

Model-Free Dynamic Control of a 3-DoF Delta Parallel Robot for Pick-and-Place Application based on Deep Reinforcement Learning

1st Hasan Jalali

*Human and Robot
Interaction Laboratory
School of Electrical and
Computer Engineering
University of Tehran
Tehran, Iran
hasanjalali@ut.ac.ir*

2nd Saba Samadi

*Human and Robot
Interaction Laboratory
School of Electrical and
Computer Engineering
University of Tehran
Tehran, Iran
saba.samadi@ut.ac.ir*

3th Ahmad Kalhor

*Control and Intelligent Processing
Center of Excellence
School of Electrical and
Computer Engineering
University of Tehran
Tehran, Iran
akalhor@ut.ac.ir*

4th Mehdi Tale Masouleh

*Human and Robot
Interaction Laboratory
School of Electrical and
Computer Engineering
University of Tehran
Tehran, Iran
m.t.masouleh@ut.ac.ir*

Abstract—This article investigates the performance of the Delta robot using model-free controller. Since it is difficult to obtain an accurate dynamic model of the robot and identification methods are complex and challenging. Therefore, in this article, a new approach is presented which checks the performance of the Delta robot in pick-and-place operation using model-free dynamic controllers. For the purposes of this paper, Deep Reinforcement Learning (DRL) and also the Deep Deterministic Policy Gradient (DDPG) network is used. The DDPG algorithm is implemented for the first time on the Delta robot, and according to the obtained results, it reveals that it can provide a favorable approach for the pick-and-place operation. In this method, based on the science of DRL, a reward function based on the pick-and-place operation is also defined, which gives negative reward (punishment) or a positive reward according to the way the robot performs. It is noteworthy that according to the obtained results, it is possible to generalize other points for the pick-and-place operation. It should be noted that the robot was able to learn the target after a long episode and its graph converged after approximately 18,000 episodes.

Index Terms—Delta Parallel Robot, Deep Reinforcement Learning, Dynamic Control, Model-Free Control.

I. INTRODUCTION

A robot is a mechanical device, in other words, a mechanical machine with electrical processors that can easily perform tasks which are complicated for humans by using the programs it receives. Robots can be divided into two general categories: series and parallel [1]. Each series and the parallel robot has unique features based on human needs, and both of them are commonly used in industry [2]. It can be said that parallel robots are among the most common manipulators, which are also used as pick-and-place robots and multi-axis machining in factories. Among the parallel robots, it can be mentioned the Delta robot, which is one of the most popular and widely used robots among industrialists and researchers [3].

Now a day, knowledge and skills such as Artificial Intelligence (AI) [4], machine learning [5], deep learning [6], and Reinforcement Learning (RL) [7] have a huge contribution to

the progress of science. AI is a system or a machine which, according to the commands and information it receives, tries to imitate humans and behave like them, and in this way improve its performance [8]. Deep learning is a subset of machine learning which works like a neural network. These neural networks try to simulate and learn the behavior of the human brain using the data they receive [9]. RL, which stands for Reinforcement Learning, is another field of machine learning which examines how intelligent agents work to maximize rewards; which will be discussed further [10], [11].

Delta robot was concocted within the early 1980s by the group of Professor Reymond Clavel. One of the prominent features of this robot is its high speed and accuracy in moving objects, especially light objects. For this reason, this robot is widely used in food and pharmaceutical packaging factories. Among other applications of the Delta robot, it can be mention the operation of filling bottles with liquids, identifying defective parts, sticking labels on packaging, and also moving objects from one place to another (pick-and-place) [12].

It is worth mentioning that the linear form of the dynamic equation of parallel robots can be used to implement non-linear adaptive control schemes. In parallel manipulators, the linearization of the desired dynamic equations is a useful task, but it is also time-consuming and complex. Therefore, with the help of virtual methods, a linear form of the dynamic equations of the Delta parallel robot has been calculated. For this reason, the screw theory is used for kinematic analysis [13]. By obtaining the input-output Jacobian matrices and the Jacobian matrices between each link and the kinematic parameters of the end-effector, the dynamic equation of the robot's movement is obtained.

