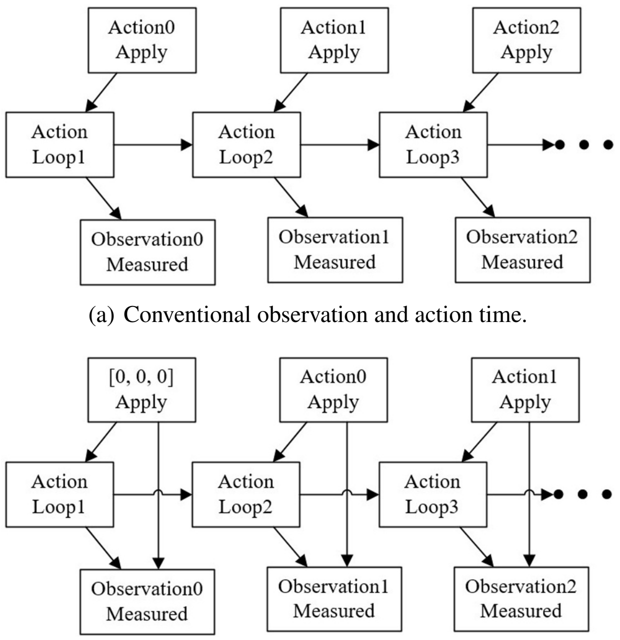


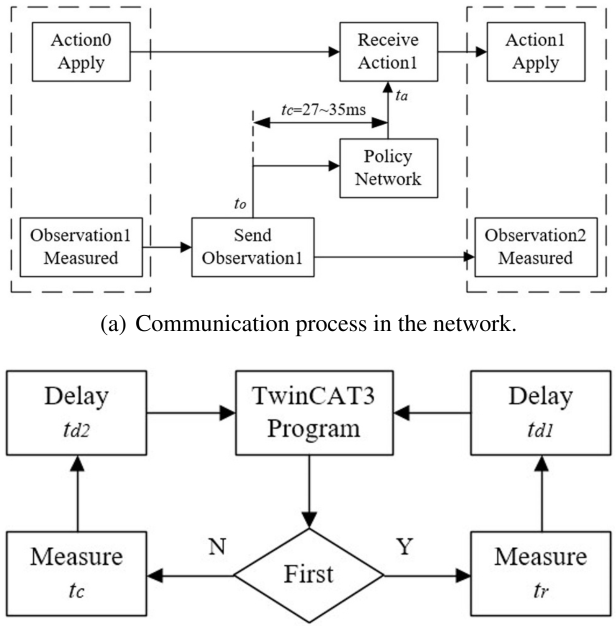
Fig. 4. Time sequence design for DRL approach.



(a) Conventional observation and action time.

(b) Adjusted observation and action time.

Fig. 5. Proposed sequential adjustment of action execution time.



(a) Communication process in the network.

(b) Timeline matching.

Fig. 6. Communication process timeline between TwinCAT3 and the policy program.

the communication described earlier, the CDPR controller cannot obtain the corresponding required actions and observations simultaneously. This results in accumulated errors at the end of the trajectory for the case of the open-loop control scheme in this study. Therefore, we mitigated the impact of this problem by delaying the execution of all actions by one action loop time and adding the action, executed at the current cycle time to the state. The rewards of actions are also delayed. The new specific observation and action time are depicted in Fig. 5(b), resulting in 40 msec for the entire process from acquiring the state to acquiring the executable action.

3.3. Timeline matching

The specific communication process timeline between TwinCAT3 and the policy program is depicted in Fig. 6(a), t_0 is the time when TwinCAT3 starts sending the observation to the network program, t_a is the time when TwinCAT3 finishes receiving the action from the network program, and t_c is the total time spent in communication and network forward calculation. Obviously, to execute the correct action instructions for the robot, t_a should be less than T_1 (40 msec). To ensure this condition strongly, we designed the control sampling time after the network program read the observation from TwinCAT3 as $t_0 < (40 - t_c)/2$. In detail, we send t_0 and observation together to the network program, and then, apply the matching method as shown in Fig. 6(b). Where “First” is introduced to find if it is the first time entering the action loop and t_r is time spent reading observation. t_{d1} is the time delay for the first action cycle and t_{d2} is the time delay for the network program execution defined as

$$t_{d1} = 40 - (2 * (t_o - 1) + t_r),$$
$$t_{d2} = 40 - (2 * (t_o - 1) + t_c). \tag{11}$$

For the first action cycle, we obtain action[0] and observation[0] before running the CDPR. Therefore, the first t_0 may not satisfy this condition and we use it to guarantee that the second t_0 must satisfy this timeline. We ensure that the controller sends the observation that has been completely modified by selecting the second motor control cycle as the start time for the observation sending. To implement the timeline-matching method, all t_0 values are equal to 1 after the first action cycle.

