International Journal of Computing and Digital Systems
2025, VOL. 17, NO.1, 1-13

http://dx.doi.org/10.12785/ijcds/1571033189

Collaborative Multi-Agent Deep Reinforcement Learning Approach for Enhanced Attitude Control in Quadrotors

Taha Yacine Trad[1], Kheireddine Choutri[1], Mohand Lagha[1] and Fouad Khenfri[2]

[1]*Aeronautical Sciences Laboratory, Aeronautical and Spatial Studies Institute, Blida 1 University, Blida, Algeria*
[2]*Energy and Embedded Systems for Transport, ESTACA'Lab, Laval, France*

Received 26 May 2024, Revised 13 July 2024, Accepted 27 August 2024

Abstract: Unmanned Aerial Vehicles (UAVs), particularly quadrotors, have become highly versatile platforms for various applications and missions. In this study, the use of Multi-Agent Reinforcement Learning (MARL) in quadrotor control systems is investigated, expanding its conventional usage beyond multi-UAV path planning and obstacle avoidance tasks. While traditional single-agent control techniques face limitations in effectively managing the coupled dynamics associated with attitude control, especially when exposed to complex scenarios and trajectories, this paper presents a novel method to enhance the adaptability and generalization capabilities of Reinforcement Learning (RL) low-level control agents in quadrotors. We propose a framework consisting of collaborative MARL to control the roll, pitch, and yaw of the quadrotor, aiming to stabilize the system and efficiently track various predefined trajectories. Alongside detailing the overall system architecture of the MARL-based attitude control system, we elucidate the training framework, collaborative interactions among agents, neural network structures, and implemented reward functions. While experimental validation is pending, theoretical analyses and simulations illustrate the envisioned benefits of employing MARL for quadrotor control in terms of stability, responsiveness, and adaptability. Central to our approach is the use of multiple actor-critic algorithms within the proposed control architecture. Through a comparative study, we evaluate the performance of the advocated technique against a single-agent RL controller and established linear and nonlinear methodologies, including Proportional-Integral-Derivative (PID) and Backstepping control, highlighting the advantages of collaborative intelligence in enhancing quadrotor control in complex environments.

Keywords: Quadrotors, Attitude Control, Multi-Agent Deep Reinforcement Learning, Collaborative Intelligence

1. Introduction

Owing to their unparalleled agility and versatility in navigating complex environments, quadrotors have become indispensable tools across a myriad of applications. Precise control of a quadrotor's attitude, encompassing roll, pitch, and yaw, is essential for achieving stable and responsive flight operations [1]. However, the complex interactions between aerodynamic forces, rotor speeds, and the vehicle's orientation lead to nonlinear dynamics in quadrotors, complicating the design of efficient and robust control systems. Furthermore, these systems exhibit underactuation, where the number of control inputs (rotors) is fewer than the degrees of freedom (six degrees: three rotational and three translational), thereby limiting direct control authority over all axes simultaneously. Significant coupling also exists between the different axes of motion (roll, pitch, and yaw), meaning adjustments in one axis can influence the dynamics of the others. These interdependencies necessitate controllers to carefully manage cross-axis interactions to achieve desired flight behaviors effectively. As a result, researchers have introduced and continue to explore multiple control theories and techniques to tackle these intricate challenges, aiming to achieve efficient attitude control for these widely utilized systems.

In the realm of quadrotor control, various methodologies and techniques have been explored. Linear methods, such as PID [2], LQR [3], and Model Predictive Control [4], offer simplicity and ease of implementation but are limited in their operational scope. To overcome these limitations, more complex nonlinear controllers have been introduced, including Sliding Mode Control [5], Backstepping [6], Adaptive Control [7], and H_∞ Control [8]. The efficacy of traditional control algorithms often depends on subjective parameter choices, which rely on a comprehensive understanding of the model and experimental context. Balancing accuracy, robustness, and efficiency within a single control function becomes notably challenging in complex scenarios. Deep Reinforcement Learning (DRL), enabled by advancements in computing power and data accessibility, has emerged

E-mail address: Trad.tahayacine@etu.univ-blida.dz, choutri.kheireddine@univ-blida.dz, laghamohand@univ-blida.dz, Fouad.KHENFRI@estaca.fr

as a powerful approach within control theory. It demonstrates remarkable advantages across diverse tasks and applications, often surpassing conventional control methods by autonomously learning control policies directly from interaction with the environment, without relying on explicit models. DRL algorithms can adapt to diverse systems by continuously updating their policies, allowing them to handle uncertain and dynamic environments where traditional control methods may struggle.

Building on the success of single-agent RL techniques, Multi-Agent Reinforcement Learning has emerged as a promising approach for addressing complex control problems, leveraging the adaptability and learning capabilities of intelligent agents. MARL enables multiple agents to interact and collaborate to achieve common goals, making it particularly suitable for scenarios involving coordination and cooperation. By harnessing the power of MARL in collaborative decision-making processes among UAVs, researchers and control engineers have recently focused on developing innovative and sophisticated control strategies that enhance the stability, responsiveness, and adaptability of quadrotor systems in various areas, including multi-UAV path planning with collision and obstacle avoidance tasks [9].

Nevertheless, since single-agent DRL agents learn through iterative interactions with the environment to discover and refine effective strategies, in the context of quadrotor control, where stability, responsiveness, and safety are paramount, these approaches typically face two primary challenges: the need for large neural networks to handle the complex dynamics of these systems, and extensive training time to converge on optimal policies in dynamic and unpredictable environments. While single-agent DRL shows promise in autonomous decision-making and adaptive control, overcoming these limitations is crucial for its effective application in time-sensitive or safety-critical operations. In contrast, Multi-Agent Reinforcement Learning offers substantial advantages for achieving intricate control objectives by decomposing them into more manageable subgoals [10]. This collaborative methodology enhances the learning process by allowing multiple agents to share information and insights, thereby accelerating the learning rate, minimizing the size of the neural networks employed, and improving the overall control performance.

In this study, we introduce an innovative MARL-based approach for quadrotor attitude control, specifically focusing on multiple baseline actor-critic RL algorithms: Twin Delayed Deep Deterministic Policy Gradient (TD3), Deep Deterministic Policy Gradient (DDPG), Soft Actor-Critic (SAC), and Proximal Policy Optimization (PPO) within a collaborative multi-agent framework. This approach marks a departure from traditional control paradigms by leveraging collaborative intelligence to optimize control strategies and expand the scope of MARL beyond multi-UAV path planning and obstacle avoidance, demonstrating its potential to revolutionize the low-level control systems of quadrotors. Through theoretical analyses and simulations, the proposed approach aims to validate the benefits of employing MARL in UAV systems, paving the way for innovative control capabilities and robustness.

Compared to existing studies in the literature, this approach offers the following contributions:

- Investigation of the quadrotor attitude control system using a MARL-based approach.
- Proposal of neural network architectures that incorporate high-order observable states to enhance collaborative behavior among the agents and ensure efficiency for the overall control system.
- Design of a straightforward reward strategy that guides the quadrotor's attitude through collaborative reward collection.
- Presentation of a novel collaborative MARL-based approach that ensures stable and rapid training convergence, along with high-performance properties.
- Performance comparison of the proposed approach against a single-agent RL controller and two conventional controllers under different scenarios, with results provided and discussed.

The subsequent sections of this paper first delve into the related works and the theoretical foundation of MARL, along with the quadrotor's dynamic model and control. We then present the architecture of the multi-agent-based attitude control system, detailing the employed framework, neural network structures, and reward functions. Finally, we present simulation results and performance comparisons against single-agent RL, as well as linear and nonlinear benchmark control techniques, including PID and Backstepping control, to validate the effectiveness of the proposed method in achieving stable and precise quadrotor attitude control across various scenarios.

