Collaborative Multi-Agent Reinforcement Learning Control of Parallel Robots

1st Amirhossein Afkhami Ardekani
Human and Robot Interaction Laboratory
School of Mechanical Engineering
University of Tehran
Tehran, Iran
amirh.afkhami@ut.ac.ir

2nd 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

3rd Mohammad Reza Hairi Yazdi
School of Mechanical Engineering
College of Engineering
University of Tehran
Tehran, Iran
myazdi@ut.ac.ir

2022 10th RSI International Conference on Robotics and Mechatronics (ICRoM) | 978-1-6654-5452-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRoM57054.2022.10025233

***Abstract*—In this paper, a novel algorithm based on Multi-Agent Reinforcement Learning for controlling parallel robots has been suggested. The dynamic models of parallel robots are complex and full of uncertainties, and deriving them requires deep knowledge of the mechanism of the robot. Therefore, the proposed algorithm is designed model-free to be independent of prior knowledge about the system from the outset. Moreover, this algorithm comprises two primary components, making it efficient in training and convergence. The proposed algorithm takes each loop or limb in parallel robots as a separate agent. These agents then learn to collaborate to fulfill the robot's defined task by producing appropriate control signals from a decentralized point of view. For studying the performance of the proposed algorithm, a 3-DOF parallel robot called Agile Eye is taken into account as a case study which is simulated in CoppeliaSim simulation environment for the task of reference tracking. Two other controllers, including the classic Proportional Integral Derivative (PID) controller and the single-agent counterpart of the suggested algorithm, have been implemented for a better performance comparison of the proposed algorithm. Using the Root Mean Square Error (RMSE) index, the recommended algorithm with an RMSE value of 0.0553 is superior to its single-agent counterpart with an RMSE of 0.1105. On the other hand, the proposed algorithm is inferior to the PID controller with an RMSE of 0.0275, mainly due to the fact that the PID Controller is in velocity control mode, while the proposed algorithm manipulates the robot in torque control mode, which is less stable.**

***Index Terms*—Reinforcement Learning, Model-free, Multi-agent systems, Collaborative systems, Parallel robots, PID controller, Torque control**

I. Introduction

In recent years, the great advent made in artificial intelligence and machine learning approaches has paved the way to extend such techniques in analyzing robotic mechanical systems. From both research, industrial and commercial points of view, robotic manipulators are the most known robots with a wide range of applications. From a structural perspective, robots fall into two categories, namely serial robots, and parallel robots. The location of the actuators, the available space, and the arrangement of the arms may be seen as the main distinctions between these two designs. Actuators in serial manipulators are placed on the joints; in contrast, parallel robots can be designed in such a way as to have fixed-frame actuators.

Different methodologies are needed to address the kinematics and dynamics of these manipulators since the parallel robots own many closed kinematic chains as opposed to the serial kind, which consists of an open kinematic chain, due to their mechanism structure. Parallel manipulators have garnered considerably more interest in recent years due to their wide variety of applications. The most well-known parallel mechanisms include the Gough-Stewart platform [1], the Delta robots [2], cable-driven robots [3], the Tripteron [4], and the Agile Eye [5]. Different approaches to figuring out the governing equations of parallel robots have been suggested in the literature. For example, in recent years, analytical techniques for the kinematics of the Delta robots have been presented in [6]-[8]. Also, in [9]-[11], the Agile eye parallel robot's dynamic and kinematic equations were tried to be derived. In [12], a 3-DOF spherical parallel robot mimicked the human head movement using a PID controller. In addition, based on the robot's dynamic model, the control of a suspended cable-driven parallel robot has been experimentally examined for object-tracking purposes in [13]. Moreover, a 3-DOF Delta robot is controlled by an online neural network self-tuned inverse dynamic controller at a fast speed and with excellent smoothness in [14]. The major challenge with parallel manipulators is the intricacy of the mathematical model due to the complex structure of these robots. The correctness and reliability of the model are crucial in this kind of robots because of how complicated the equations are.

Various controllers have recently been developed in [15] and [16] to control parallel manipulators. The fundamental studies of parallel robots provide problems regarding their kinematic and dynamic analysis. Due to the parallel robots' completely linked dynamics and nonlinear kinematics, standard controllers are unable to control these types of robots effectively. Moreover, because of the unmodeled dynamics and imprecise system characteristics of parallel robots, kinematic model-based approaches are not accurate. Furthermore, dynamic model-based solutions are difficult to apply in real-

time and take a lot of time, preventing them from producing precise and adaptive performances. The existing uncertainty in dynamic model equations also makes controlling such robots using classic controllers a challenging task. For that reason, using artificial intelligence-based controlling methods can be regarded as an appropriate alternative for controlling these dynamically sophisticated robots. These methods can not only compensate for the uncertainties of the system but also can reach a high level of accuracy in the control of parallel robots even without having any knowledge about the governing equations of the system. That is the main advantage of intelligent methods like Reinforcement Learning (RL), where an accurate controller can be designed using such methods without having prior knowledge about the system. In [17], the required force for controlling a manipulator has been computed using RL, and then the desired position has been achieved using PID admittance control. Also, RL has been used as a complement to classic control methods for compensating uncertainties such as frictions and contacts in the dynamic system [18]. There are various state of the art RL- based feedback control techniques that use optimal control as principles for controlling and monitoring single-agent and multi-agent systems [19]. In another approach, RL is used for controlling an industrial robot with six axes for complicated motion planning on continuous trajectories [20].