The inverse dynamic controller (IDC) of the self-adjusting online neural network can be used to control the high-speed and smooth path tracking of the Delta robot. In this way, with this approach, a reasonable controller can be given for a large

number of distinct non-linear ways additionally diminish end-effector fluctuations at high speed. One of the drawbacks of a speed-based controller is the failure to track highly dynamic ways, for this reason, an IDC which can perform quick maneuvers with accuracy is utilized to decrease these defects [14]. Regarding the issue of recognizing the actuator of a 3-DoF Delta parallel robot, the linear and nonlinear identification methods can be used, which within the context of motion controllers for robotic manipulators play a critical role. In general, it can be said that for this problem, the prediction error method is used to develop a model that can identify a design for the Servo motor-Servo drive enclosure [15].

As aforementioned, RL is a science in which the Agent learns by collaboration with the environment and observing the results it gets, such as rewards and punishments. In fact, way of RL is similar to the learning and training of children and animals. For this reason, reinforcement learning is used today to solve various problems whose common solutions are somewhat difficult [16]. In reinforcement learning, it is possible to investigate the desired problem without needing to fully understand the robot or the intended subject by using the positive and negative rewards that are obtained [17]. Among the topics which can be investigated with RL science, two categories of gaming [18] (choosing the best move in the game according to the amount of reward it receives) and control [19] (controlling robots without having detailed information about them) can be discussed. One of the advantages of using RL in the field of control is that the robot can adapt quickly to environmental changes through retraining and improving its performance. In this way, it chooses the most optimal way by using the amount of reward and punishment it gets [20].

In this article, the Delta robot is investigated using a model-free dynamic controller. According to the obtained results and to the best knowledge of authors, it is worth noting that this approach has been implemented for the first time on the Delta robot, while in the previous methods [14], controllers were used that need to know the dynamic model of the robot, which in this way causes complexity in the process. For this purpose, the reward function based on the pick-and-place operation has been introduced that provided convergence for the training of the Deep Reinforcement Learning (DRL) algorithm. In addition to this function, the Deep Deterministic Policy Gradient (DDPG) algorithm is also used, which was implemented on the Delta robot for the first time.

In this paper, the proposed divisions will be as follows. At first, a general explanation was given about parallel robots and optimization methods used in the industry. In Section II, the complete Delta robot will be examined and Section III will explain the implemented approach, the DRL method and the DDPG algorithm. Section IV presents the results of the implemented methods, and finally, in Section V, which is the final part, by using the obtained results, conclusion about the work done is discussed, as well as the effect of this method in the scope of work.

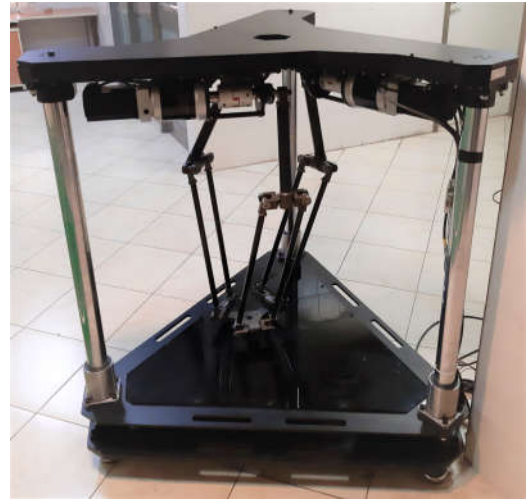


Fig. 1: The 3-DoF Delta Parallel robot built at the Human-Robot Interaction Laboratory, University of Tehran.



Fig. 2: Delta robot structure in Simscape.

II. 3-DOF DELTA PARALLEL ROBOT

As can be seen from Fig. 1, a view of the Delta robot structure is given. The Delta robot is a robot with three arms which are connected by a universal joint at the base and is placed in the category of parallel robots with 3 degrees of freedom. In order to control a 3-DOF delta parallel robots with high accuracy for different planning paths at high speed and high acceleration, it is necessary to identify the parametric model of the robot and motor in order to achieve an accurate control model. The proposed strategies are executed on a 3-DoF Delta parallel robot in Simscape, shown in Fig. 2. This robotic manipulator is composed of three identical limbs, each making a closed kinematic chain. The upper arm is connected to an actuated joint and the lower arm is linked to a parallelogram which comes to the end-effector [21].