3.4. Cost function and network implementation

To track the desired trajectory, the output state is defined as the position in the environment. Here, the action is the movement in that the agent regenerates an updated position from one state to another. The reward is given to increase the Q-value in (8) for the corrected action taken by the agent at the specific position. Therefore, during tracking control, the total cost of CDPR motion is related to the tracking error; accumulated position error. Herein, the ultimate goal of the control is simply defined to minimize the total cost as a Euclidean norm of a tracking error as follows that is considered to design the reward function in (10).

$$\text{cost} = \arg\min \sum \|p_{ref,i} - p_{actual,i}\|. \tag{12}$$

To achieve this goal, the DRL is applied to provide an updated trajectory for improved control accuracy. Herein, for the perspective of simple and effective application, we utilized a policy extraction and parameter optimization available in the IQL explanation in Section 2. Where the final algorithm is composed of fitting the value function and Q by using gradient updates and performing stochastic gradient descent on a modified TD learning procedure. While implementing, we used the official PyTorch implementation of the IQL package as a basis for building training algorithms [27]. We build all networks with a two-layer multilayer perceptron (MLP) with rectified linear unit (ReLU) activations and 256 hidden layers, including policy, two value functions, two target value functions, and a state value function. After obtaining the network trained with PyTorch, the LibTorch library is utilized to rebuild the network in C++.

3.5. Validation in simulation

In this simulation study for algorithm verification, the CDPR analysis and simulation platform for research (CASPR) is used for dynamics simulation on MATLAB [30]. The IPAnema2 type CDPR was selected from the CASPR database for the simulation because it is the identical configuration CDPR of the actual experimental robot in our study [31]. The training data consists of linear and circular trajectories. And the circular trajectory is on the OXY plane, the center is the origin, and the radius is

$$\text{radius} = 1.0 - 0.2k \text{ with } k = 0,\ 1,\ 2,\ 3,\ 4. \tag{13}$$

The final training and validation trajectories of the policy network are depicted in Fig. 7. The blue arrowed lines are the trajectories used for training, and the red ones are used for validation.

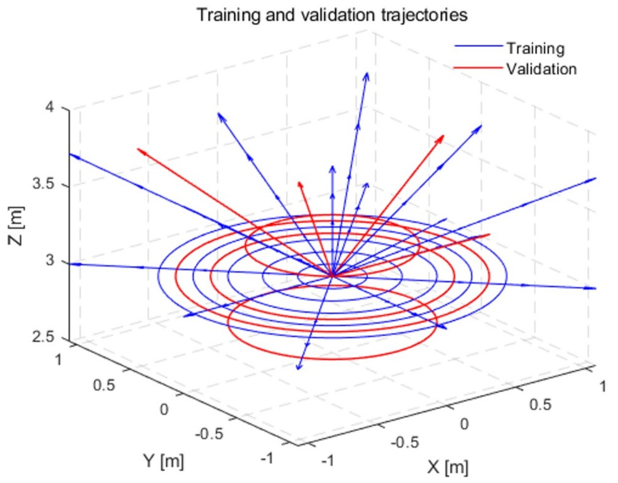


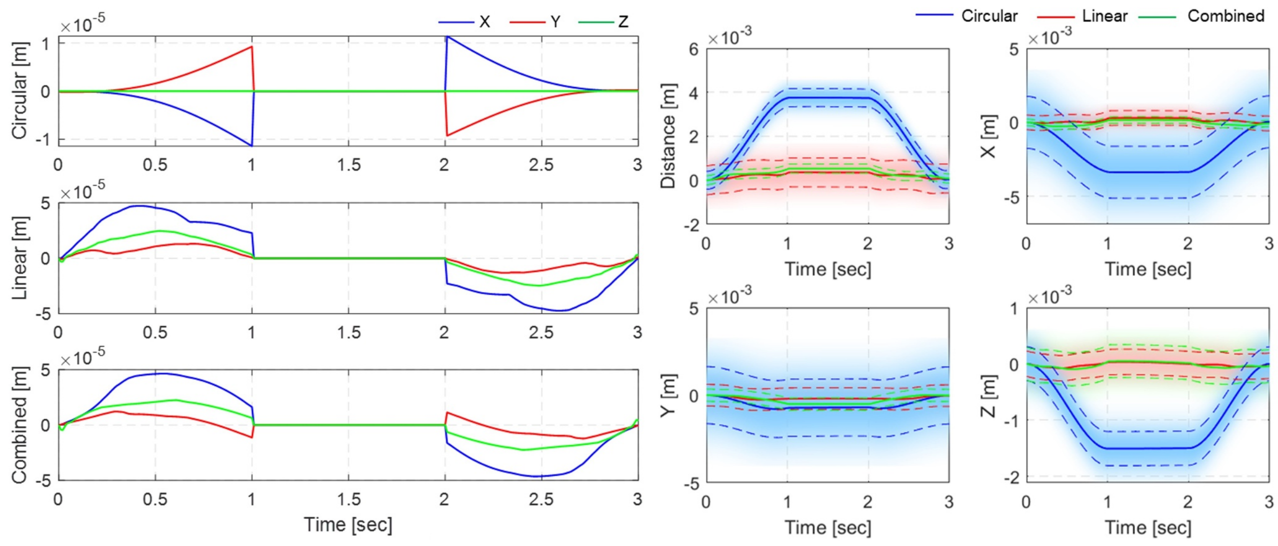
Fig. 7. Training and validation trajectories for simulation.