The main contribution of this paper consists in proposing a novel algorithm based on collaborative Multi-Agent Reinforcement Learning (MARL) for controlling parallel robots. Since parallel robots have multiple dynamic loops in their structure, in this algorithm each of these loops has been considered as a separate agent, which then tries to cooperate with each other to fulfill the defined task for the robot regarding the essence of the problem. Parallel robots are complicated in terms of equations of motion. For that reason, the suggested algorithm is model-free, so there is no need to have knowledge about the dynamics and governing equations of these robots. Another feature of the designed algorithm is that it can deal with dynamic environments and reach a generality in such environments, which is a prevalent setup in robotics.

The remaining sections of the paper have been set up as follows. In Section II, some fundamental knowledge about RL and collaborative MARL has been provided. Examining the proposed method based on MARL for controlling parallel robots and different components of the suggested algorithm is the focus of Section III. The proposed algorithm's convergence and performance during training have been investigated first in Section IV. For that, the designed algorithm has been simulated on a parallel robot known as the 3-DOF Agile Eye robot, which is considered as a robot with sophisticated dynamics; this robot is illustrated in Fig. 1. The algorithm was implemented for the task of End-Effector (EE) reference tracking. Finally, in this section, the performance of the suggested algorithm has been compared with two other controllers using an introduced performance index.

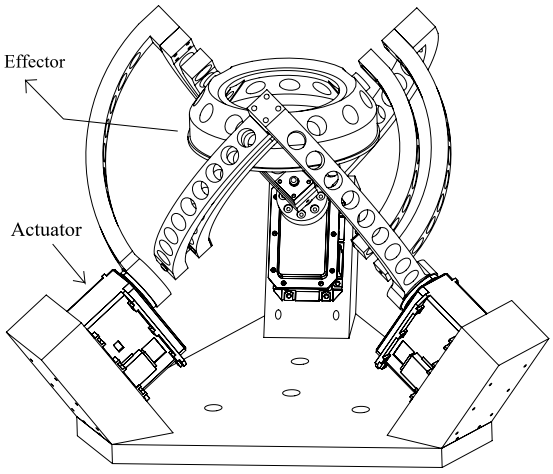

Fig. 1. The case study 3-DOF Agile Eye parallel robot with three limbs.

II. Reinforcement Learning

Reinforcement Learning's primary idea is based on learning through interaction with the environment. An RL agent engages with its surroundings and, after experiencing the results of its actions, can evolve to modify its behavior in response to the returned reward from the environment [21]. In the RL setup, an agent which is guided by a machine learning algorithm, observes a state s_t from its surroundings at time step t. The agent at state s_t interacts with the environment by taking an action a_t and then due to the selected action, the agent and environment enter a new state, s_{t+1}. The rewards which the environment offers define the optimum course of actions. Every time the environment changes states, it also gives the agent feedback in the form of a scalar reward, r_{t+1}. The standard RL objective is the expected sum of rewards and can be represented by following formula:

$$J = \sum_{t} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r\left(\mathbf{s}_t, \mathbf{a}_t\right) \right] \tag{1}$$

The agent's objective is to discover a policy $\pi\left(\mathbf{a}_t \mid \mathbf{s}_t\right)$ which maximizes the discounted reward. A policy then provides an action to take in response to a given state. In Maximum Entropy Reinforcement Learning [22], the standard objective is generalized by adding an entropy term, making the optimum policy also attempt to maximize the entropy during training:

$$\pi^* = \arg\max_{\pi} \sum_{t} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r\left(\mathbf{s}_t, \mathbf{a}_t\right) + \alpha \mathcal{H}\left(\pi\left(\cdot \mid \mathbf{s}_t\right)\right) \right] \tag{2}$$

In the above equation, α is the temperature parameter, which governs the stochasticity of the ideal policy, and dictates the weight of the entropy component compared to the reward.

A. Cooperative Multi-Agent Reinforcement Learning

Algorithms for multi-agent reinforcement learning deal with systems made up of a number of agents like robots, machines, and automobiles interacting in a single environment. In each

time step, each agent takes action and collaborates with the other agents to complete a specific, preset task. The purpose of cooperative MARL algorithms is to discover a policy for every agent which will enable them to work together in order to accomplish the system's objective. When modeling a multi-agent system, many characteristics of the system are crucial. To name a few, some important aspects in designing MARL systems are centralized or decentralized control, cooperative or competitive environment, and fully or partially observable environment. A central unit makes a choice for each agent in each time step inside a centralized controller. In contrast, each agent makes a choice for themselves in a decentralized system. Additionally, the agents might compete with one another to maximize their personal rewards, or they may work together to accomplish a shared objective. In any of these scenarios, the agent may have access to all of the other agents' information and sensory observations, or alternatively, each agent may just be able to view its own local information. The primary concern with MARL algorithms and multi-agent issues is the non-stationarity of the environment. Using a completely observable critic is one of the usual strategies for dealing with this problem. The environment remains stable even while other agents' policies vary because the fully observable critic takes into account the observations and acts of all agents. Because the environment always returns an equal future state, regardless of how other agents' policies change, one has:

$$P\left(s' \mid s, a_1, \ldots, a_N, \pi_1, \ldots, \pi_N\right) = P\left(s' \mid s, a_1, \ldots, a_N, \pi'_1, \ldots, \pi'_N\right) \tag{3}$$

In this paradigm, the critic's non-stationarity is eliminated until it is entirely observable, at which point local actors may employ the critic as a proper leader.