2. Background

A. Related Works

In this section, we explore the extensive body of literature surrounding single-agent RL and MARL. We begin with relevant papers and studies on data-driven control approaches for quadrotor systems, followed by recent reviews and algorithmic advancements in MARL to provide an insightful overview of the methodologies, insights, and research trends. Finally, we focus on the application of MARL in UAV domains, highlighting novel breakthroughs and promising directions for future exploration.

Data-driven control systems, particularly RL approaches, have been extensively investigated both independently and in conjunction with conventional model-based techniques for UAVs and quadrotor control. Various RL algorithms have been employed in this field. For example, [11] utilized DQN with the Mean Squared Error (MSE) of

Euclidean distance in the reward function and the Adadelta optimizer, showing superior performance in quadrotor flight compared to Q-learning and SARSA. This approach was tested and validated across different optimizers (RMSProp, Adadelta, SGD, and Adam) and reward functions. Furthermore, [12] introduced on-policy algorithms for quadrotor control, focusing on the Proximal Policy Optimization algorithm. The study validated the efficacy of PPO in practical experiments, demonstrating its capability in tasks such as maintaining stationary positions and following predefined paths. Moreover, [13] implemented the Deep Deterministic Policy Gradient algorithm to address trajectory tracking for quadrotors in three ways: incorporating instantaneous path information, anticipating path curves, and optimizing speed based on path slopes. Other strategies have recently explored combining RL with traditional control techniques. For instance, in [14], the authors used the Deep Deterministic Policy Gradient algorithm to optimize state and control weighting matrices for LQR control of a quadrotor. This method significantly improved response times and minimized integral square error compared to four commonly used methods, demonstrating potential for broader application in enhancing control efficiency. Additionally, in [15], researchers developed a Feedforward Neural Network (FFNN) to predict a quadrotor's translational dynamics. Integrated into a Model Predictive Control framework, this Neural Network-based Model Predictive Controller (NN-MPC) reduced the average path-following error by 40% compared to traditional PID controllers.

To comprehensively cover the current state of research in MARL, we have selected several reviews highlighting the methodologies, challenges, and breakthroughs in this rapidly evolving field. The study in [16] underscores the significance of MARL in multi-robot systems, emphasizing collaborative learning. It identifies a scarcity of recent surveys in the field, discusses challenges, and proposes future applications for enhancing multi-robot systems. In [17], the authors explored recent advances in algorithms employed in MARL, concentrating on five key approaches for addressing cooperative multi-agent problems. The review offers detailed explanations, discusses challenges, and highlights connections among various papers. Additionally, the article covers emerging research areas, real-world applications, and MARL research environments. In [18], the paper provides an overview divided into three main parts. It first examines training schemes for multiple agents, followed by an analysis of agent behavioral patterns in cooperative, competitive, and blended scenarios. The third part addresses challenges unique to the multi-agent domain and reviews methods used to address them. Moreover, in [19], a thorough analysis of MARL algorithms is provided, categorizing them based on features and offering a detailed taxonomy. This review explores application fields, pros and cons, and compares algorithms in terms of nonstationarity, scalability, and observability, while discussing common benchmark environments.

The landscape of Multi-Agent Reinforcement Learning has witnessed remarkable advancements recently, marked by the introduction of innovative algorithms designed to tackle the complexities of cooperative and competitive multi-agent environments. However, due to the vast array of MARL algorithms available, only a select few will be presented here. In [20], the study addresses the challenge of state uncertainty in practical MARL implementations. It introduces the Markov Game with State Perturbation Adversaries (MG-SPA) model and proposes a Robust Multi-Agent Q-learning (RMAQ) algorithm with convergence guarantees, along with a Robust Multi-Agent Actor-Critic (RMAAC) algorithm. The study demonstrates their efficacy in handling high-dimensional state-action spaces and outperforming existing methods in scenarios with state uncertainty. In [21], the authors introduced RACE, a hybrid framework combining Evolutionary Algorithm (EA) and MARL to address challenges such as collaboration, low-quality reward signals, and high non-stationarity. RACE achieved improved convergence, robustness, and signal quality in various tasks compared to other algorithms. Additionally, the authors of [22] introduced a distributed zeroth-order policy optimization method for MARL. This method enables agents to compute local policy gradients using only partial state and action data, reducing communication overhead and improving learning performance. Numerical experiments demonstrated enhanced sample efficiency compared to existing one-point estimators. In [23], the paper addresses computational inefficiency in Population-based MARL (PB-MARL) by proposing a solution that employs a stateless central task dispatcher and stateful workers. This approach facilitates parallelism and efficient problem-solving. The proposed framework, MALib, integrates a task control model, independent data servers, and a streamlined representation of MARL training methods, offering enhanced computational efficiency.

In the field of UAVs, MARL approaches have been applied in diverse ways, showcasing their versatility and effectiveness in addressing a wide range of challenges and objectives. In [24], the authors present a novel method to maximize data offloading efficiency from terrestrial Base Stations (BSs) using multiple UAVs. By jointly optimizing UAV trajectories and user association indicators under Quality-of-Service (QoS) constraints, the method aims to enhance user association with UAVs. Leveraging Multi-Agent Reinforcement Learning, each UAV operates independently while fostering cooperative behavior among them. Extensive simulations validate the effectiveness of the proposed technique, showing higher performance than both Q-learning and Particle Swarm Optimization (PSO). In a similar context, [25] introduces a novel approach to UAV cellular communication using multi-agent learning techniques. This approach enables multiple UAVs to learn from each other through communication and interaction with the environment, providing better coverage compared to conventional terrestrial Base Station (BS) deployment. Additionally, [26] introduces a novel Graph-Attention Multi-

Agent Trust Region (GA-MATR) reinforcement learning framework to address the multi-UAV assisted communication problem. This approach utilizes multiple UAVs to maximize data offloading efficiency from terrestrial BSs by jointly optimizing UAV trajectories and user association indicators under QoS constraints. The method was validated through simulations, demonstrating superior performance compared to benchmark techniques.

Another aspect in the UAV domain explored using MARL is described in [27], where the paper introduces a UAV-aided Mobile Edge Computing (MEC) framework employing a multi-agent deep reinforcement learning algorithm, specifically the Multi-Agent Deep Deterministic Policy Gradient (MADDPG). This framework aims to optimize geographical fairness, UAV User Equipment (UE) load fairness, and overall energy consumption for UEs, showcasing superior performance compared to traditional algorithms. In [28], the study investigated dynamic resource allocation in UAV communication networks using a MARL framework. This framework optimizes long-term rewards without inter-UAV information exchange, demonstrating enhanced performance with balanced exploration and exploitation parameters. The proposed approach achieved a favorable balance between performance improvements and the overhead of exchanging information, contrasting with scenarios where UAVs exchange complete information.

Several papers incorporating MARL approaches have recently emerged on the topics of path following and swarm formations. In [29], the authors introduced a decentralized Multi-Agent Deep Reinforcement Learning (MADRL) method using maximum reciprocal reward. They leveraged a Pointwise Mutual Information (PMI) neural network to capture dependencies among UAVs and proposed the Reciprocal Reward Multi-Agent Actor-Critic (MAAC-R) algorithm for cooperative tracking policies in UAV swarms. The study demonstrated enhanced cooperation and scalability in unknown environments compared to baseline algorithms. In [30], the authors presented a model for cooperative air combat maneuvers involving multiple UAVs, based on Bidirectional Recurrent Neural Networks (BRNN) and actor-critic architecture within the MARL framework. The model effectively achieved cooperative tactical maneuver policies, providing UAVs with situational advantages and tactical success in air combat scenarios. Another study, [31], introduced an autonomous tracking system for a UAV swarm to localize a radio frequency (RF) mobile target. This study utilized omnidirectional Received Signal Strength (RSS) sensors and an enhanced MARL technique to optimize real-time target tracking. The approach demonstrated superior performance in search time and successful localization probability compared to standard Q-learning and multi-agent Q-learning algorithms.