For the feasible verification of the algorithm, we designed the behavior policy in simulation with respect to the acceleration of the end-effector as follows:

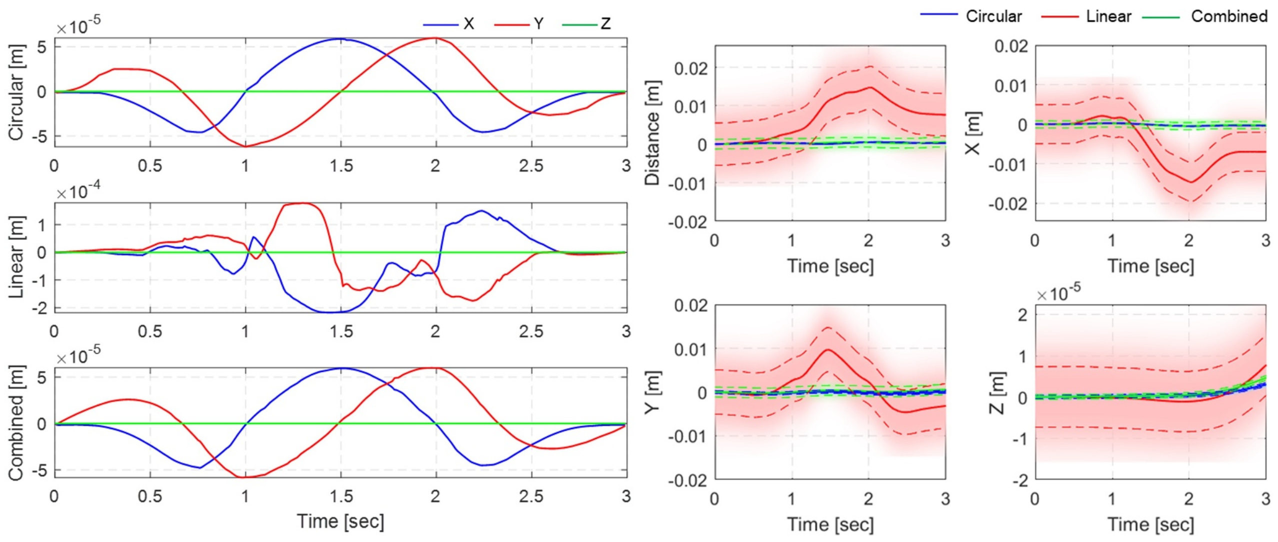
$$\boldsymbol{\pi}_{sim} = (1+\eta)m\boldsymbol{a}. \tag{14}$$

The simulation policy is based on a dynamic model of the CDPR in which the end-effector movement is dominated by acceleration which is directly controlled by cable tension distribution. In (14), m is the coefficient sampled from the truncated distribution [32] of $[0,\ 1.2e-4]$ for the linear trajectory and $[0,\ 2e-5]$ for the circular (changed every episode), η is the exploration noise sampled from the truncated distribution of $[-0.03,\ 0.03]$ and changed every action cycle, and $\boldsymbol{a}$ is the acceleration vector.

For the linear trajectory, the simulation interval was set at 10 msec, and the action execution interval was 10 msec. For the circular trajectory, the dynamics simulation interval was 2 msec, and the action execution interval was 10 msec. Each trajectory was repeated with noise disturbance 50 times for the 16 linear trajectories and 80 times for the 10 circular trajectories to collect data; a total of 800 datasets for each linear and circular trajectory were obtained with 300 steps per episode. The actions were transformed to $[-1,\ 1]$, and the reward was transformed to $[0,\ 1]$. The learning rate for the policy was 1×10^{-5}, the learning rate for the value function was 2×10^{-5}, and the discount for the reward was 0.99. The policy weights update was delayed by three cycles. We prepared datasets into three categories; linear dataset, circular dataset, and a combination of the linear datasets and the circular datasets to train three policy networks separately. And we ended up deriving three different policy networks; a linear policy, a circular policy, and a combined policy. We validate these three policy networks in linear trajectories with different



(a) Results on the linear trajectory by linear, circular, and combined policy; (LEFT) output actions on each direction, (RIGHT) position errors.