III. Multi-Agent Reinforcement Learning Method for Controlling Parallel Robots

In this section, a method based on MARL is introduced for controlling robots with more than one actuated joint in dynamic environments. A good example of such robots is parallel ones that have kinematic loops and multiple limbs. In the proposed algorithm, each constituting limb of the robot is assumed as a separate RL agent. The algorithm is designed in such a way that all agents learn to collaborate with each other in order to firstly hold the kinematic constraints imposed by the type synthesis of the robot and secondly to satisfy the task which has been defined for them to achieve the highest amount of defined reward based on the essence of the problem. The introduced algorithm is comprised of two main parts. First, it has an experience assembling mechanism in its replay buffer which facilitates the convergence of the algorithm. Second, as mentioned, it benefits from MARL as the pivotal algorithm to control the robot by actuating each robot's motor individually.

A. Experience Assembling

In order to ease the convergence of the algorithm, especially in dynamic environments, Experience Assembling (EA) has been embedded in the replay buffer of the Soft Actor-Critic (SAC) algorithm, which is the backbone of the MARL algorithm. The idea of experience assembling has come from [23], in which this method was used for solving the sparse reward problems, considering that the goal is to deal with a goal-reaching problem. However, the problem here is neither specified as a sparse reward problem nor goal-reaching. The EA method is used to cope with the dynamics of the environment and is applied to the dynamic states. Emerging here is the term dynamic states, which stands for the states that change at every time step and agents are unable to manipulate them, and they are originally part of the environment. As common replay buffers, the stored transition in the replay buffer can be written as the form of $(s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ where s_t^i, a_t^i, r_t^i, s_{t+1}^i are states, actions, reward value and next states at time step t of the agent i, respectively. Here for dealing with dynamic states, three parts are incorporated in each state which is being stored in the buffer. States for each agent at time step t can be written as $s_t^i =< o_t^i, s_t^d, s_{ref}^i >$, in which o_t^i is the observation of the agent i at time step t and s_t^d is the dynamic states at time step t which are jointed between all agents, since theses states are uncontrollable by the agents. s_{ref}^i is the reference signal for agent i at time step t and can be obtained for example using Inverse Kinematics (IK) of the robot, this additional states indicate the desired motor angle for each agent. These states are collected only for computing the reward value of each agent at time step t and are not given as inputs to the RL algorithm and the agents are not aware of them. Also, for assembling experiences and generating new trajectories, the reference signal states are used in a way to find two trajectories in which the observation of each agent at time step t of the first trajectory matches the reference signal of the agents at time step t' of the second trajectory. After finding these two trajectories, similar to the reasoning applied in [23], the new trajectory is assembled by combining o_t^i of the first trajectory which is found with the s_{ref}^i of the second trajectory. Algorithm 1 is called Multi-Agent Soft Actor-Critic with Experience Assembling (MASAC-EA).

B. Multi-Agent Reinforcement Learning

In the proposed algorithm, the framework of centralized training with decentralized execution has been used. For the sake of better understanding, a brief review of the relation and results of [24] are reviewed in what follows. As aforementioned, agents were trained independently from the model of the robot. This means the agents were trained without prior knowledge about the robot's dynamics. Since here the focus is on the dynamic control of the robots, meaning that the control input signals are considered as motor torque input signals, the model-free approach for addressing the problem could be regarded as a remedy to several challenges. This is because this method not only gives the privilege to be free from frictions and unmodeled dynamics of the system but also would be a savior from struggling with dynamic equations of the system, which most of the time are sophisticated and contain a lot of uncertainty. For using MARL, some assumptions are made. To the outset, the policy network for each agent only uses

the observations of that specific agent without having any information about the states and actions of other agents. Second, there is no specific communication structure between agents.

Consider a robot with N dynamic loops. For each of these loops an agent with Q-function and policy network parameterized by $\theta = \{\theta_1, ..., \theta_N\}$ and $\phi = \{\phi_1, ..., \phi_N\}$ can be considered, respectively. The soft Q-function for each of the agents is trained in a way that minimizes the loss function below which is known as the soft Bellman residual [25]:

$$J_Q(\theta_i) = \mathbb{E}_{(\mathbf{s}_t^i, \mathbf{a}_t^i) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_{\theta_i}(\mathbf{x}, \mathbf{a}_t) - \hat{Q}_{\theta_i}(\mathbf{x}, \mathbf{a}_t) \right)^2 \right] \quad (4)$$

Where one has [25]:

$$\hat{Q}_{\theta_i}(\mathbf{x}, \mathbf{a}_t) = r\left(\mathbf{s}_t^i, \mathbf{a}_t^i\right) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}^i \sim p} \left[V_{\bar{\theta}_i}\left(\mathbf{s}_{t+1}^i\right) \right] \quad (5)$$

in which $\hat{Q}_{\theta_i}(\mathbf{x}, \mathbf{a}_t)$ is the centralized action-value function for that it gets actions of all agents along with states of each of them. Therefore $\mathbf{a}_t$ can be written as $\mathbf{a}_t = \{a_t^1, ..., a_t^N\}$ and moreover, $\mathbf{x}$ is comprised of observations of all agents and the dynamic systems, so $\mathbf{x}$ can be expressed as $\mathbf{x} = \{o_t^1, ..., o_t^N, s_t^d\}$. Using Eqs. (4) and (5) , the stochastic gradients of Eq. (4) can be written as following [25]:

$$\hat{\nabla}_{\theta_i} J_Q(\theta_i) = \nabla_{\theta_i} Q_{\theta_i}(\mathbf{x}, \mathbf{a}_t) \left(Q_{\theta_i}(\mathbf{x}, \mathbf{a}_t) - \hat{Q}_{\theta_i}(\mathbf{x}, \mathbf{a}_t) \right) \quad (6)$$