While most studies have predominantly focused on applying Multi-Agent Reinforcement Learning approaches in areas such as swarm intelligence, path planning, collision avoidance, task allocation, communication networks, and target tracking, there is a notable gap in exploring MARL's potential in low-level control systems, particularly in the domain of quadrotor flight. MARL has primarily been utilized in high-level control strategies within swarm and multi-UAV contexts, often overlooking its potential to revolutionize the control of individual quadrotors. By extending its traditional usage and exploring the application of collaborative intelligent agents to enhance quadrotor attitude control systems, this paper aims to unlock new possibilities for improving stability, maneuverability, and responsiveness at the fundamental level of quadrotor flight.

To the best of our knowledge, the approach most similar to the one presented in this study is introduced in [32], where the authors proposed a cascade MARL control framework for quadrotors. They decompose the system's dynamics into six one-dimensional subsystems. Their method employs a small-angle restriction to reduce dependency among multiple single agents trained independently using traditional RL algorithms without cooperative interaction, thereby significantly constraining the quadrotor's maneuverability. Another study, [10], introduced a MARL framework for quadrotor low-level control by decomposing the system dynamics into distinct translational and yawing components. They assigned two RL agents to collaborate and manage each component. Through simulations, this MARL framework outperformed traditional single-agent methods, significantly enhancing training efficiency and maintaining high-quality flight control performance. However, compared to our work, the authors used only the TD3 algorithm to validate their approach and did not assess the trained policies' generalization across various scenarios involving complex and dynamic trajectories.

B. *Quadrotor Dynamics*

The dynamics of the quadrotor UAV model, as illustrated in Figure 1, are crucial for understanding its behavior and control. These dynamics can be effectively described using the Euler-Lagrangian formulation, which provides a systematic approach to deriving the equations of motion governing both the rotational and translational motion of the quadrotor [33].

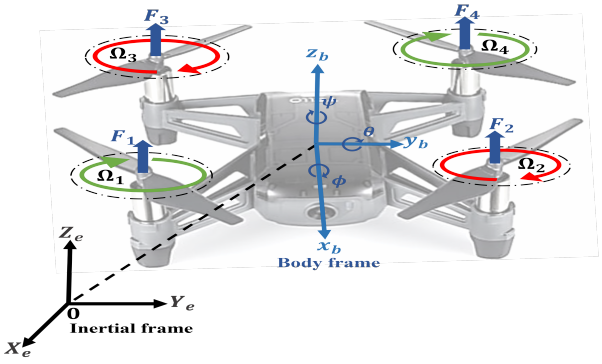

Figure 1. Quadrotor body and inertial frames

The translational motion of a quadrotor along the x, y and z axes is influenced by the forces generated by its propellers. Treating the quadrotor as a rigid body, the equations of motion can be expressed as follows:

$$\begin{cases} \ddot{x} = \frac{U_1}{m}(cos(\varphi)sin(\theta)cos(\psi) + sin(\varphi)sin(\psi)) \\ \ddot{y} = \frac{U_1}{m}(cos(\varphi)sin(\theta)sin(\psi) - sin(\varphi)cos(\psi)) \\ \ddot{z} = \frac{U_1}{m}(cos(\varphi)cos(\theta)) - g \end{cases} \quad (1)$$

The rotational motion of the quadrotor involves angular velocities about its body-fixed axes. The moments induced by the propellers result in rotational acceleration, which can be described as:

$$\begin{cases} \ddot{\varphi} = \frac{-(J_{zz}-J_{yy})\dot{\theta}\dot{\psi}+U_2}{J_{xx}} \\ \ddot{\theta} = \frac{-(J_{zz}-J_{xx})\dot{\phi}\dot{\psi}+U_3}{J_{yy}} \\ \ddot{\psi} = \frac{-(J_{yy}-J_{xx})\dot{\phi}\dot{\theta}+U_4}{J_{zz}} \end{cases} \quad (2)$$

By deriving and solving these equations, one can gain insights into the quadrotor's behavior and design control strategies to achieve desired maneuvers and stability. This analysis focuses on the following control inputs:

$$\begin{cases} U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ U_2 = lb(\Omega_4^2 - \Omega_2^2) \\ U_3 = lb(\Omega_3^2 - \Omega_1^2) \\ U_4 = d(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{cases} \quad (3)$$

The symbols used are as follows: m, g, and l represent the quadrotor's mass, gravitational acceleration, and half-length, respectively. J_{xx}, I_{yy}, and I_{zz} denote the inertial tensor of the symmetric body around the x, y and z axis. Additionally, d and b are the drag and thrust factors. Ω_i corresponds to the speed of each rotor i and U_i represents the lift force and moments of roll, pitch, and yaw.

C. Multi-Agent Reinforcement Learning Formulation

Multi-Agent Reinforcement Learning (MARL) expands upon single-agent reinforcement learning by addressing scenarios involving multiple autonomous entities interacting in shared environments. Unlike single-agent RL, MARL introduces challenges related to non-stationarity and the combinatorial nature of interactions among agents. Agents within a multi-agent system must learn policies while adapting to the dynamic behaviors of others, which adds complexity to the learning process. This collaborative or competitive aspect requires advanced algorithms capable of managing the intricacies and uncertainties inherent in multi-agent environments.

The MARL approach aims to enhance the collective reward achieved by a group of agents and can be represented as a Markov game involving a community of N agents (Figure 2). Markov Decision Processes (MDPs) are characterized by sequential decision-making, where actions impact both the immediate rewards of the agent and the future states of the environment. The state variable, $s \in S$, represents the environment's current state. For each agent $i = \{1, ..., N\}$, the action and observation spaces are represented respectively by $A_i \in A = \{A_1, ..., A_N\}$ and $O_i \in O = \{O_1, ..., O_N\}$. At each time step t, given a state s_t, agent i receives a local observation o_i and subsequently, interacts with the environment following a random policy denoted as π_{ϕ^i}, by taking action $a_t \in A$ and receiving an immediate reward R_{t+1}, The environment then transitions from the current state $s_t \in S$ to the next state $s_{t+1} \in S$. Corresponding to the action and the state, the rewards attributed to the independent agent i are denoted by r_i. The objective of each agent is to maximize the expected return $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$, where T is the final time step, and γ is a discount factor.

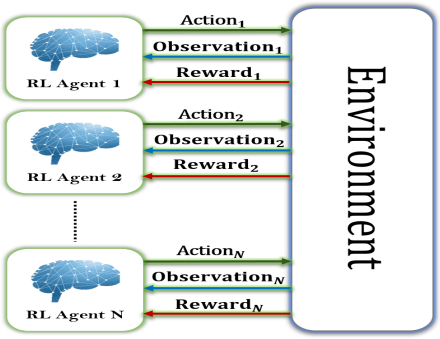

Figure 2. MARL framework

Similar to the single-agent case, the goal in MARL is to acquire the optimal stochastic policy or the optimal Q-value. However, due to the dynamic nature of each agent's policies during training, the environment exhibits non-stationarity from the perspective of any individual agent. In essence, $P(s_{t+1}|s_t, a_i, \pi_{\phi^1}, ..., \pi_{\phi^N}) \neq P(s_{t+1}|s_t, a_i, \pi'_{\phi^1}, ..., \pi'_{\phi^N})$ with any $\pi_{\phi^i} \neq \pi'_{\phi^i}$, causing the loss of the underlying assumption of a Markov Decision Process. This implies that each agent's experience involves interactions with diverse co-agent policies, making it challenging to stabilize these policies for training. Consequently, training such models often results in fluctuations during the training process, presenting a significant challenge.

To tackle this challenge, a commonly used strategy is to employ a fully observable critic. This approach entails integrating the actions and observations of all agents into the critic's perspective, thereby ensuring stability in the environment even as other agents' policies change. In simpler terms, even if $\pi_{\phi^i} \neq \pi'_{\phi^i}$, the probability of transitioning to a new state remains equal when other agents alter their policies, enabling the creation of either a single central critic in fully cooperative scenarios or N critic models when each agent observes a local reward. In both cases, the fully observable critic resolves the non-stationarity issue, providing an effective guidance mechanism for local actors.