As mentioned earlier, the Delta robot is a robot which by receiving the relevant themes, can pick up an object and put it at a specific point, as well as arrange the products in their

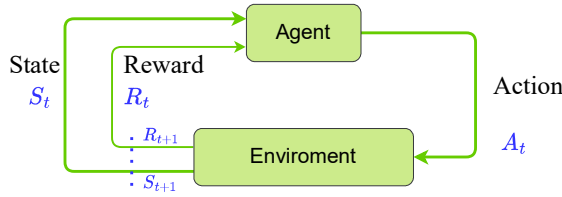


Fig. 3: Reinforcement Learning structure.

special places at a high speed. Therefore, nowadays this robot is widely used in industrial factories such as food industry, pharmaceutical industry, etc.

III. DRL AND DDPG

DRL and DDPG network methods are used in this article. In this section, first, a brief explanation of RL and DRL will be given. Then, the DDPG network will be presented. In the following, this algorithm is used for model-free control based on dynamics, the purpose of which is to learn the robot in the pick-and-place operation.

A. DRL

DRL is a sub-branch of ML science, which is the result of the combination of two sciences, deep learning, and RL. RL as a branch of ML is one of the most discussed and hot topics in the field of machine learning and intelligent systems. RL structure is shown in Fig. 3; RL tries to go through a process similar to the learning process of the human brain to reach a desirable and reliable model. This alludes to the thought that people continually take data from the environment and take fundamental activities to adjust to the environment. The process of teaching models based on RL is based on reward and punishment. In this way, whenever the agent shows appropriate and desirable performance by receiving rewards and whenever it performs an inappropriate action by receiving punishment, learns what kind of performance to show in different environmental conditions and at special times. In other words, the agent carries on like a human. The agent gets the prize in persistent interaction with the environment. The remunerate acts as input data to assess the agent's execution and makes a difference to the agent to select activities towards the most prominent aggregate reward value and the agent, at last, finds the optimal technique. DRL algorithms can have very large inputs and perform actions which optimize the desired goal. RL is based on the Markov decision prepare framework and is as a rule spoken to by the following five parameters:

$$(S, A, R, P, \gamma) \quad (1)$$

where S and A speak to the state space and action space which an agent can perform, R defines the reward space, P means the state transition probability, γ is a parameter that calculates the aggregate reward function, and finally specifies the policy π that shows the mapping from the state space to the action space. The agent in the situation and at time t is $S_t \in S$ and the selected action based on the existing policy at time t is $A_t \in A$. Due to

acting at the same moment, it receives a reward which is R_t , and after that, the status of the agent is transferred from S_t to S_{t+1} . The entire reward is represented by G_t , and the prompt compensate of each stage is spoken to by the discount factor γ , which is a number between zero and one added and its formula is as follows:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (2)$$

In order to parameterize the effects of state and activity on the whole remuneration, a state-action value function is developed. Therefore, the expected total reward under policy π is defined as follows:

$$Q_{\pi}(s, a) = E_{\pi} [G_t | S_t = s, A_t = a] \quad (3)$$

The objective of RL is to discover the ideal approach among all policies, such that the entire desire of the reward under the optimum procedure π^* is maximum, which means that the state-action value function corresponding to the optimal arrangement π^* has the largest state-action value function between all policies:

$$Q^*(s, a) = \max_{\pi} Q_{\pi}(s, a) \quad (4)$$

Some RL methods are based on the Q table, which forms the agent's central functional core in the environment. This method is also called the method based on policy or strategy. If the agent takes the decision-making policy well and completely, it can decide what is the best practice in different situations and take action to adopt it. Now that the RL based on the decision table has been explained, it is possible to explain the DRL. In DRL, instead of using a Q table, a multi-layer deep neural network is utilized to demonstrate value functions and policies. Considering the advantages of DRL, in this article, this method is used together with the DDPG algorithm. It is also worth mentioning that this method has provided better results compared to other methods published in the past. In the next part, more about DDPG algorithm will be explained.