It should be noted that the soft Q-function for any policy can be written as following [25]:

$$V(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi} \left[Q(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi(\mathbf{a}_t \mid \mathbf{s}_t) \right] \quad (7)$$

By combining Eqs. (5) and (7), the term $\hat{Q}_{\theta_i}(\mathbf{x}, \mathbf{a}_t)$ in Eq. (6) can be written as:

$$\hat{Q}_{\theta_i}(\mathbf{x}, \mathbf{a}_t) = r\left(\mathbf{s}_t^i, \mathbf{a}_t^i\right) + \gamma \left(Q_{\bar{\theta}_i}\left(\mathbf{o}_{t+1}^1, ..., \mathbf{o}_{t+1}^N, s_t^d, \mathbf{a}_{t+1}\right) - \alpha \log\left(\pi_{\phi_i}\left(\mathbf{a}_{t+1}^i \mid \mathbf{s}_{t+1}^i\right)\right)\right) \quad (8)$$

In the above equation, $\bar{\theta}_i$ represents the weights of the target Q-function for the ith agent and stabilizes the training and γ is the discount factor. For updating the policy network weights, the loss function is derived from KL-divergence [26]. The loss function for policy networks is as follows [25]:

$$J_\pi(\phi_i) = \mathbb{E}_{\mathbf{s}_t^i \sim \mathcal{D}} \left[\mathbb{E}_{\mathbf{a}_t^i \sim \pi_{\phi_i}} \left[\alpha \log\left(\pi_{\phi_i}\left(\mathbf{a}_t^i \mid \mathbf{s}_t^i\right)\right) - Q_{\theta_i}(\mathbf{x}, \mathbf{a}_t) \right] \right] \quad (9)$$

The policy for each agent can be represented using a neural network like below [25]:

$$\mathbf{a}_t^i = f_{\phi_i}\left(\epsilon_t; \mathbf{s}_t^i\right) \quad (10)$$

In Eq. (10), ϵ_t is a noise vector which is added to the actions produced by the actor network. By substituting this equation into Eq. (9), the following equation is obtained for agent i:

$$J_\pi(\phi_i) = \mathbb{E}_{\mathbf{s}_t^i \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}} [\alpha \log \pi_{\phi_i}\left(f_{\phi_i}\left(\epsilon_t; \mathbf{s}_t^i\right) \mid \mathbf{s}_t^i\right) - Q_{\theta_i}\left(\mathbf{s}_t^i, f_{\phi_i}\left(\epsilon_t; \mathbf{s}_t^i\right)\right)] \quad (11)$$

Algorithm 1 MASAC-EA for N-DOF Robot

$\mathcal{D} \leftarrow \emptyset$ ▷ Initialize an empty replay buffer
for $i \leftarrow 1, N$ **do**
  Input: $\theta_i^1, \theta_i^2, \phi_i$ ▷ Initial network parameters
  $\bar{\theta}_i^1 \leftarrow \theta_i^1, \bar{\theta}_i^2 \leftarrow \theta_i^2$ ▷ Initialize target networks
end for
for each iteration **do**
  for each environment step **do**
    for $i \leftarrow 1, N$ **do**
      $\mathbf{a}_t^i \sim \pi_{\phi_i}\left(\mathbf{a}_t^i \mid \mathbf{s}_t^i\right)$ ▷ Select action for agents
    end for
    $\mathbf{a}_t \leftarrow (\mathbf{a}_t^1, ..., \mathbf{a}_t^N)$ ▷ Concatenate agent actions
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$ ▷ Sample transition
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for $E_i \in \mathcal{D}$ **do** ▷ Experience Assembling
    Search another $E_j \in \mathcal{D}$ where $\mathbf{o}_t^{i,p} = \mathbf{s}_{ref}^{j,q}$
    if $E_j \neq \varnothing$ **then**
      Considering $\mathbf{s}_{ref}'^t = \mathbf{s}_{ref}^{j,q-m+t}$ in E_j
      Clone trajectory $\left\{\mathbf{s}_{ref}'^0, \cdots, \mathbf{s}_{ref}'^m\right\}_{m=\min\{p,q\}}$
      for $t = \{0, \cdots, m-1\}$ **do**
        $r_t' := r(\mathbf{s}_{i,p-m+t}, \mathbf{a}_{i,p-m+t})$
        Store $(\mathbf{s}_{i,p-m+t}, \mathbf{a}_{i,p-m+t}, r_t', \mathbf{s}_{i,p-m+t+1})$
      end for
    end if
  end for
  for each gradient step **do** ▷ Update networks
    Sample from buffer $\mathcal{D}$
    for $i \leftarrow 1, N$ **do**
      $\theta_i^j \leftarrow \theta_i^j - \lambda_Q \hat{\nabla}_{\theta_i^j} J_Q\left(\theta_i^j\right)$ for $j \in \{1, 2\}$
      $\phi_i \leftarrow \phi_i - \lambda_\pi \hat{\nabla}_{\phi_i} J_\pi(\phi_i)$
      $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$
      $\bar{\theta}_i^j \leftarrow \tau \theta_i^j + (1-\tau)\bar{\theta}_i^j$ for $j \in \{1, 2\}$
    end for
  end for
end for