3. MARL Control System

With the foundational background established, we delve in this section into the intricacies of the quadrotor control system, explore the innovative approach for quadrotor attitude control, and examine the neural network architectures and reward functions employed.

A. MARL Attitude Control System

The quadrotor, emblematic of under-actuated systems, intricately balances its movement across six degrees of freedom using only four control inputs. While the high-level controller orchestrates the quadrotor's position and altitude, it is the low-level controller that serves as the backbone for precise maneuvering and stabilization.

At the heart of the low-level controller lies a network of RL agents trained in a full cooperation configuration to maintain a steady orientation, ensuring that the quadrotor's roll (φ), pitch (θ), and yaw (ψ) angles remain consistent amidst dynamic environmental conditions by adeptly interpreting input commands (U_2, U_3, U_4) to achieve precise adjustments in the quadrotor's moments. These adjustments are calibrated to synchronize with the high-level controller's directives, allowing for seamless integration between attitude control and overall navigation, as illustrated in Figure 3. Applying this approach is pivotal in achieving stability and precision in flight and overcoming the coupled dynamics issue of the quadrotor system.

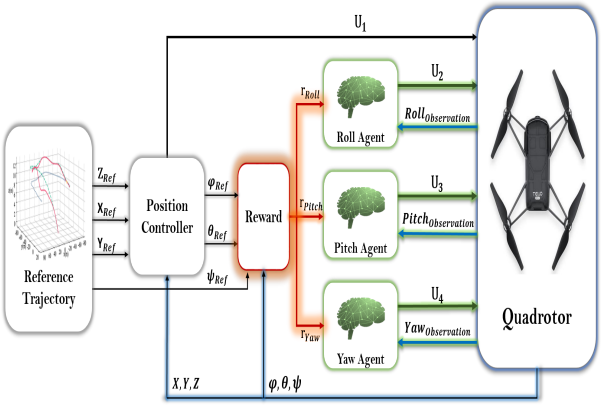

Figure 3. Multi-agent attitude control system

For this critical task that necessitates policies with high-dimensional and continuous action space capabilities, we have selected the TD3, DDPG, SAC, and PPO algorithms. Each of these benchmark RL algorithms brings unique advantages to the table. TD3 is known for its stability and performance in continuous control tasks, making it well-suited for precise attitude control in dynamic environments. DDPG, on the other hand, offers efficient exploration capabilities, allowing the agents to learn robust policies even in high-dimensional action spaces. SAC stands out for its ability to handle continuous action spaces with stochastic policies, providing flexibility and adaptability in uncertain conditions. Finally, PPO offers a balance between sample efficiency and simplicity, making it suitable for online learning scenarios where computational resources are limited. By leveraging the strengths of these algorithms in a cooperative formation, the proposed framework can effectively maintain a steady orientation under various conditions, ensuring the stability and reliability of the advocated control system.

B. MATD3 Algorithm

In this section, we will provide a detailed explanation of the MATD3 attitude controller. While the overall framework applies to all configurations using the same reward functions and neural network architectures, including MADDPG, MASAC, and MAPPO.

In a multi-agent environment, the authors of [34] present a model-free MARL approach designed to address scenarios where agent i, during time step t, uses its individual local observation o_i, actions a_i, and rewards r_i. Their approach encompasses competitive, cooperative, and mixed cooperative and competitive games. They introduce the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm, where each agent undergoes training with a DDPG algorithm. In this setup, the actor $\pi_{\phi^i}(o_i;\phi^i)$, characterized by policy weights ϕ^i, processes local observations. Meanwhile, the critic $Q^{\mu}_{\theta^i}$ (with θ^i representing its weights) has access to the actions, observations, and target policies of all agents during the learning phase. Subsequently, each agent's critic concatenates all state-actions as input and, utilizing the local reward, calculates the corresponding Q-value. The training process involves optimizing for each of the N critics the following loss function:

$$L(\mu_{\theta^i}) = \mathbb{E}_{o^t,a,r,o^{t+1}}[(Q_{\theta^i}(s^t, a_1^t, ..., a_N^t; \mu_{\theta^i}) - y)^2] \qquad (4)$$

with:

$$y = r_i^t + \gamma Q_{\theta^i}(o^t, \widetilde{a}_1^{t+1}, ..., \widetilde{a}_N^{t+1}; \widetilde{\mu}_{\theta^i})|_{\widetilde{a}_j^{t+1}=\widetilde{\pi}(o_j^{t+1})} \qquad (5)$$

where o^t represents the observation of all agents, $\widetilde{\mu}$ is the target critic, and $\widetilde{\pi}$ is the target policy. Consequently, the critic of each agent operates within a stationary environment, requiring access only to local information.

Building upon the strengths of the TD3 algorithm and the need for effective MARL algorithms, the authors of [35] introduced the Multi-Agent TD3 (MATD3) algorithm, which retains the utilization of twin Q-networks to enhance the deterministic policy in a multi-agent context. MATD3 extends this concept to accommodate the interactions and dependencies between multiple agents. Each agent, denoted as agent i, maintains its own actor $\pi_{\phi^i}(o_i;\phi^i)$ with policy weights ϕ^i and observes its local environment.

Expanding on these approaches, the collaborative MARL framework proposed in our work for the quadrotor's attitude control is presented in Figure 4. This architecture aims to enable each agent to understand the collective dynamics and decisions of the entire group. Each of the *Roll*, *Pitch*, and *Yaw* critics considers not only its individual

local observations and actions but also integrates insights from the target policies, actions, and observations of all agents in the system. This process generates action-value functions representing the expected return for each agent. Using these action-value functions, all agents update their policies and make independent decisions. This fosters a cooperative learning process using a Centralized Training, Decentralized Execution (CTDE) mode. It is worth noting that the Centralized Training, Centralized Execution (CTCE) mode can also be employed for this particular MARL application, where the agents are not distributed entities and communication between them can be easily guaranteed. However, the size of the agent and specifically the actor network has to be relatively larger to accommodate centralized execution. This approach essentially converges to a single-agent configuration, where one RL agent generates the three control inputs U_2, U_3, and U_4.

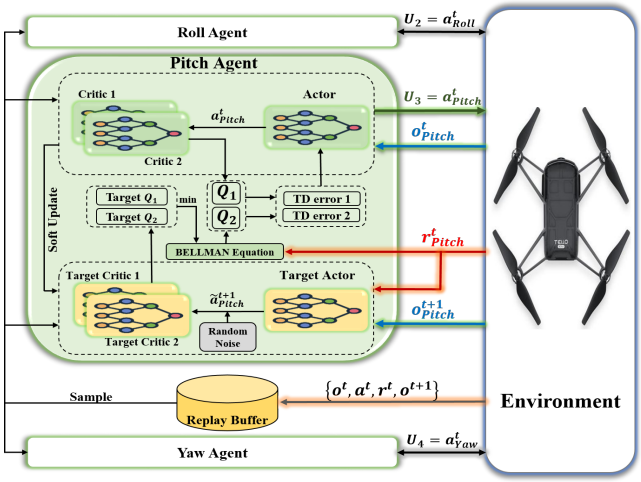

Figure 4. Multi-agent TD3 attitude control system

Algorithm 1 presents the pseudo-code of the MATD3 algorithm tailored to address the quadrotor's attitude control system, where:

First, the actor and critic networks for each attitude agent (*Roll*, *Pitch*, and *Yaw* agents) along with the replay buffer are initialized. At the start of each episode, a random noise for action exploration ϵ_i is generated, and the initial state is obtained, encompassing the concerned orientations, orientation rates, and orientation errors with:

$$o = \{\phi, \dot{\phi}, \phi_{error}, \theta, \dot{\theta}, \theta_{error}, \psi, \dot{\psi}, \psi_{error}\} \tag{6}$$

Subsequently, episode iterations proceed, and the agents select and execute the Roll, Pitch, and Yaw moments $[U_2, U_3, U_4]$ according to their policies.