B. DDPG Algorithm

DDPG which is actually an algorithm which simultaneously learns a Q-function and a policy. DDPG is a model-free out-policy algorithm for continuous action learning that combines DPG (Deterministic Policy Gradient) and DQN (Deep Q-Network) methods. Regarding the DDPG network, it can be said that it is an algorithm outside of politics which can only be used for environments with continuous action intervals. This network can be considered as a deep Q-learning for continuous action spaces.

The assignment of reaching the required position within the three-dimensional space by the end-effector of the three-degree-of-freedom robot Delta has a continuous state space. Hence, the DDPG algorithm is utilized in this investigation. This algorithm can be connected in continual state space. The calculation receives a deterministic manner procedure and gets the specified action value via the respective function. Using the Actor-Critic framework, A deterministic strategy $\xi(s | \theta^{\xi})$ is

demonstrated using the actor network with parameters θ^ξ and the state value function is demonstrated using the critic network with parameters $Q(s, a | \theta^Q)$ [22].

Actor networks are the same as neural systems. To train these networks, independent and uniformly distributed input data are required, however, the reinforcement learning model is a Markov decision process and the data is collected consecutively, not meeting the stipulations of independence and uniform dissemination. Hence, the involvement of DQN data collection is utilized to break the data correlation, encourage progressing the convergence and stability of the algorithm.

For the critic, the objective of the assessment network is to minimize the value of the cost function $L(\theta^Q)$:

$$L(\theta^Q) = E_{s_t, a_t, r_t, a_{t+1}} [(r(s_t, a_t) + \gamma Q'(s_{t+1}, \xi(s_{t+1}) | \theta^Q) - Q(s_t, a_t | \theta^Q))^2] \quad (5)$$

For the actor, the evaluation network updates the parameters according to the following formula [23]:

$$\nabla_{\theta} J_{\beta}(\theta^\xi) = E_{s \sim \rho^\beta} [\nabla_a Q(s, a | \theta^Q) |_{a=\xi(s)} \nabla_{\theta^\xi} \xi(s | \theta^\xi)] \quad (6)$$

During the training process of the algorithm, the parameters of the actor-critic network are updated by the Soft-Update algorithm, whose formula is as follows:

$$\begin{cases} \theta^{\xi'} = \alpha \theta^\xi + (1 - \alpha) \theta^{\xi'} \\ \theta^{Q'} = \alpha \theta^Q + (1 - \alpha) \theta^{Q'} \end{cases} \quad (7)$$

where α is the update variable.

IV. IMPLEMENTED ALGORITHM AND RESULTS

A. Model-free control of Delta robot

This section examines the performance of the DDPG model-free dynamic controller for the 3-DOF Delta robot to perform a pick-and-place application. At first, the considered state and action are determined. During the algorithm training process, DDPG data related to the state includes the angle, speed, and acceleration of the joints, also the speed and position of the effector are considered as the state, in addition to the previously selected actions as a mode is considered. Because the goal is the dynamic control of the delta robot, three torques of each joint have been selected as action, which is well shown in Fig. 4.

It should be noted that the operation examined in this article is pick and place. Considering that the state space is such that at the end of the training, the actions are selected in such a way that they reach the desired point. The need to increase the target is created as a state-by-state vector. In other words, for example, if the robot is placed at a certain angle, it only knows that it should go towards the desired goal, and it does not know whether it is putting or picking up. For this reason, another state will be added to the state vector, which will specify the state of the robot in being placed or taken. The new state vector changes as follows:

$$S = g_1, g_2, \theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \ddot{\theta}_1, \ddot{\theta}_2, \ddot{\theta}_3, X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}, A, t-1$$

$$A = \tau_1, \tau_2, \tau_3$$

Matlab environment is used to implement this algorithm. In the algorithm DDPG, the reward function is a numerical function and evaluates whether the change of joint angles is in accordance with the work requirements or not. If it fulfills the conditions of the task, the agent will receive the reward form, and otherwise, it will receive the punishment reward form. The reward function can be expressed mathematically as follows:

$$R = \begin{cases} R_1, & \text{if } t < T_{PP} \\ R_2, & \text{if } t > T_{PP} \end{cases} \quad (8)$$

where T_{PP} is the time when the robot position changes from pick mode to place mode. R_1 and R_2 are defined as follows:

$$R_1 = -0.1 \times (\theta_i - \theta_{d_{pick}})^2 + r(\theta_{d_{pick}}, \theta_i) - 0.05 \times \tau_i^\top + 250 \times \frac{T_s}{T_f} \quad (9)$$

$$R_2 = -0.1 \times (\theta_i - \theta_{d_{place}})^2 + r(\theta_{d_{place}}, \theta_i) - 0.05 \times \tau_i^\top + 250 \times \frac{T_s}{T_f} \quad (10)$$

where T_s and T_f are defined in TABLE I and $\theta_{d_{pick}}$ and $\theta_{d_{place}}$ are the desired angles of pick-and-place operation and the r function is defined as follows:

$$r(\theta_g, \theta_i) = \begin{cases} \frac{100}{|\theta_i - \theta_g| + 0.5}, & \text{if } 1(\text{deg}) < (\theta_i - \theta_g) < 3(\text{deg}) \\ 10000, & \text{if } (\theta_i - \theta_g) \leq 1(\text{deg}) \end{cases} \quad (11)$$

Another important point is that after the passage of time, T_{PP} target states change as follows:

$$(g1, g2) = \begin{cases} (0, 1), & \text{if } t < T_{PP} \\ (1, 0), & \text{if } t > T_{PP} \end{cases} \quad (12)$$

The reason for this is that after passing through it, it will change to the target time and perform the action modes which reach the harvest point. The equivalent of the reward function is considered in such a way that the end-effector of the robot reaches the primary point or goal in the shortest possible time before reaching the T_{PP} time and keeps the final performer there, which is seen in the r function. And after passing the T_{PP} time, the robot will bring its final executioner to the second target or harvest point. Also, in the reward function, there are two parts τ and $\frac{T_s}{T_f}$, respectively, the first part is to reduce and shrink the control signal, and the second part receives points in exchange for the robot not leaving the workspace. The considered parameters are as follows:

$$\theta_{d_{pick}} = (71, 29, 49)(\text{deg})$$

$$\theta_{d_{place}} = (30, 30, 30)(\text{deg})$$

$$T_{PP} = 2.5s$$

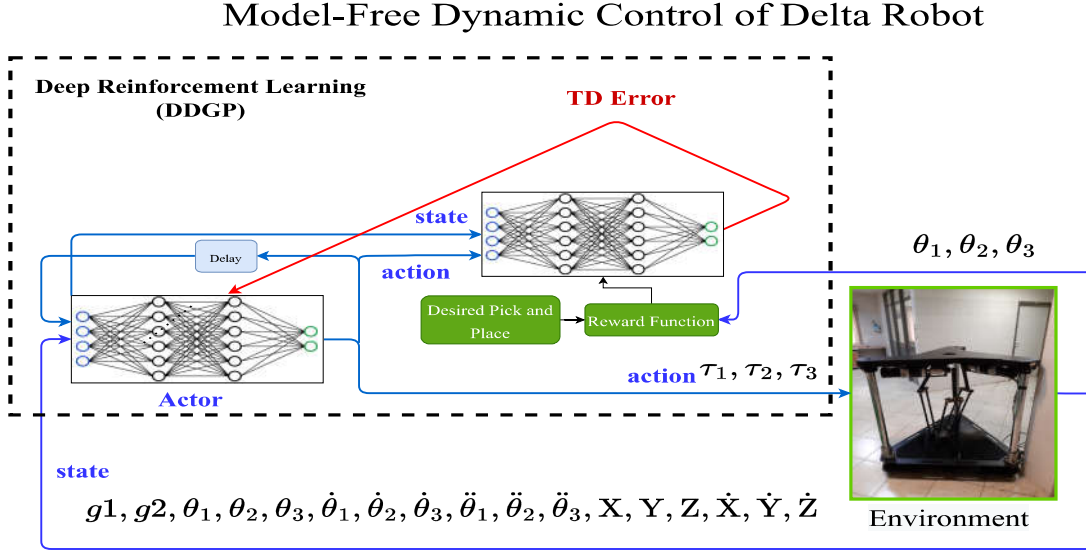


Fig. 4: Structure of model-free dynamic control of Delta robot based on DDPG.