For updating the actor parameters, the gradient of the Eq. (11) can be derived as follows [25]:

$$\hat{\nabla}_{\phi_i} J_\pi(\phi_i) = \nabla_{\phi_i} \alpha \log\left(\pi_{\phi_i}\left(\mathbf{a}_t^i \mid \mathbf{s}_t^i\right)\right) + \left(\nabla_{\mathbf{a}_t^i} \alpha \log\left(\pi_{\phi_i}\left(\mathbf{a}_t^i \mid \mathbf{s}_t^i\right)\right) - \nabla_{\mathbf{a}_t^i} Q\left(\mathbf{s}_t^i, \mathbf{a}_t^i\right) \right) \nabla_{\phi_i} f_{\phi_i}\left(\epsilon_t; \mathbf{s}_t^i\right) \quad (12)$$

C. Method Summary

Algorithm 1 represents the proposed method for controlling parallel robots using MARL in detail. The algorithm is comprised of two main components. First, it uses Experience Assembling in the replay buffer to alleviate the convergence of the algorithm. Second, it benefits from Multi-Agent Reinforcement Learning for controlling each of the dynamic loops or each of the degrees of freedom of the robot in a collaborative way. The underlying Deep RL algorithm is the so-called SAC, which is robust and data efficient.

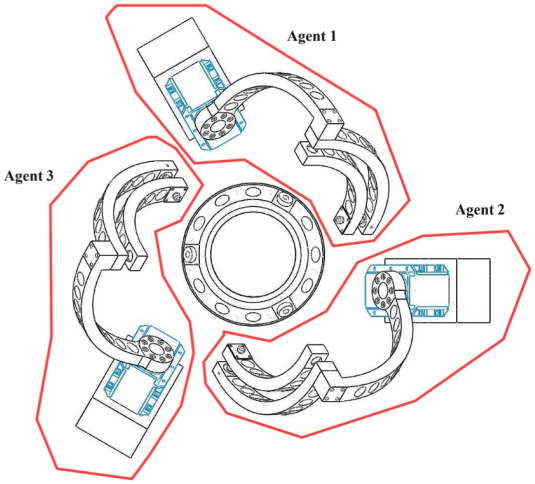

Fig. 2. The schematic of multi agent definition in the proposed algorithm for controlling parallel robots, considering each limb as a separate agent.

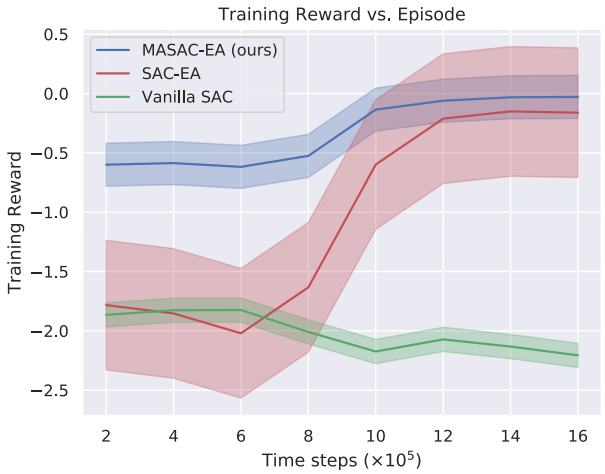

Fig. 3. Training reward for three different methods. All of the algorithms are trained for the robot's End-Effector reference tracking. For training these methods, all the hyper parameters are the same, also the same reward functions has been used for training.

IV. Simulations and Results

The proposed method has been evaluated on a type of parallel robot called the Agile Eye, a 3-DOF robot performing a spherical motion pattern with three dynamic loops and motors and complex dynamics with lots of uncertainty. For dynamically controlling the robot, each of the limbs of the robot has been considered as a separate agent like Fig. 2. In order to specify each of these limbs, states should be defined in a way to give complete information about each agent and because of that one can simply take the position and angular velocity of a motor as states of each agent. This is because motors are robot's actuated joints, and orientation and position of all other joints can be obtained using forward kinematics from knowing the states of actuated joints.

A. Simulation Setup

As a case study, the 3-DOF Agile Eye parallel robot has been simulated in the CoppeliaSim simulator along with Robotics Operation System (ROS). The proposed method has been utilized for the robot's EE reference tracking under disturbances, which makes it a challenging problem due to the random nature of disturbances. The disturbance is exerted on the system using a disturbance bed under the robot which has random movements. The objective of the proposed algorithm, i.e., Algorithm 1, consists in generating control input signals, which are torques here, in a way that leads to tracking the reference signal for the EE of the robot and makes the EE follow the prescribed orientation. In the defined problem, each component of the state of each agent can be determined. o_t^i is comprised of position and velocity of the motor i, s_t^d is the euler angles of the disturbance bed which is random and dynamic at each time step and is common between all agents, and lastly, s_{ref}^i is the reference motor position for agent i at time step t and indicates the motor positions for all agents in a way that if all agents reach their specified motor position, then the required reference trajectory will be tracked by the EE of the Agile Eye robot. This reference signal is obtained using the IK at each time step and as mentioned in Section III, agents are not aware of them. Furthermore, for achieving more realistic results, some noise was added into the values obtained from IK equations.

Moreover, the reward function which is used for agent i is the L_1 norm of subtraction between observation and reference signal of the agent i:

$$r_t^i = \left|\left|o_t^i - s_{ref}^i\right|\right|_1 \tag{13}$$

B. Experiment Overview

Upon implementing the proposed algorithm, several questions may arise as follows:

1) Is MARL an appropriate approach for controlling parallel robots?
2) Is MARL performance better than a single agent for controlling robots with complex dynamics?
3) What is the effect of Experience Assembling which is being used in the proposed method?
4) Can the proposed method outperform classic controllers, such as PID?

In order to answer and clarify the above questions, in whats follows the obtained results are fully discussed.