Following this, each agent receives a reward r_i^t, which provides feedback on the effectiveness of the selected moments, and updates its observations o_i^{t+1}. All actions, rewards, and observations are stored in the replay buffer R to allow the agents to learn from past experiences.

During the training process, each agent randomly samples a small subset from the replay buffer to set the target value of the Q-function to y_i, using the minimum Q-value from the critic networks. Thereafter, the parameters θ_1^i and θ_2^i of the critic networks are updated for the selected samples, along with the policy parameters ϕ^i of the actor networks, by maximizing the gradient ascent. Following the update of the critic and actor networks by each agent, the target network parameters $\theta^{i_{target}}_{n=1,2}$ and $\phi^{i_{target}}$ are adjusted to ensure learning stability by restricting the rate of update for the target values.

C. Neural Network Structures

To ensure a fair comparison and maintain consistency in our MARL setup, all the employed agents are designed to be identical, with critic networks that comprise two pathways. The first path takes the state as input and incorporates three feed-forward hidden layers, each consisting of 128, 128, and 64 neurons, respectively. The output from this pathway is then merged with the action pathway, which contains a single hidden layer comprising 64 neurons. This integrated structure is responsible for generating the Q-value specific to each agent, as highlighted in Figure 5.

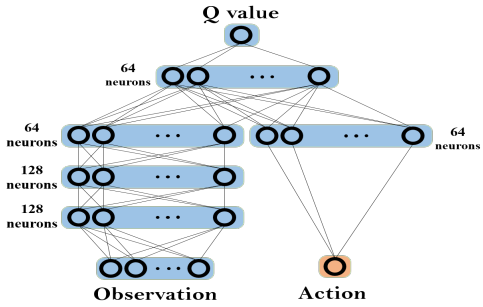

Figure 5. Critic network

As presented in Figure 6, the actor network for each agent is constructed with four feed-forward hidden layers, each containing 10 neurons. With the exception of the action output layer that employs a *Sigmoid* activation function, all the layers use the *Rectified Linear Unit* (ReLU).

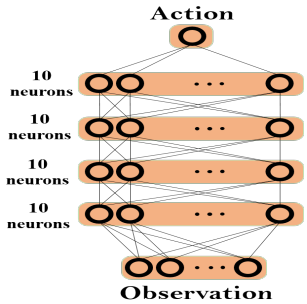

Figure 6. Actor network

Algorithm 1 MATD3 algorithm for the quadrotor attitude control system

for *each agent* i **do**
  Set actor network π_{ϕ^i} *and two critic networks* $Q_{\theta_1^i}, Q_{\theta_2^i}$ *with random weights* $\phi^i, \theta_1^i, \theta_2^i$ *respectively;*
  Set target networks $\phi^{i_{target}} \leftarrow \phi^i$, $\theta_1^{i_{target}} \leftarrow \theta_1^i$, $\theta_2^{i_{target}} \leftarrow \theta_2^i$;
  end for
Initialize the replay buffer R;
for $episode = 1, Max$ **do**
  Generate random noise: $\varepsilon_i \sim clip(\mathcal{N}(0, \widetilde{\sigma}), a_{i_{min}}, a_{i_{max}})$;
  Acquire original observations $o = \{o_{Roll}, o_{Pitch}, o_{Yaw}\}$;
  for $t = 1, max_{timesteps}$ **do**
    Select the control input from the action space of each agent: $a_i^t = \pi_{\phi^i}(o_i^t) + \varepsilon_i$;
    Execute the actions that represent the roll, pitch and yaw moments respectively:
      $a(t) = [a_1^t, a_2^t, a_3^t] = [U_2(t), U_3(t), U_4(t)]$
    Calculate the rewards r_i^t, *and observe* o_i^{t+1};
    Store transition $(o^t, a_1^t, a_2^t, a_3^t, r^t, o^{t+1})$ *in R;*
    Update the state $s \leftarrow s^{t+1}$;
    for *agent* $i = 1$ *to* 3 **do**
      Sample a random mini-batch of N transitions from R;
      Calculate target critic Q value:
        $y_i^t \leftarrow r_i^t + \gamma min[\ Q_{\theta_n^{i_{target}}}(o_{Roll}^{t+1}, o_{Pitch}^{t+1}, o_{Yaw}^{t+1}, \widetilde{a}_1^{t+1}, \widetilde{a}_2^{t+1}, \widetilde{a}_3^{t+1})]_{n=1,2}$
      Update critics:
        $\theta_1^i \leftarrow min_{\theta_1^i}\ \frac{1}{N} \sum [y_i^t - Q_{\theta_1^i}(o_{Roll}^t, o_{Pitch}^t, o_{Yaw}^t, \widetilde{a}_1^t, \widetilde{a}_2^t, \widetilde{a}_3^t)]^2$
        $\theta_2^i \leftarrow min_{\theta_2^i}\ \frac{1}{N} \sum [y_i^t - Q_{\theta_2^i}(o_{Roll}^t, o_{Pitch}^t, o_{Yaw}^t, \widetilde{a}_1^t, \widetilde{a}_2^t, \widetilde{a}_3^t)]^2$
      If t *mod* d **then**
        Update ϕ *using the DPG:*
        $\nabla_{\phi^i} J(\phi^i) = \frac{1}{N} \sum \nabla_{\phi_i} \pi_{\phi_i}(o_i^t) \nabla_{a_i} Q_{\theta^i}(o_{Roll}^t, o_{Pitch}^t, o_{Yaw}^t, a_1^t, a_2^t, a_3^t)|_{a_i^t = \pi_{\phi^i}(o_i^t)}$
        Update the target networks:
        $\theta_n^{i_{target}} \leftarrow \tau\theta_n^i + (1-\tau)\theta_n^{i_{target}}|_{n=1,2}$
        $\phi^{i_{target}} \leftarrow \tau\phi^i + (1-\tau)\phi^{i_{target}}$
      end If
    end for
  end for
end for

D. Reward Functions

Formulating the reward function is a crucial and unique element of reinforcement learning. It must be designed meticulously to provide clear and consistent feedback on the agent's actions, ensuring that the optimal policy aligns with the desired behavior.

After experimenting with different reward functions and designs, we developed a strategy to guide the quadrotor's attitude through collaborative reward collection. This strategy considers a global reward for overall control cooperation, defined as:

$$r_{global} = -\alpha \sqrt{e_\phi^2 + e_\theta^2 + e_\psi^2} \tag{7}$$

Each agent's goal is to minimize its corresponding attitude error by taking into account the global reward, the corresponding attitude tracking error, and its derivative, as follows:

$$r_i = r_{global} - k.sign(e_i.\dot{e_i})|_{i=\phi,\theta,\psi} \tag{8}$$

With k and α as positive weights, motivating the agents to minimize both their specific and global tracking errors.

4. Results and Discussion

This section presents and analyzes the outcomes of the proposed approach. It assesses the efficacy of the top-performing agents in tracking various trajectories with different levels of complexity and dynamics. Additionally, we conduct a comparative evaluation against conventional single-agent RL, PID, and Backstepping controllers to highlight the adaptability, stability, and precision of our MARL approach. The simulation environment used for this study was built using MATLAB software, and the training process was executed on a system equipped with an 11th Gen Intel® Core™ i7-1165G7 processor, operating at 2.80 GHz.

TABLE I. Quadrotor's physical parameters.

Symbol	Description	Value
m	The quadrotor's mass	0.8 kg
g	The gravitational acceleration	9.81 $m \cdot s^{-2}$
l	The quadrotor's helf length	0.06 m
I_{xx}, I_{yy}, I_{zz}	The moment of inertia	(0.0097, 0.0097, 0.017) $kg \cdot m^2$
b	The thrust factor	1 $N \cdot s^2$
d	The drag factor	0.08 $N \cdot m \cdot s$

For the training setup, the objective is to stabilize a quadrotor system (with the nominal parameters summarized in Table I) from any given configuration within physical limits using Multi-Agent (MA) based controllers. The desired state is defined using reference Euler angles φ_{Ref}, θ_{Ref}, and ψ_{Ref} as shown in Figure 3. Three agents, trained through the approach detailed in section 3, ensure the quadrotor adheres to the orientation commands set by the high-level controller and produce accurate control inputs U_1, U_2, and U_3.