(b) Results on the circular trajectory by linear, circular, and combined policy; (LEFT) output actions on each direction, (RIGHT) position errors.

Fig. 8. Dynamic simulation results of proposed DRL-based control approach.

endpoints with the same starting point and circular trajectories with different radii for preliminary validation.

Fig. 8 illustrates the outputs (actions) of the three policies under the validated linear and circular trajectory and their effects on the robot motion error. The picture on the right shows the motion position error of the end-effector after applying the three strategy networks respectively. The blue line is the circular policy, the red line is the linear policy, and the green line is the combined policy. The linear policy performs well on validated linear trajectories. The tracking error of the robot is significantly reduced in the x, y, and z directions, but this policy performs poorly on circular trajectories, and its output is disordered. The simulation results of circular policy are the opposite, performing well on circular trajectories and poorly on linear trajectories. Obviously, the performance of a policy depends on the composition of its training data. Therefore, training data containing the corresponding type of trajectory data is a necessary condition for an excellent policy. For the combined policy, the simulation results show that the tracking performance is slightly worse than the linear policy on linear trajectory and circular policy on circular trajectory, but still exhibits very high performance. Therefore, we could confirm that it is feasible to first combine the data of all required motion trajectories, and then use this data to train a policy network that can be applied to all these trajectories. This is reasonable inference for a real CDPR control where the tension distribution and cable length of the CDPR are desired at each position in the workspace as shown in (1)-(5).

4. EXPERIMENTAL RESULTS

4.1. Data collection and practical behavior policy design

We conducted experiments by using an actual CDPR system as shown in Fig. 2. By considering the practically available measurements of the actual CDPR system; e.g., acceleration of the end-effector is not measurable, we defined a new behavior policy and implemented the DRL approach. Here, we assumed that the position error of the CDPR is proportional to the moving distance.

$$\boldsymbol{\pi}_{exp} = m_j \Delta x_j + \boldsymbol{\eta}_j \text{ and } j = x,\ y,\ z, \tag{15}$$

where Δx_j is the moving distance of the end-effector, m_j is the linear coefficient sampled from the truncated distribution of $[-0.02,\ 0.02]$ and changed every episode, and $\boldsymbol{\eta}_j$ is the exploration noise sampled from the truncated distribution of $[-0.1m\Delta x,\ 0.1m\Delta x]$ and changed every action cycle.

We arrange the following five motion paths (unit: meter) with the fifth-order trajectory planning and the end-effector stop for one second between any two parts. Each path is repeatedly tested 50 times, totaling 250 episodes. In Fig. 10(b), blue arrowed lines illustrate the approximate form of these trajectories. The last part of the path is homing the cable robot, so the behavior policy does not apply in this part. All data is transformed into reply buff $D = \{(\boldsymbol{s}_t^i, \boldsymbol{act}_t^i, \boldsymbol{s}_{t+1}^i, r_t^i, T_t^i)\}_{i=0}^{250}$, where T_t^i is whether the i-th episode is finished.

The TwinCAT3 control program runs 20 times in one action cycle. We ensure the smooth movement of the robot by dividing an action into 20 segments for execution according to the trajectory planning. The action performed in each segment is as

$$act_{tj}^i = \frac{d_j^i}{d_j^{20}} act_{tj} \text{ and } j = x,\ y,\ z, \tag{16}$$

where d_j^i is the distance from the i-th segment to the first segment. Therefore, the updated actual position input to the CDPR is obtained as follows:

$$\boldsymbol{p}_a = \boldsymbol{p}_r + \boldsymbol{act}_{cum} + \boldsymbol{act}_t^i, \tag{17}$$

where $\boldsymbol{p}_a$ is the adjusted position value, $\boldsymbol{p}_r$ is the reference position value, and $\boldsymbol{act}_{cum}$ is the cumulative actions.

4.2. Data matching for training data

The CDPR and the Vicon camera measurement system were implemented on different control computers; the CDPR system records the reference position data and the camera processor measures the actual position data of the end-effector. This brought a practical issue mentioned above and we need to match the data from both to calculate the reward while training. We designed a two-step method to address this problem as mentioned in Subsection 3.3. We use the motion start point in the data on the CDPR system based on a reference trajectory and search the motion start point of the data in the camera measurement system to match the data. Firstly, we set a movement threshold equal to 0.15 mm, and then, the time point when the CDPR movement exceeds this threshold is used as the starting point, c_1. Secondly, the method traverses each point in the interval $[c_1 - 700 \text{ msec},\ c_1 + 300 \text{ msec}]$ based on c_1 and search the time of minimum error between the two trajectories in 1 sec after the respective starting point. We then successfully take the point with the smallest error as the starting point of the motion recorded by the camera.