C. Results and Discussion

In this section, the first thing which is being investigated is the effect of each of the components in the convergence and training procedure of the algorithm. As mentioned in Section III, Experience Assembling and Multi-Agent RL are the principal elements of the introduced algorithm. In order to study the effect of each of these elements in the algorithm, three different models were trained for the task of reference tracking for the EE of the Agile Eye parallel robot. The first

TABLE I
RMSE VALUE COMPARISON FOR THREE CONTROLLING METHODS

Table RMSE	Controllers		
	Dynamic MASAC-EA	*Dynamic SAC-EA*	*Velocity PID*
Motor1	0.0208	0.0669	0.0095
Motor2	0.0173	0.0245	0.0090
Motor3	0.0171	0.0191	0.0090
Sum	0.0553	0.1105	0.0275

model is the proposed algorithm in this paper, i.e., MASAC-EA, the second one is only a single-agent SAC along with EA, and the last one is a vanilla single-agent SAC algorithm. By comparing the proposed algorithm with the second and the third abovementioned algorithms, the effect of each of the components of the suggested method can be evaluated. This comparison is depicted in Fig. 3. It is obvious that the proposed algorithm converges faster than two other algorithms and has a higher cumulative reward at the end, which demonstrates that the MARL approach is an appropriate approach for controlling parallel robots and answers to the first question from the Experiment Overview section. Moreover, by visual inspection of the third algorithm and comparing it with two other methods, the importance of the EA in the proposed method can be perceived which is the answer to the question number 3 from the Experiment Overview section. In overall, Fig. 3 illustrates the importance of each of the components of the suggested algorithm in its performance at training and convergence.

After investigating the effect of each of the MASAC-EA algorithm components in the convergence and training procedure, the performance of the recommended method with regard to other methods should be studied. For this purpose, two other controllers, one based on RL, the SAC-EA, and the other based on classic control, PID, have been taken into account. The represented Fig. 4 shows the Euler angles of the EE of the robot while it is being controlled dynamically using the MASAC-EA algorithm for reference tracking in a dynamic environment. These angles are measured with respect to the local reference coordinate of the robot in a way that if EE Euler angles follow the desired value represented with respect to the local robot's coordinate, then the desired trajectory for the robot's EE will be followed. In turn, Fig. 5 depicts the robot's EE Euler angles controlled by SAC-EA in a dynamic environment, considering torque value as the input control signal, similar to MASAC-EA. The reference trajectory for the robot is as same as the reference trajectory represented in Fig. 4 for better comparison between the algorithms. Finally, Fig. 6 illustrates the EE Euler angles of the robot being controlled by means of a PID controller in velocity mode. Moreover, for better illustration, Fig. 7 shows the error sum of all three Euler angles of the robot's EE at each time step in the trajectory, which has been depicted in Figs. 4, 5, and 6. For a better comparison between the three mentioned algorithms for the defined task, the Root-Mean-Squared Error (RMSE) between the desired position value and the actual position value of the

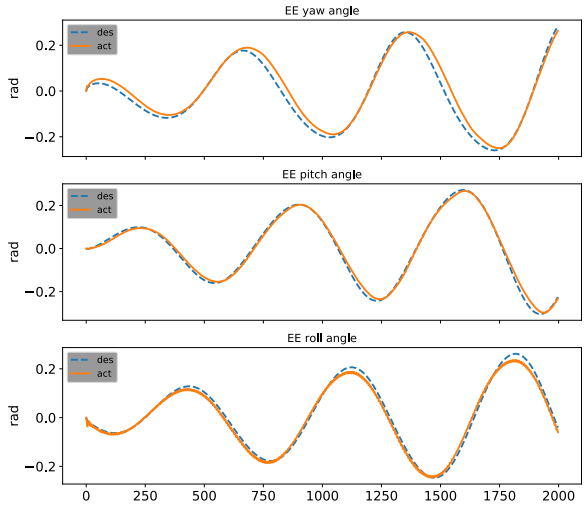

Fig. 4. EE Euler angles represented in local reference coordinate being dynamically controlled by MASAC-EA algorithm for reference tracking in dynamic environment.

motors has been utilized as an index. The RMSE formula is as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_{\mathrm{act}} - x_{\mathrm{des}}\right)^2} \tag{14}$$

The comparison between the performance of the algorithms has been shown in the Table I for each of the three motors. As shown in this table, it can be observed that dynamic MASAC-EA has a superior performance to the dynamic SAC-EA algorithm. It indicates that not only multi-agent control of parallel robots is better in convergence, but it also has a surpassing performance compared to a single agent; this is the answer to question number 2 from the Experiment Overview section. On the other hand, dynamic MASAC-EA performance is inferior to classic velocity PID controller, which answers question number 4 from the Experiment Overview section. It should be noticed that controllers in velocity mode are more stable in comparison with torque controllers, and that is one of the reasons that make PID controller performance better than the two other controllers, which control the robot dynamically. Moreover, it should be recognized that implementing a PID controller in torque mode for models with complex dynamics like the Agile Eye robot is challenging owing to the existing uncertainties in dynamic models, and it is not practically straightforward to apply torque PID in such systems. RL methods can compensate these uncertainties when it comes to torque control.