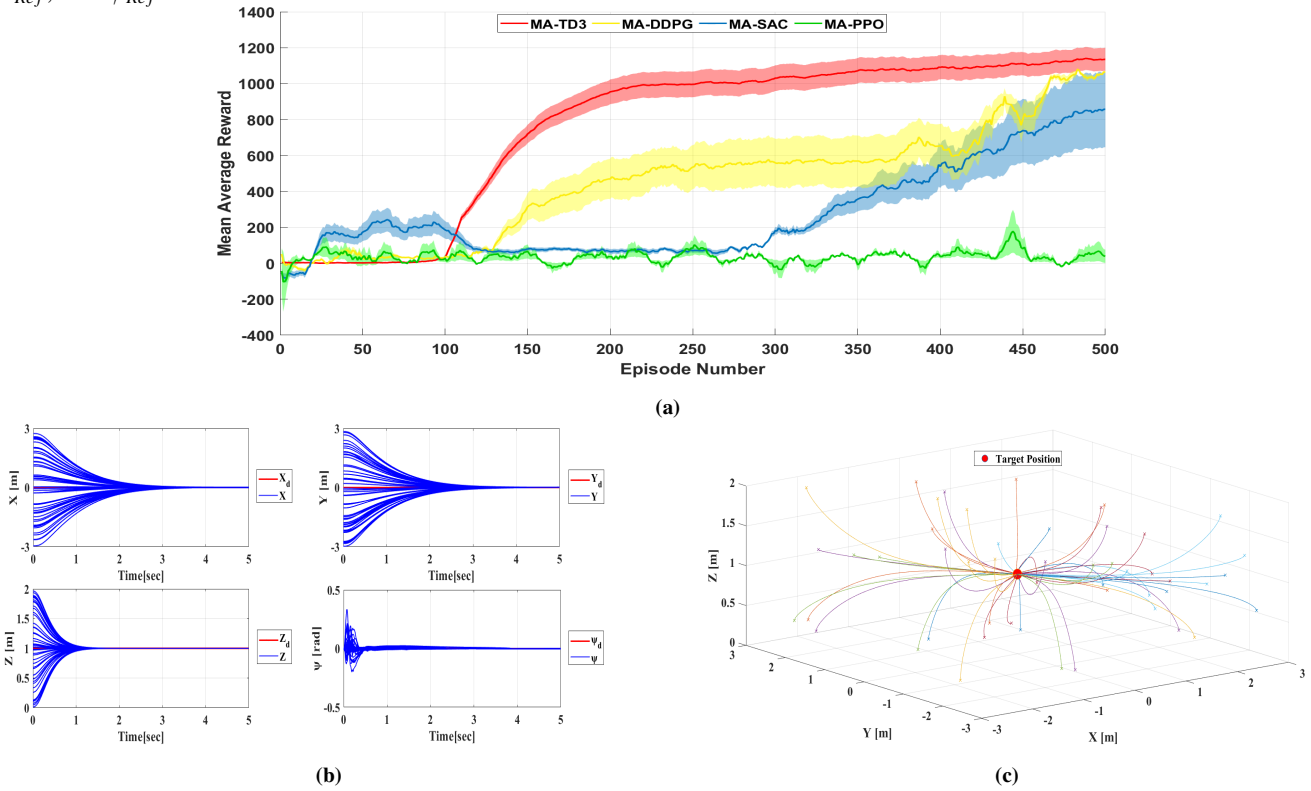

Figure 7. **(a)** Training sessions of the proposed MARL attitude control system, **(b)** MA-TD3 best agents' performance for stabilization task (50 random position and orientation initializations), **(c)** 3D goal tracking from 50 random initializations

The training results summarized in Figure 7.a, shed light on the convergence and learning capabilities of various algorithms using the advocated framework. We conducted a performance comparison of the four algorithms MATD3, MADDPG, MASAC, and MAPPO, aimed at providing a comprehensive analysis of different approaches, showcasing the mean average reward collected by the *Roll*, *Pitch* and *Yaw* agents over 500 training episodes.

- The MATD3 algorithm emerged as the most efficient and effective, demonstrating both speed and stability in training. The resulting policies exhibited rapid convergence and consistently achieved high rewards throughout the training sessions. The validation tests of the obtained policies are shown in Figure 7.b and Figure 7.c, where the trained controllers successfully stabilized the system across 50 random position and orientation initializations.

- MADDPG exhibited a two-phase training behavior, initially achieving stable learning with low collected rewards before eventually exploring the action space more and approaching the reward levels of MATD3. While MADDPG's performance was inferior to MATD3, the delayed convergence suggests potential for improvement with longer training durations or additional fine-tuning.

- The MASAC demonstrated slower convergence but ultimately achieved higher rewards by the end of the training session. This behavior may be attributed to its emphasis on entropy regularization, which encourages exploration and prevents premature convergence to suboptimal policies.

- While the other algorithms successfully converged to collaborative policies, MAPPO struggled to achieve effective cooperation for handling quadrotor attitude control. Despite efforts to fine-tune hyperparameters and exploration strategies, MAPPO's performance remained subpar compared to MATD3, MADDPG, and MASAC. These challenges underscore the importance of selecting the right algorithm, especially for complex control tasks such as quadrotor systems.