Also, considering that the intended task for learning is episodic, it means that this problem has a starting point and an end point. A function has been considered for the completion of the event in MATLAB, which means that if the robot leaves the workspace, the event will end, and if not, it will be the end time of the simulation. Finally, the structure of actor and critic neural networks and parameters considered for training are shown in TABLE I. Sampling time is the time when the actions considered by the robot are entered and the amount of changes in each of the observations is calculated it enters the reward loop and the reward is calculated for that time. Episode Time is the time when an event or period ends and the reward for the entire event is calculated and the actions and observations of the robot are completely erased and the robot starts the next event from its starting point. The discount factor determines how much importance the DRL factor gives to rewards in the distant future compared to rewards in the present moment. If this value is equal to zero, it means that the agent is completely myopic and seeks to obtain the maximum reward in the present moment. On the other hand, if this value is equal to one, it means that the agent seeks to maximize the rewards received in the future. Also, noise with the mean and variance stated in the table was added to the intelligent agent so that the intelligent agent can explore more and gain more experience to achieve more accurate control. The total number of courses for intelligent agent training is 20,000. It is important to mention that due to restrictions applied during robot control, an episode may end early and the entire period or event may be restarted.

Considering that the output of the actor's neural network is between -1 and 1, the applied torque is nine times, and its value is between -9 and 9.

TABLE I: Agent training parameters

Parameter	Value
SampleTime(T_s)	0.01
EpisodeTime(T_f)	5
DiscountFactor	0.99
MiniBatchSize	128
ExperienceBufferLength	500000
TargetSmoothFactor	0.001
NoiseOptionsofMeanAttractionConstant	5
NoiseOptionsofVariance	0.5
NoiseOptionsofVarianceDecayRate	0.00001
MaxEpisodes	20000

The results obtained from learning the DDPG network are shown in Fig. 5. These results are based on the average reward and reward in each step according to the episode and steps. For the considered values after training, the state or angle of actuators and action are shown in Fig. 7 and Fig. 6.

By investigate the result, the robot has been able to keep the angles of the arms around the desired angles and perform the pick-and-place operation well. But the noteworthy point is that the torque changes are extremely large and it can be said that the implementation of this actuator is very difficult in practice and future research can make torque signals implementable by changing the reward function.

V. CONCLUSION

In this study, the performance of the Delta robot using model-free controllers for the pick-and-place operation has been investigated. In this method, DRL and DDPG network methods have been used, in which desired states and actions have