The final issue that should be discussed here is the complexity arising from using MARL and whether it is worth using it. In cases where precise torque control is required for parallel robots with complex uncertain equations, this

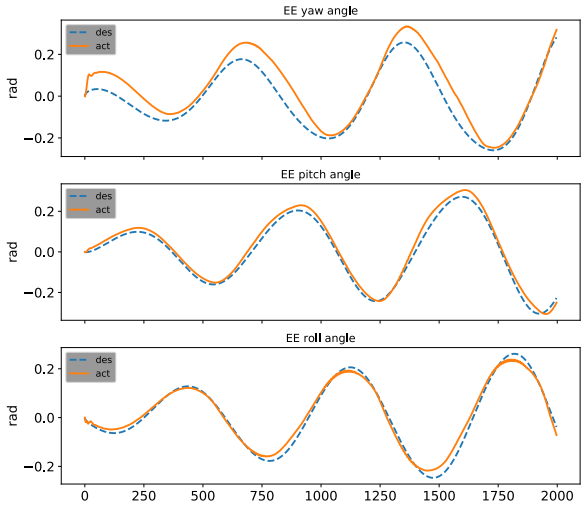

Fig. 5. EE Euler angles represented in local reference coordinate being dynamically controlled by SAC-EA algorithm for reference tracking in dynamic environment.

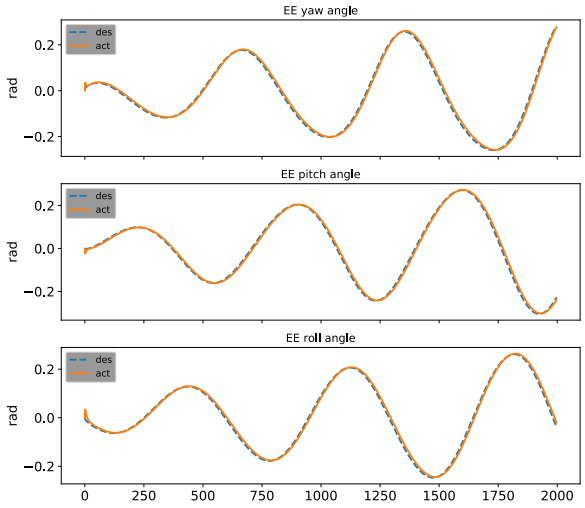

Fig. 6. EE Euler angles represented in local reference coordinate being controlled by PID algorithm in velocity control mode for reference tracking in dynamic environment.

method is effective. It should be noted that the computational complexity of such intelligent methods is the cost of the accuracy that these methods bring. In addition, it should be considered that in many robots and because of the disability of motors in actuating accurate torque, it is somehow impossible to apply traditional torque control methods.

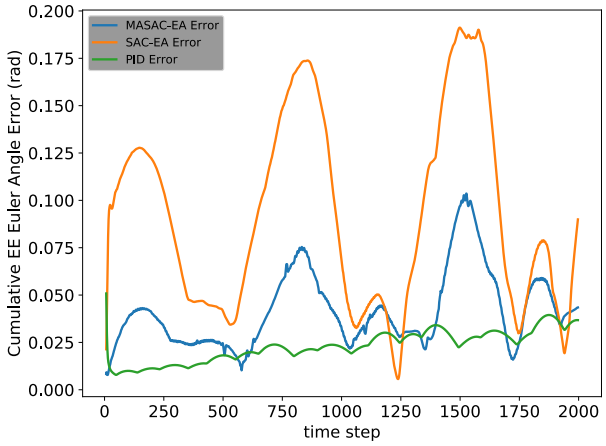

Fig. 7. EE Euler angles cumulative error for the same trajectory expressed in Figs. 4, 5 and 6, represented in local reference coordinate being controlled by three different algorithms, including dynamic MASAC-EA, dynamic SAC-EA and velocity PID controller for reference tracking purpose.

D. Simulation Details

All the hyper-parameters for training both MASAC-EA and SAC-EA algorithms were chosen the same. For training these algorithms, each epoch consists of 100 episodes with 2000 time steps. Learning rate for both critic and actor networks were chosen equal to 0.0003 and discount factor was equal to 0.98. The structure of actor and critic networks in both methods were almost the same except in the output layer. Actor and critic in both algorithms are 4 layer networks with 256 neurons in each of the hidden layers. Moreover, the actor network returns the probabilistic action from a gaussian distribution. For more stability, target critic networks were also utilized for training the agents. About the PID controller, K_p was chosen as 30 and K_i equal to 5 for all motors.

V. Conclusion

In this paper, a new method called MASAC-EA based on Multi-Agent RL has been suggested for controlling parallel robots which have multiple dynamic loops and actuated motors. The proposed algorithm is comprised of two main components, first Experience Assembling and second Multi-Agent RL, for controlling each of the dynamic loops of the robot separately but in a collaborative way. The underlying deep RL algorithm which is used in this algorithm is Soft Actor-Critic which can be regarded as a robust algorithm. The 3-DOF Agile Eye parallel robot was used as a case study to evaluate the introduced method. For that, firstly, the convergence and training process aspects of the mentioned algorithm have been studied by comparing the cumulative training reward of MASAC-EA with SAC-EA and vanilla SAC for the task of reference tracking of EE under random disturbance. This comparison has shown that the proposed algorithm is superior to two others in the training process by reaching a higher ultimate reward and faster convergence. Secondly, the

performance of the MASAC-EA was investigated by comparing its performance with SAC-EA and velocity PID controller, and due to the results, MASAC-EA with RMSE index value of 0.0553 has better performance than its similar single agent algorithm, SAC-EA with RMSE index value of 0.1105. On the other hand, velocity PID controller performance with an RMSE index value of 0.0275 exceeds MASAC-EA in torque mode control, and the reason is that velocity controllers are more robust than torque controllers. Moreover, it should be noticed that because of the existing uncertainties in complex dynamic models, especially in parallel robots, it is challenging to practically implement classic torque controllers in such robots, so that a better choice could be utilizing intelligent methods such as RL-based controllers for torque control of sophisticated dynamic systems. As an ongoing work, the designed algorithm can be refined in a way that outperforms classic controllers even in velocity mode. Also, the practical implementation of the proposed algorithm on a real robot can be considered for future work.