4.3. Training networks

After the buffer was fully and properly constructed, the network was trained by its datasets. The parameters for the network training are summarized in Table 1. Our model is trained on Google CoLab which provides some GPUs with limited usage time but was sufficient to achieve high performance. The typical computation time on Tesla T4 is about 5.4 s/epoch. The average return predicted by two value functions in the training process is depicted in Fig. 9. The prediction of both two value functions increases rapidly and becomes stable after 100 epochs.

Table 1. Parameters of network training.

Learning rate for policy	3e-5
Learning for value function	1e-5
Batch size	512
Learning epoch	300
Gradient steps per epoch	600
Discount γ	0.99
Beta	5.0
Quantile	0.85
Reward transform	$0.04x + 2.8$
Policy class	Gaussian policy

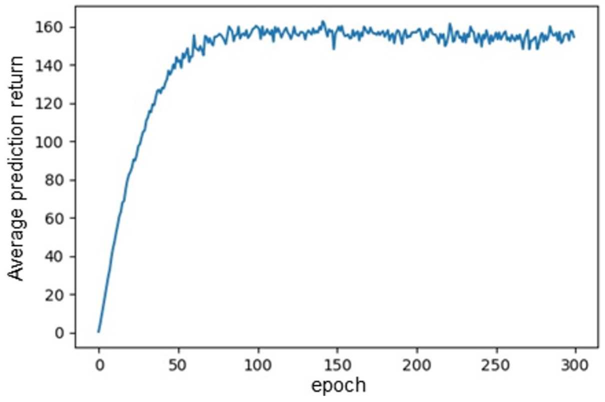


Fig. 9. Average prediction return.

4.4. Tracking control performance evaluation

We validated the trained policy network and our policy controller by conducting an experiment on CDPR shown in Fig. 2 and evaluated the tracking performance for the desired linear trajectory from (0 m, 0 m, 0 m) to (0.5 m, 1 m, 1 m) presented in Fig. 10(b) with red arrowed line. The policy operation when the policy is valid is depicted in Fig. 10(a). Simultaneously, we utilized two other controllers to compare the performance while moving on the same path; a conventional open-loop controller and a modified open-loop controller with the tension distribution form. Herein, we didn't use the external sensor (i.e., Vicon Camera) as the end-effector position feedback for control, then, a PID controller is limited to a cable-length control at each motor-winch side.

The PID control algorithm in the open-loop control employs the encoder value of the motor converted from cable length to adjust the position of the end-effector as

$$\boldsymbol{u}(k) = \boldsymbol{K_p e}(k) + \boldsymbol{K_i} \sum \boldsymbol{e}(k) + \boldsymbol{K_d \delta e}(k). \tag{18}$$

In addition, the modified open-loop controller with the tension distribution in (3)-(5) from the parallel robot's closed form is implemented to the conventional open-loop controller for the performance comparison.

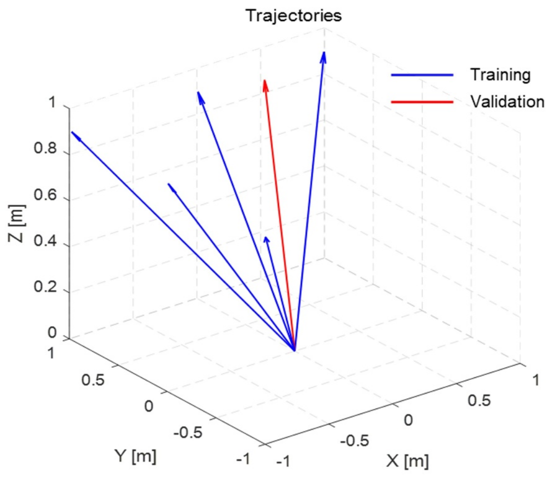
In the experiments, we computed position tracking errors and used them as a control performance comparison metric among three controllers; the open-loop controller, the modified open-loop controller, and the proposed DRL control. The results show that our policy controller performs the best results in all three directions as can be seen in Figs. 10 and 11.

Throughout the motion control, the tracking error of the proposed DRL method is significantly smaller in the x- and z-directions than that of the other two methods, and similar performance in the y-direction. The dynamic response of the DRL method achieved the best result, where the position error at the final point is 8.12 mm, 16.26 mm, and 12.33 mm for the proposed method, the open-loop control, and the modified open-loop controller, respectively.