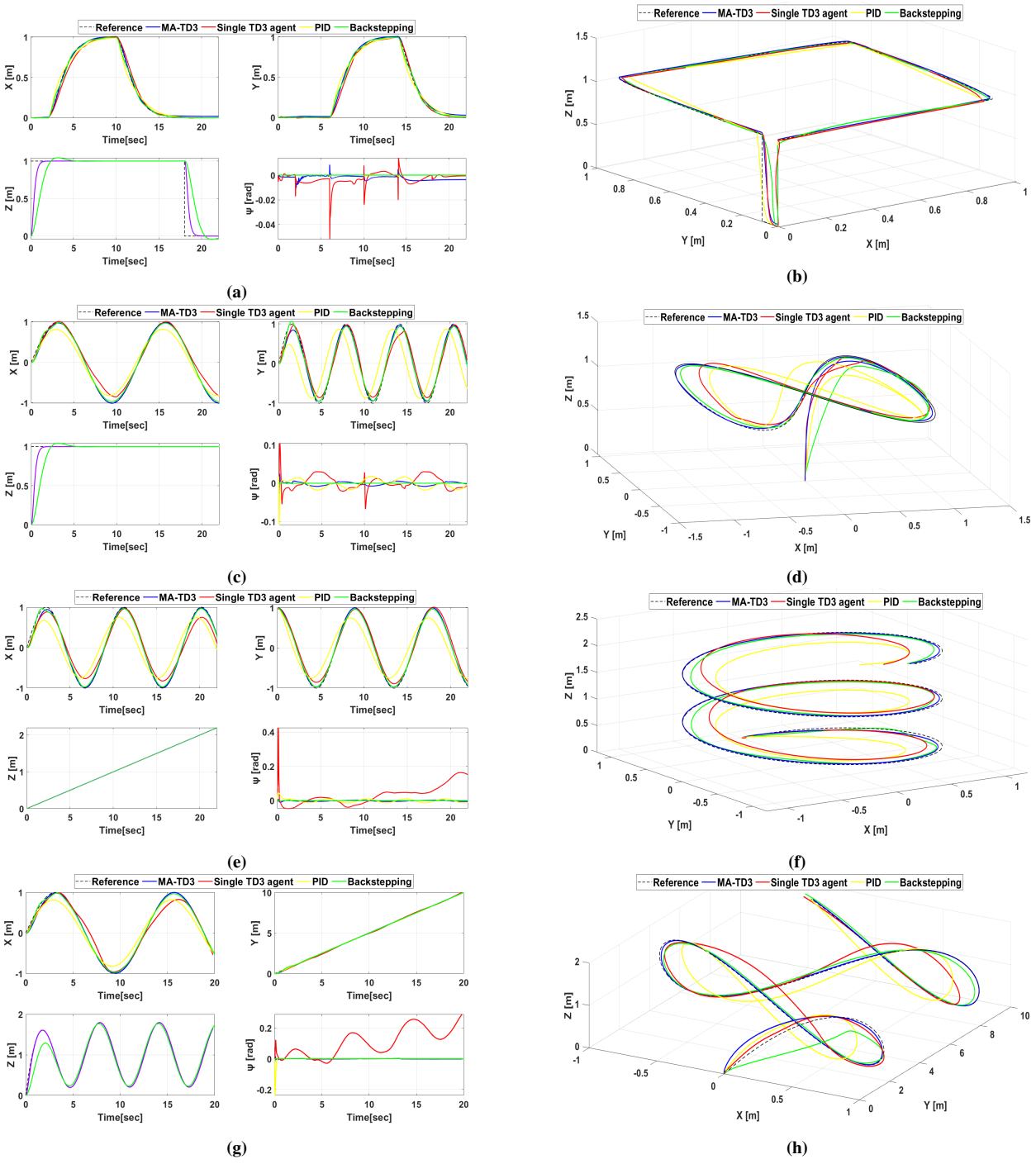

Figure 8. **(a)** Square trajectory position and orientation tracking, **(b)** 3D square trajectory, **(c)** Lemniscate trajectory position and orientation tracking, **(d)** 3D lemniscate trajectory, **(e)** Ellipsoid trajectory position and orientation tracking, **(f)** 3D ellipsoid trajectory, **(g)** Acrobatic trajectory position and orientation tracking, **(h)** 3D acrobatic trajectory

The efficiency of the MARL-trained agents is assessed using the Mean Squared Error (MSE) metric, which quantifies the average squared difference between predicted and actual values across all data points. MSE is particularly advantageous for evaluating our controller's performance as it provides a robust measure of accuracy and consistency in trajectory tracking. By squaring the errors, MSE emphasizes larger discrepancies, thereby sensitively detecting deviations from desired paths across various predefined trajectories. In Table II, the lowest MSE recorded for each path is

TABLE II. Mean squared error (MSE) on the paths displayed in Figure 8.

Trajectory	Controller	X	Y	Z	ψ
Square	MA-TD3	0.000171	0.0003026	0.02772	0.7982×10^{-5}
	Single TD3 agent	0.001385	0.0007229	0.02772	2.913×10^{-5}
	PID	0.001313	0.001313	0.02772	2.138×10^{-8}
	Backstepping	0.0002831	0.0003845	0.07093	4.418×10^{-40}
Lemniscate	MA-TD3	0.001046	0.004755	0.0134	2.247×10^{-5}
	Single TD3 agent	0.01358	0.01409	0.0134	33.59×10^{-5}
	PID	0.02287	0.2863	0.0134	0.0001551
	Backstepping	0.001209	0.004823	0.0343	2.18×10^{-42}
Ellipsoid	MA-TD3	0.002068	0.0003255	9.226×10^{-9}	1.256×10^{-5}
	Single TD3 agent	0.02165	0.006435	9.226×10^{-9}	0.00444
	PID	0.08643	0.07427	9.226×10^{-9}	0.0001013
	Backstepping	0.002616	0.001351	7.534×10^{-8}	1.111×10^{-42}
Acrobatic	MA-TD3	0.001194	0.001151	0.0005283	3.038×10^{-6}
	Single TD3 agent	0.0142	0.00359	0.0005283	0.01863
	PID	0.02408	0.0003556	0.0005283	0.0002218
	Backstepping	0.001221	0.005915	0.02669	5.986×10^{-42}

highlighted in green, indicating superior performance. The Backstepping controller's MSE for the Z position differs due to its integration into the overall control system, while other approaches operate at the low-level control system.

The validation and comparison process employed in our study involves navigating diverse paths, including square, lemniscate, ellipsoid, and acrobatic trajectories (as depicted in Figure 8), chosen for their varying complexity and dynamics. Additionally, evaluating our MARL approach through these trajectories offers insights into its adaptability and generalization capabilities, considering the training phase was limited to stabilizing from random configurations (refer to Figure 7).

The results demonstrated that the MATD3 controller consistently outperformed the other methods across all trajectories, showcasing its superior stability and robustness in dynamically adjusting policies to varying flight conditions. Its ability to maintain precise tracking, particularly in challenging trajectories such as lemniscate, ellipsoid, and acrobatic paths, highlights its advanced responsiveness and adaptability in complex scenarios.

The Backstepping controller effectively stabilized the yaw angle and demonstrated robust control in simpler trajectories such as the square path. However, it showed limitations in dynamic adaptability compared to MATD3, suggesting that while reliable, its responsiveness and adaptability in more complex maneuvers may be suboptimal for scenarios requiring agile and flexible responses.

The PID controller, while straightforward and easy to implement, required meticulous parameter tuning to achieve enhanced performance, particularly in more dynamic and demanding trajectories.

The single TD3 agent showed promising performance in simpler trajectories aligned with its learning configuration but faced challenges in generalizing to more diverse and complex scenarios. This highlights the importance of further training and refinement to enhance its adaptability and robustness, addressing stability concerns in varied environments.

To further emphasize the advantages of collaborative intelligence and the proposed MARL framework, particularly in contrast to single-agent RL approaches, we conducted a comparative analysis of parameter sizes between the two configurations:

- The actor neural network utilized in this study comprises four hidden layers with 128 neurons each, resulting in a total of 51,587 parameters for the single TD3 agent.
- The proposed MARL architecture consists of three agents, each with 441 parameters.

The MARL system with three agents, each containing 441 parameters, represents a significant 97.44% reduction in parameter size compared to a single agent with 51,587 parameters, while still achieving superior performance. This calculation underscores the reduction in parameter complexity achieved by employing multiple simpler agents within the MARL framework. This reduction not only enhances computational efficiency but also augments the system's robustness, adaptability, and scalability in practical implementations, particularly in challenging and dynamic environments such as quadrotor control systems.

It is worth noting that a significant challenge of the proposed MARL approach, which is also pertinent to all RL techniques, is the hyperparameter optimization and tuning process. This process can be complex and time-consuming, as identifying the optimal set of hyperparameters is crucial for achieving optimal performance.

TABLE III. MATD3 hyper-parameters.

Hyper-parameter	Value
Time step	0.01
Replay buffer size	10^6
Actor network's learning rate	10^{-4}
Critic network's learning rate	10^{-3}
Discount factor	0.98
Minibatch size	100
Noise variance	0.15
Variance Decay Rate	10^{-5}

5. Conclusion and Future Work

While recent studies primarily focus on collaborative control strategies among multiple quadrotors using Multi-Agent Reinforcement Learning techniques for tasks such as formation flying, path planning, collision, and obstacle avoidance, this work introduces a novel MARL approach for enhanced quadrotor attitude control. By employing MARL algorithms including MATD3, MADDPG, MASAC, and MAPPO, we investigate their adaptability and performance compared to single-agent RL and various linear and non-linear controllers. Through extensive training and validation phases, we observed that while all controllers performed similarly in scenarios resembling the training configurations, the MARL approach demonstrated superior capabilities and robustness when navigating complex trajectories such as lemniscate and ellipsoid paths. Specifically, MATD3 exhibited rapid and stable learning, outperforming other MARL algorithms and benchmark methodologies in adaptability and tracking performance on paths featuring high-speed maneuvers and rapid altitude changes. Furthermore, the proposed MARL framework demonstrates heightened efficiency and adaptability, marked by a significant 97.44% reduction in parameter size compared to single-agent RL. This underscores the potential efficacy of MARL techniques in real-world quadrotor applications, particularly where precise and adaptive control is imperative and computational resources are constrained.