REFERENCES

[1] J. Gallardo-Alvarado, "A gough-stewart parallel manipulator with configurable platform and multiple end-effectors," Meccanica, vol. 55, no. 3, pp. 597-613, 2020.

[2] M. Azmoun, A. Rouhollahi, M. T. Masouleh, and A. Kalhor, "Kinematics and control of a 4-dof delta parallel manipulator," in 2018 6th RSI International Conference on Robotics and Mechatronics (IcRoM). IEEE, 2018, pp. 494-500.

[3] M. Zarei, A. Aflakian, A. Kalhor, and M. T. Masouleh, "Oscillation damping of nonlinear control systems based on the phase trajectory length concept: An experimental case study on a cable-driven parallel robot," Mechanism and Machine Theory, vol. 126, pp. 377-396, 2018.

[4] C. M. Gosselin, M. T. Masouleh, V. Duchaine, P.-L. Richard, S. Foucault, and X. Kong, "Parallel mechanisms of the multipteron family: kinematic architectures and benchmarking," in Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, 2007, pp. 555-560.

[5] S. Ansari-Rad, M. Zarei, M. G. Tamizi, S. M. Nejati, M. T. Masouleh, and A. Kalhor, "Stabilization of a two-dof spherical parallel robot via a novel adaptive approach," in 2018 6th RSI International Conference on Robotics and Mechatronics (IcRoM). IEEE, 2018, pp. 369-374.

[6] L.-W. Tsai, Robot analysis: the mechanics of serial and parallel manipulators. John Wiley and Sons, 1999.

[7] R. E. Stamper, "A three degree of freedom parallel manipulator with only translational degrees of freedom," Ph.D. dissertation, 1997.

[8] Y. Li, Y. Ma, S. Liu, Z. Luo, J. Mei, T. Huang, and D. Chetwynd, "Integrated design of a 4-dof high-speed pick-and-place parallel robot," CIRP Annals, vol. 63, no. 1, pp. 185-188, 2014.

[9] Bonev, Ilian A., Damien Chablat, and Philippe Wenger. "Working and assembly modes of the Agile Eye." Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.. IEEE, 2006.

[10] Gosselin, Clement M., E. St Pierre, and Martin Gagne. "On the development of the agile eye." IEEE Robotics and Automation Magazine 3.4 (1996): 29-37.

[11] Arian, Alaleh, Behzad Danaei, and Mehdi Tale Masouleh. "Kinematics and dynamics analysis of a 2-dof spherical parallel robot." 2016 4th international conference on robotics and mechatronics (ICROM). IEEE, 2016.

[12] Radmehr, Amirmohammad, Milad Asgari, and Mehdi Tale Masouleh. "Experimental Study on the Imitation of the Human Head-and-Eye Pose Using the 3-DOF Agile Eye Parallel Robot with ROS and Mediapipe Framework." 2021 9th RSI International Conference on Robotics and Mechatronics (ICRoM). IEEE, 2021.

[13] Zare, S., Yazdi, M. R. H., Masouleh, M. T., Zhang, D., Ajami, S., and Ardekani, A. A. (2022). Experimental study on the control of a suspended cable-driven parallel robot for object tracking purpose. Robotica, 1-15.

[14] Rahimi, S., Jalali, H., Yazdi, M. R. H., Kalhor, A., and Masouleh, M. T. (2022). 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, 84, 102772.

[15] H. D. Taghirad, Parallel robots: mechanics and control. CRC press, 2013.

[16] Rad, S. A., Tamizi, M. G., Mirfakhar, A., Masouleh, M. T. ,and Kalhor, A. (2021). Control of a two-DOF parallel robot with unknown parameters using a novel robust adaptive approach. ISA transactions, 117, 70-84.

[17] Perrusquía, Adolfo, Wen Yu, and Alberto Soria. "Position/force control of robot manipulators using reinforcement learning." Industrial Robot: the international journal of robotics research and application 46.2 (2019): 267-280.

[18] Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., ... and Levine, S. (2019, May). Residual reinforcement learning for robot control. In 2019 International Conference on Robotics and Automation (ICRA) (pp. 6023-6029). IEEE.

[19] Kiumarsi, B., Vamvoudakis, K. G., Modares, H.,and Lewis, F. L. (2017). Optimal and autonomous control using reinforcement learning: A survey. IEEE transactions on neural networks and learning systems, 29(6), 2042-2062.

[20] Meyes, R., Tercan, H., Roggendorf, S., Thiele, T., Büscher, C., Obdenbusch, M., ... and Meisen, T. (2017). Motion planning for industrial robots using reinforcement learning. Procedia CIRP, 63, 107-112.

[21] Richard S Sutton and Andrew G Barto. Reinforcement Learning: An Introduction. MIT Press, 1998.

[22] Ziebart, B. D. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. Carnegie Mellon University, 2010.

[23] Fang, M., Zhou, C., Shi, B., Gong, B., Xu, J.,and Zhang, T. (2018, September). DHER: Hindsight experience replay for dynamic goals. In International Conference on Learning Representations.

[24] Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O.,and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems, 30.

[25] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P. and Levine, S., 2018. Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905.

[26] Haarnoja, T., Zhou, A., Abbeel, P.,and Levine, S. (2018, July). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning (pp. 1861-1870). PMLR.