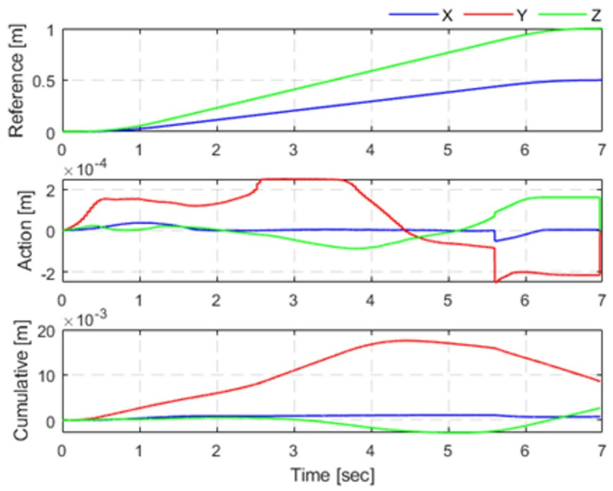
For the error of the end effector position, the DRL method performs even better. Compared with the open-loop method, the position error at the end position is reduced along the x-, y-, and z-directions by 33.3%, 54.2%, and 59%, respectively. Furthermore, compared with the open-loop controller with the tension form method, the position error is reduced to 20.7%, 35%, and 62.8% in each direction. The results can validate that the proposed method outperforms the others in both motion process and endpoint position error.

4.5. Position estimation performance as an alternative to forward kinematics

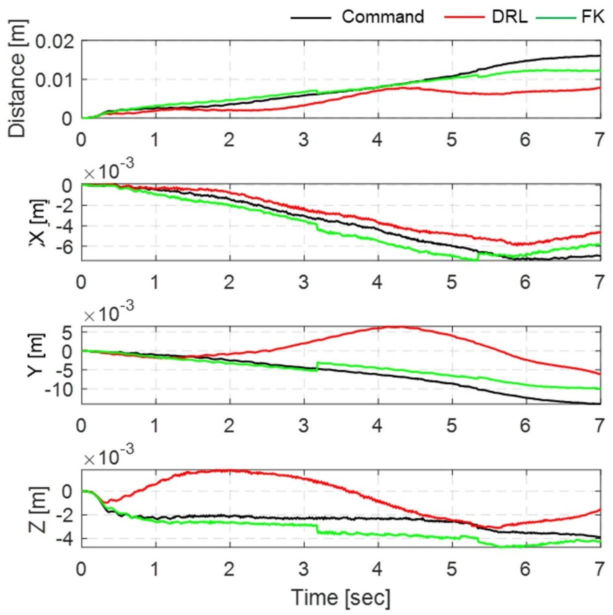
The estimated position error comparison of the command value of motors, DRL estimates position, and forward



(a) Training and testing trajectory in experiments.



(b) Policy operation.



(c) Tracking error comparison.

Fig. 10. Line trajectory experimental results.

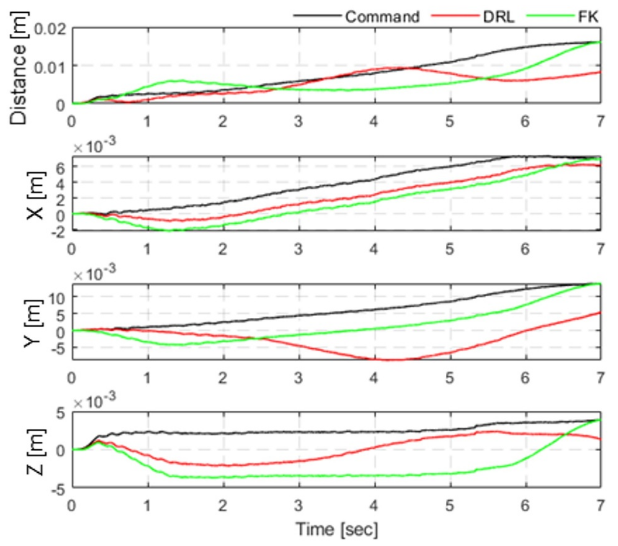


Fig. 11. End-effector position estimation performance comparisons in experiments; Command (Black line) is the reference, DRL (Red line) is the proposed method, and FK (Green line) is the conventional forward kinematics.

kinematic estimates position is depicted in Fig. 11.

Because of the expensive long-distance position and posture measuring sensor, we may not be able to obtain the location information of the robot end-effector directly, especially the large workspace application of a CDPR. We typically estimate the position of the end-effector based on the forward kinematic of the robot. However, the numerical solution of a CDPR's forward kinematic is more complex and not necessarily accurate [33]. We assume that the actions produced by the DRL network are partly a prediction of future moving errors, so we can use this information in conjunction with the reference trajectory to estimate the position of the end-effector [34]. We used the motion trajectories and experimental data from the experiments in the previous section to compare the errors of the estimated position of the end-effector.

The results indicate that the position esimtation error of the proposed DRL shows smaller than the conventional forward kinematics in the y- and z- directions and similar results in the x- direction. We confirmed that the proposed method is slightly better and comparable to the conventionally used forward kinematics when estimating the end-effector position.