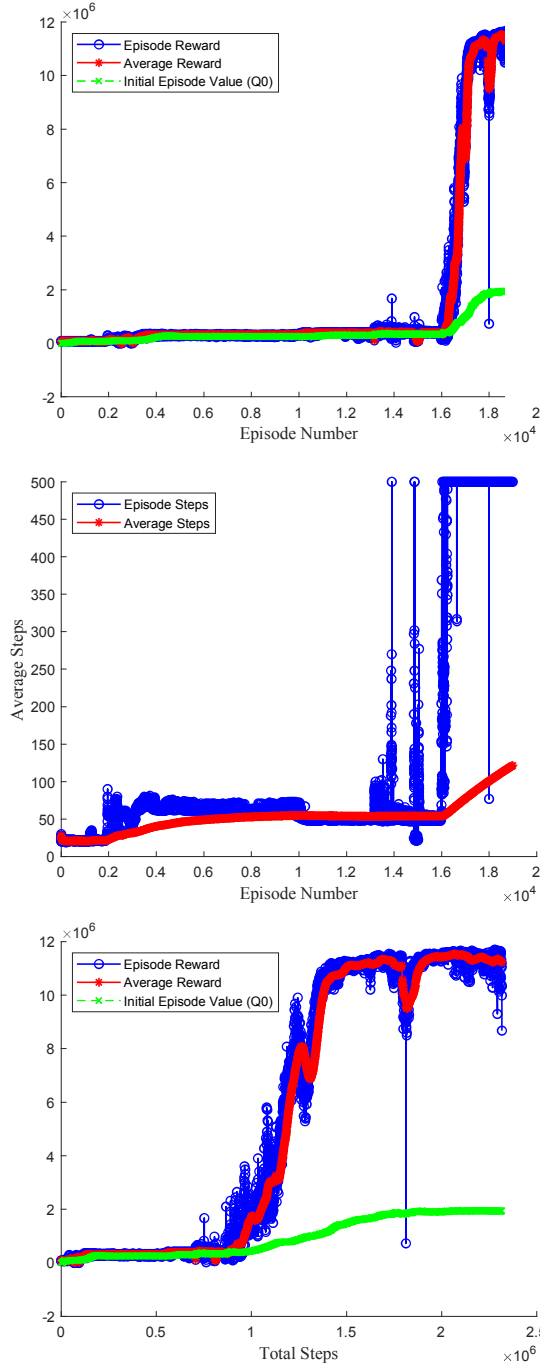


Fig. 5: Average reward and average step result of the DDPG for pick and place.

been introduced. Also, in this approach, the reward function based on the pick-and-place operation is defined that provided convergence for the training of the DDPG algorithm and improves the performance of the Delta robot by positive and negative rewards. Finally, after a significantly long episode, the under study robot was able to learn the desired goal with a small amount of error compared to other common methods

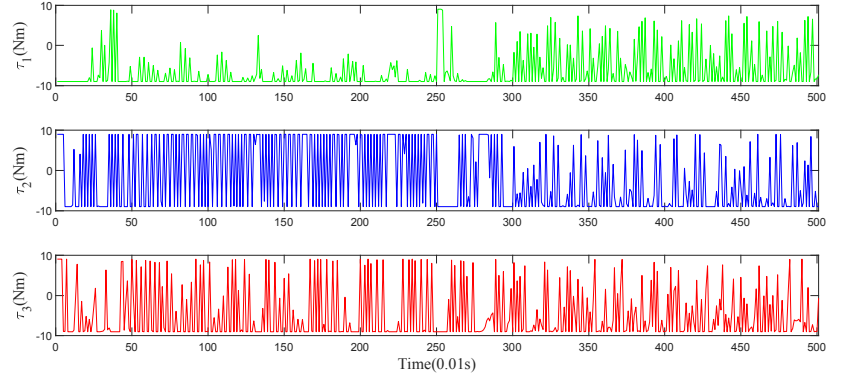


Fig. 6: Action result after training the model.

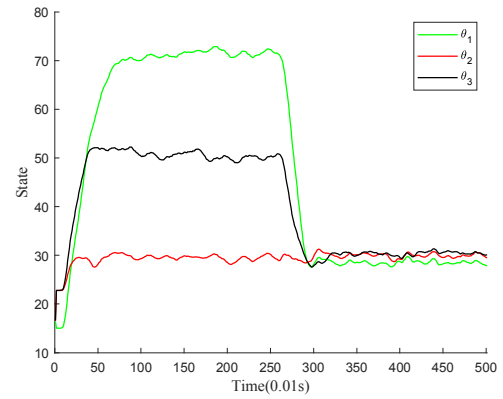


Fig. 7: Angle of actuator after training.

and its graph converges after approximately 18,000 episodes. It should be mentioned that the value of its torques was such that it made it impossible to apply them to the real system. As ongoing work, this approach can be extended by using DRL and networks other than the DDPG algorithm, as well as changing the mentioned reward function.