In future work, we aim to evaluate the performance of the proposed approach in dynamic and unpredictable environments to assess its robustness properties. Addressing the challenge of hyperparameter optimization and tuning, crucial for achieving optimal performance in RL techniques, remains a key area for future research. Additionally, we plan to conduct experiments using the developed control technique on real-world quadrotor platforms, particularly on low-cost quadrotors with constrained computational resources such as the Parrot Mambo Mini-drone. This will validate the effectiveness and performance of our approach in practical applications.

References

[1] T. Y. Trad, K. Choutri, M. Lagha, R. Fareh, and M. Bettayeb, "Simulated annealing-deep deterministic policy gradient algorithm for quadrotor attitude control," in *2023 Advances in Science and Engineering Technology International Conferences (ASET)*. IEEE, 2023, pp. 1–6.

[2] S. Wang, A. Polyakov, and G. Zheng, "Quadrotor stabilization under time and space constraints using implicit pid controller," *Journal of the Franklin Institute*, vol. 359, no. 4, pp. 1505–1530, 2022.

[3] R. Thusoo, S. Jain, and S. Bangia, "Path planning of quadrotor using a* and lqr," in *Recent Developments in Electrical and Electronics Engineering: Select Proceedings of ICRDEEE 2022*. Springer, 2023, pp. 227–238.

[4] M. Okasha, J. Kralev, and M. Islam, "Design and experimental comparison of pid, lqr and mpc stabilizing controllers for parrot mambo mini-drone," *Aerospace*, vol. 9, no. 6, p. 298, 2022.

[5] A.-W. Saif, K. B. Gaufan, S. El-Ferik, and M. Al Dhaifallah, "Fractional order sliding mode control of quadrotor based on fractional order model," *IEEE Access*, 2023.

[6] J. Wang, K. A. Alattas, Y. Bouteraa, O. Mofid, and S. Mobayen, "Adaptive finite-time backstepping control tracker for quadrotor uav with model uncertainty and external disturbance," *Aerospace Science and Technology*, vol. 133, p. 108088, 2023.

[7] M. B. Artuc and I. Bayezit, "Robust adaptive quadrotor position tracking control for uncertain and fault conditions," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, p. 09544100231181869, 2023.

[8] F. W. Alsaade, H. Jahanshahi, Q. Yao, M. S. Al-zahrani, and A. S. Alzahrani, "A new neural network-based optimal mixed h2/h control for a modified unmanned aerial vehicle subject to control input constraints," *Advances in Space Research*, vol. 71, no. 9, pp. 3631–3643, 2023.

[9] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7512–7519.

[10] B. Yu and T. Lee, "Multi-agent reinforcement learning for the low-level control of a quadrotor uav," *arXiv preprint arXiv:2311.06144*, 2023.

[11] Y. Z. Jembre, Y. W. Nugroho, M. T. R. Khan, M. Attique, R. Paul, S. H. A. Shah, and B. Kim, "Evaluation of reinforcement and deep learning algorithms in controlling unmanned aerial vehicles," *Applied Sciences*, vol. 11, no. 16, p. 7240, 2021.

[12] C.-H. Pi, K.-C. Hu, S. Cheng, and I.-C. Wu, "Low-level autonomous control and tracking of quadrotor using reinforcement learning," *Control Engineering Practice*, vol. 95, p. 104222, 2020.

[13] B. Rubí, B. Morcego, and R. Pérez, "Deep reinforcement learning for quadrotor path following with adaptive velocity," *Autonomous Robots*, vol. 45, no. 1, pp. 119–134, 2021.

[14] V. Kashyap and R. Vepa, "Reinforcement learning based linear quadratic regulator for the control of a quadcopter," in *AIAA SCITECH 2023 Forum*, 2023, p. 0014.

[15] B. Jiang, B. Li, W. Zhou, L.-Y. Lo, C.-K. Chen, and C.-Y. Wen, "Neural network based model predictive control for a quadrotor uav," *Aerospace*, vol. 9, no. 8, p. 460, 2022.

[16] J. Orr and A. Dutta, "Multi-agent deep reinforcement learning for multi-robot applications: a survey," *Sensors*, vol. 23, no. 7, p. 3625, 2023.

[17] A. Oroojlooy and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *Applied Intelligence*, vol. 53, no. 11, pp. 13 677–13 722, 2023.

[18] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, pp. 1–49, 2022.

[19] L. Canese, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, and S. Spanò, "Multi-agent reinforcement learning: A review of challenges and applications," *Applied Sciences*, vol. 11, no. 11, p. 4948, 2021.

[20] S. He, S. Han, S. Su, S. Han, S. Zou, and F. Miao, "Robust multi-agent reinforcement learning with state uncertainty," *arXiv preprint arXiv:2307.16212*, 2023.

[21] P. Li, J. Hao, H. Tang, Y. Zheng, and X. Fu, "Race: improve multi-agent reinforcement learning with representation asymmetry and collaborative evolution," in *International Conference on Machine Learning*. PMLR, 2023, pp. 19 490–19 503.

[22] Y. Zhang and M. M. Zavlanos, "Cooperative multi-agent reinforcement learning with partial observations," *IEEE Transactions on Automatic Control*, 2023.

[23] M. Zhou, Z. Wan, H. Wang, M. Wen, R. Wu, Y. Wen, Y. Yang, Y. Yu, J. Wang, and W. Zhang, "Malib: A parallel framework for population-based multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 24, no. 150, pp. 1–12, 2023.

[24] A. Mondal, D. Mishra, G. Prasad, G. C. Alexandropoulos, A. Alnahari, and R. Jantti, "Multi-agent reinforcement learning for offloading cellular communications with cooperating uavs," *arXiv preprint arXiv:2402.02957*, 2024.

[25] R. Sabogu-Sumah, K. A. Opare, J. D. D. Gadze, J. J. Kponyo, A.-R. Ahmed, E. Fianko *et al.*, "Optimal coverage enhancement for multiple uavs using multi-agent learning technique," *International Journal of Computing and Digital Systems*, vol. 13, no. 1, pp. 1–1, 2023.

[26] Z. Feng, D. Wu, M. Huang, and C. Yuen, "Graph attention-based reinforcement learning for trajectory design and resource assignment in multi-uav assisted communication," *arXiv preprint arXiv:2401.17880*, 2024.

[27] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning-based trajectory planning for multi-uav assisted mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 73–84, 2020.

[28] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for uav networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 729–743, 2019.

[29] Z. Wenhong, L. Jie, L. Zhihong, and S. Lincheng, "Improving multi-target cooperative tracking guidance for uav swarms using multi-agent reinforcement learning," *Chinese Journal of Aeronautics*, vol. 35, no. 7, pp. 100–112, 2022.

[30] Z. Jiandong, Y. Qiming, S. Guoqing, L. Yi, and W. Yong, "Uav cooperative air combat maneuver decision based on multi-agent reinforcement learning," *Journal of Systems Engineering and Electronics*, vol. 32, no. 6, pp. 1421–1438, 2021.

[31] Y.-J. Chen, D.-K. Chang, and C. Zhang, "Autonomous tracking using a swarm of uavs: A constrained multi-agent reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13 702–13 717, 2020.

[32] H. Han, J. Cheng, Z. Xi, and B. Yao, "Cascade flight control of quadrotors based on deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 134–11 141, 2022.

[33] R. Benotsmane and J. Vásárhelyi, "Towards optimization of energy consumption of tello quad-rotor with mpc model implementation," *Energies*, vol. 15, no. 23, p. 9207, 2022.

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

[35] F. Zhang, J. Li, and Z. Li, "A td3-based multi-agent deep reinforcement learning method in mixed cooperation-competition environment," *Neurocomputing*, vol. 411, pp. 206–215, 2020.