5. CONCLUSION

In this study, a compensatory motion control method based on DRL is proposed and its feasibility is verified by experiments. To address the practical implementation; 1) The problem of asynchronous state observation and action execution, we put delaying the action execution time by one action applied cycle and add the action to be executed to the appropriate state. 2) The policy loop program is also programmed to delay to an appropriate time decided by the measured time to match the motion control program in TwinCAT3 caused by the difference between policy loop time and motion control loop time. 3) The reference trajectory of the first second of the robot's motion is used to traverse all the time points near the approximate starting point calculated by the threshold to obtain the corresponding camera-recorded start time of the robot movement. 4) Finally, the policy trained with the measured motion data of the desired trajectory can reduce the tracking error on testing trajectories.

Under the condition of no available position feedback sensor, the proposed method shows a better performance than a conventional open-loop control and a modified open-loop control with tension distribution. Moreover, the end-effector position estimation capability of the proposed method could show improved performance than the conventional forward kinematic solution in terms of end-effector position estimation.

The simulation and the experimental results in this study could demonstrate the feasibility of the proposed DRL method to the CDPR application for position tracking control and as an alternative method of forward kinematics. However, it has limitations in that we could not test the proposed method in other various trajectories and different payload conditions. Also, for the more complex trajectory application, the learning should be enhanced by using more training datasets with a high computation capability GPU. Ultimately, we will validate the proposed method under multiple trajectories and varying payloads for practical implementation based on the proposed approach in this paper.

CONFLICT OF INTEREST

The authors declare that there is no competing financial interest or personal relationship that could have appeared to influence the work reported in this paper.

The guest editor authors this article in this special issue. We declare that the peer review process of this article is the same as the peer review process of the journal in general and the authors (guest editor) have not handled the peer review process.

REFERENCES

[1] S. Qian, B. Zi, W. W. Shang, and Q. S. Xu, "A review on cable-driven parallel robots," *Chinese Journal of Mechanical Engineering* (English Edition), vol. 31, no. 4. Springer Singapore, 2018.

[2] M. A. Khosravi and H. D. Taghirad, "Robust PID control of fully-constrained cable driven parallel robots," *Mechatronics*, vol. 24, no. 2, pp. 87-97, 2014

[3] M. C. Kim, H. Choi, J. Piao, E. S. Kim, J. O. Park, and C. S. Kim, "Remotely manipulated peg-in-hole task conducted by cable-driven parallel robots," *IEEE/ASME Transactions on Mechatronics*, 2022.

[4] J. Piao, M. C. Kim, E. S. Kim, and C. S. Kim, "A self-adaptive inertia hybrid control of a fully constrained cable-driven parallel robot," *IEEE Transactions on Automation Science and Engineering*, vol. PP, pp. 1-11, 2023, doi: 10.1109/TASE.2023.3249826.

[5] A. Pott, *Cable-driven Parallel Robots: Theory and Application*, vol. 120, 2018.

[6] K. J. Astrom and T. Hägglund, "Advanced PID control," *IEEE Control Systems*, vol. 26, no. 1, pp. 98-101, 2006.

[7] 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 Transactions on Robotics*, vol. 31, no. 6, pp. 1444-1457, 2015.

[8] X. Tang and Z. Shao, "Trajectory generation and tracking control of a multi-level hybrid support manipulator in FAST," *Mechatronics*, vol. 23, no. 8, pp. 1113-1122, 2013.

[9] R. Babaghasabha, M. A. Khosravi, and H. D. Taghirad, "Adaptive robust control of fully-constrained cable driven parallel robots," *Mechatronics*, vol. 25, pp. 27-36, 2015.

[10] X. Jin, J. Jung, S. Ko, E. Choi, J.-O. Park, and C.-S. Kim, "Geometric parameter calibration for a cable-driven parallel robot based on a single one-dimensional laser distance sensor measurement and experimental modeling," *Sensors*, vol. 18, no. 7, 2392, July 2018.

[11] Z. Zhang, G. Xie, Z. Shao, and C. Gosselin, "Kinematic calibration of cable-driven parallel robots considering the pulley kinematics," *Mechanism and Machine Theory*, vol. 169, 104648, 2022.

[12] V. L. Nguyen and R. J. Caverly, "Cable-driven parallel robot pose estimation using extended Kalman filtering with inertial payload measurements," *IEEE Robotics and Automation Lettersry*, vol. 6, no. 2, pp. 3615-3622, 2021.

[13] T. Dallej, M. Gouttefarde, N. Andreff, P. E. Hervé, and P. Martinet, "Modeling and vision-based control of large-dimension cable-driven parallel robots using a multiple-camera setup," *Mechatronics*, vol. 61, no. April, pp. 20-36, 2019.

[14] M. Tao, B. Feng, L. Li, and L. Li, "Forward kinematics solution of cable robot based on neural network and L-M algorithm," *Proc. of IAEAC 2021 - IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference*, vol. 2021, pp. 2519-2524, 2021.