REFERENCES

- [1] M. A. Khosravi and H. D. Taghirad, "Dynamic modeling and control of parallel robots with elastic cables: Singular perturbation approach," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 694–704, 2014.
- [2] S. Briot and I. A. Bonev, "Are parallel robots more accurate than serial robots?" *Transactions of the Canadian Society for Mechanical Engineering*, vol. 31, no. 4, pp. 445–455, 2007. [Online]. Available: <https://doi.org/10.1139/tcsme-2007-0032>
- [3] H. D. Taghirad, *Parallel robots: mechanics and control*. CRC press, 2013.
- [4] J. Liu, X. Kong, F. Xia, X. Bai, L. Wang, Q. Qing, and I. Lee, "Artificial intelligence in the 21st century," *IEEE Access*, vol. 6, pp. 34 403–34 421, 2018.
- [5] S. Angra and S. Ahuja, "Machine learning and its applications: A review," in *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*, 2017, pp. 57–60.
- [6] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2094–2107, 2014.

- [7] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.
- [8] L. Chen, P. Chen, and Z. Lin, "Artificial intelligence in education: A review," *IEEE Access*, vol. 8, pp. 75 264–75 278, 2020.
- [9] F. Q. Lauzon, "An introduction to deep learning," in *2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*, 2012, pp. 1438–1439.
- [10] M. Fang, Y. Li, and T. Cohn, "Learning how to active learn: A deep reinforcement learning approach," *arXiv preprint arXiv:1708.02383*, 2017.
- [11] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.
- [12] I. Bonev, "Delta parallel robot-the story of success," *Newsletter*, available at <http://www.parallelmic.org>, 2001.
- [13] F. A. Azad, M. Reza Hairi Yazdi, and M. T. Masouleh, "Kinematic and dynamic analysis of 3-dof delta parallel robot based on the screw theory and principle of virtual work," in *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, 2019, pp. 717–724.
- [14] S. Rahimi, H. Jalali, M. Hairi Yazdi, A. Kalhor, and M. Masouleh, "Design and practical implementation of a neural network self-tuned inverse dynamic controller for a 3-dof delta parallel robot based on arc length function for smooth trajectory tracking," *Mechatronics*, vol. 84, p. 102772, 06 2022.
- [15] S. Rahimi, H. Jalali, M. R. Hairi Yazdi, A. Kalhor, and M. T. Masouleh, "Experimental study on neural network-arx and armax actuation identification of a 3-dof delta parallel robot for accurate motion controller design," in *2021 9th RSI International Conference on Robotics and Mechatronics (ICRoM)*, 2021, pp. 399–406.
- [16] M. H. Olyaei, H. Jalali, A. Olyaei, and A. Noori, "Implement deep sarsa in grid world with changing obstacles and testing against new environment," in *Fundamental Research in Electrical Engineering*. Springer, 2019, pp. 267–279.
- [17] P. Dayan and B. W. Balleine, "Reward, motivation, and reinforcement learning," *Neuron*, vol. 36, no. 2, pp. 285–298, 2002. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0896627302009637>
- [18] M. A. Samsuden, N. M. Diah, and N. A. Rahman, "A review paper on implementing reinforcement learning technique in optimising games performance," in *2019 IEEE 9th International Conference on System Engineering and Technology (ICSET)*, 2019, pp. 258–263.
- [19] Z. Li, J. Liu, Z. Huang, Y. Peng, H. Pu, and L. Ding, "Adaptive impedance control of human–robot cooperation using reinforcement learning," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 10, pp. 8013–8022, 2017.
- [20] D. Lee, N. He, P. Kamalaruban, and V. Cevher, "Optimization for reinforcement learning: From a single agent to cooperative agents," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 123–135, 2020.
- [21] S.-D. Stan, M. Manic, C. Szep, and R. Balan, "Performance analysis of 3 dof delta parallel robot," in *2011 4th International Conference on Human System Interactions, HSI 2011*, 2011, pp. 215–220.
- [22] R. Zeng, M. Liu, J. Zhang, X. Li, Q. Zhou, and Y. Jiang, "Manipulator control method based on deep reinforcement learning," in *2020 Chinese Control And Decision Conference (CCDC)*, 2020, pp. 415–420.
- [23] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.