[15] W. Li, M. Yue, J. Shangguan, and Y. Jin, "Navigation of mobile robots based on deep reinforcement learning: Reward function optimization and knowledge transfer," *International Journal of Control, Automation, and Systems*, vol. 21, no. 2, pp. 563-574, 2023, doi: 10.1007/s12555-021-0642-7.

[16] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuška, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291-1307, 2012.

[17] Y. P. Pane, S. P. Nageshrao, J. Kober, and R. Babuška, "Reinforcement learning based compensation methods for robot manipulators," *Engineering Applications of Artificial Intelligence*, vol. 78, pp. 236-247, 2019.

[18] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: Lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698-721, 2021.

[19] Y. Liu, Z. Chen, Y. Li, M. Lu, C. Chen, and X. Zhang, "Robot search path planning method based on prioritized deep reinforcement learning," *International Journal of Control, Automation, and Systems*, vol. 20, no. 8, pp. 2669-2680, 2022.

[20] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1-40, 2016.

[21] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," *Proc. of IEEE International Conference on Robotics and Automation*, pp. 3389-3396, 2017.

[22] C. H. Min and J. B. Song, "Hierarchical end-to-end control policy for multi-degree-of-freedom Manipulators," *International Journal of Control, Automation, and Systems*, vol. 20, no. 10, pp. 3296-3311, 2022.

[23] K. Dmitry, A. Iroan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," *Proc. of 2nd Conference on Robot Learning*, vol. 87, no. CoRL, pp. 651-673, 2018.

[24] X. Zhan, H. Xu, Y. Zhang, X. Zhu, H. Yin, and Y. Zheng, "DeepThermal: Combustion optimization for thermal power generating units using offline reinforcement learning," *Proc. of 36th AAAI Conference on Artificial Intelligence (AAAI 2022)*, vol. 36, pp. 4680-4688, 2022.

[25] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," arXiv:2005.01643, pp. 1-43, 2020.

[26] J. Piao, X. Jin, J. Jung, E. Choi, J. O. Park, and C. S. Kim, "Open-loop position control of a polymer cable-driven parallel robot via a viscoelastic cable model for high payload workspaces," *Advances in Mechanical Engineering*, vol. 9, no. 12, 1687814017737199, 2017.

[27] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit Q-learning," arXiv:2110.06169, pp. 1-13, 2021.

[28] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," *Proc. of 36th International Conference on Machine Learning (ICML 2019)*, vol. 2019-June, pp. 3599-3609, 2019.

[29] A. Kumar, J. Fu, G. Tucker, and S. Levine, "Stabilizing off-policy Q-learning via bootstrapping error reduction," *Advances in Neural Information Processing Systems*, vol. 32, no. NeurIPS, 2019.

[30] D. Lau, J. Eden, Y. Tan, and D. Oetomo, "CASPR: A comprehensive cable-robot analysis and simulation platform for the research of cable-driven parallel robots," *Proc. of IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-Novem, pp. 3004-3011, 2016.

[31] A. Pott, H. Mütherich, W. Kraus, V. Schmidt, P. Miermeister, T. Dietz, and A. Verl, "Cable-driven parallel robots for industrial applications: The IPAnema system family," *Proc. of 44th International Symposium on Robotics (ISR 2013)*, 2013.

[32] C. P. Robert, "Simulation of truncated normal variables," *Statistics and Computing*, vol. 5, no. 2, pp. 121-125, 1995.

[33] A. Ghasemi, M. Eghtesad, and M. Farid, "Neural network solution for forward kinematics problem of cable robots," *Journal of Intelligent & Robotic Systems*, vol. 60, no. 2, pp. 201-215, 2010.

[34] H. Chen, M. C. Kim, and C. S. Kim, "Compensated motion control and forward kinematics of a cable-driven parallel robot based on deep reinforcement learning," *Proc. of the 38th ICROS Annual Conference (ICROS 2023)*, pp. 230-231, 2023.

Huaishu Chen received his B.S. degree in mechanical engineering from the Department of Mechanical Engineering at Wen-Zhou University, Wenzhou, China in 2019 and an M.S. degree in mechanical engineering from Chonnam National University, Korea, in 2022. His research interests are cable robot and deep reinforcement learning.

Min-Cheol Kim received his B.S. degree in mechanical engineering from Chonnam National University, Korea, in 2015. He is now pursuing a Ph.D. degree in mechanical engineering at Chonnam National University, Korea. His research interests are in dynamics and control applications for cable-driven parallel robots in industrial field.

Yeongoh Ko received his B.S. degree from the School of Mechanical Engineering, Chonnam National University, in 2021 and now he is an M.S. candidate in the Department of Mechanical Engineering at Chonnam National University. His research interests are control and robotics.

Chang-Sei Kim Please see vol. 21, no. 3, p. 947, March, 2023 of this journal.