Modern artificial intelligence technics for unmanned aerial vehicles path planning and control

Yasmine Zamoum[1], Karim Baiche[1], Youcef Benkeddad[2], Brahim Bouzida[2], Razika Boushaki[3]
[1]Laboratoire d'Ingénierie des Systèmes et Télécommunications, Ingénierie de Génie Elèctrique, Faculté de Technologie, Université M'hamed Bougara de Boumerdes, Boumerdes, Algérie
[2]Institut de Génie Electrique et Electronique, Université M'hamed Bougara de Boumerdes, Boumerdes, Algérie
[3]Laboratoire d'Automatique Appliquée, Département d'Elèctotechnique et Automatique, Institut de Génie Electrique et Electronique, Université M'hamed Bougara de Boumerdes, Boumerdes, Algérie

Article Info

Article history:

Received Feb 22, 2024
Revised Aug 28, 2024
Accepted Sep 4, 2024

Keywords:

Deep Q-learning
Dyna Q-learning
Fuzzy logic
Quadrotor
Unmanned aerial vehicle path planning

ABSTRACT

Unmanned aerial vehicles (UAVs) require effective path planning algorithms to navigate through complex environments. This study investigates the application of Deep Q-learning and Dyna Q-learning methods for UAV path planning and incorporates fuzzy logic for enhanced control. Deep Q-learning, a reinforcement learning technique, employs a deep neural network to approximate Q-values, allowing the UAV to improve its path planning capabilities by maximizing cumulative rewards. Conversely, Dyna Q-learning leverages simulated scenarios to update Q-values, refining the UAV's decision-making process and adaptability to dynamic environments. Additionally, fuzzy logic control is integrated to manage UAV movements along the planned path. This control system uses linguistic variables and fuzzy rules to handle uncertainties and imprecise information, enabling real-time adjustments to speed, altitude, and heading for accurate path following and obstacle avoidance. The research evaluates the effectiveness of these methods individually, with a focus on model-free learning in a gradual training approach, and compares their performance in terms of path planning accuracy, adaptability, and obstacle avoidance. The paper contributes to a deeper understanding of UAV path planning techniques and their practical applications in various scenarios.

Corresponding Author:

Yasmine Zamoum
Laboratoire d'Ingénierie des Systèmes et Télécommunications, Ingénierie de Génie Elèctrique
Faculté de Technologie, Université M'hamed Bougara de Boumerdes
35000 Boumerdes, Algeria
Email: y.zamoum@univ-boumerdes.dz

1. INTRODUCTION

Deep reinforcement learning has become the state-of-the-art for many tasks in recent years [1]. Path planning is very important for facilitating effective and secure navigation, a basic overview of obstacle avoidance based on reinforcement learning was provided [2]. Path planning's intelligent algorithms, such as Deep Q-learning and Dyna Q-learning, enhances the path planning capabilities of autonomous systems. By integrating a fuzzy logic control System can provide robust control and improve the system's ability to follow the generated path.

The objective of many autonomous drone racing, is to move as quickly as possible through a sequence of checkpoints [3], so this study aims to address the research problem of optimizing path planning using Deep Q-learning, Dyna Q-learning, and using both standard and adaptive fuzzy logic control systems.

The primary objective is to compare the performance of these two reinforcement learning algorithms in generating optimal paths for autonomous systems. Additionally, we seek to investigate the effectiveness of integrating a fuzzy logic control System and compare it to its more advanced version “adaptive fuzzy logic” to improve the system's ability to accurately follow the generated paths.

The Q-learning algorithm is used to overestimate action values under specific assumptions [4], and suffer from some limitations when dealing with high-dimensional or continuous inputs [5]. In this research, we solved the problem by implementing an obstacle avoidance and path planning algorithm for unmanned vehicles using the Deep Q-learning and Dyna-Q reinforcement learning algorithms [6]. The investigation will involve simulating various scenarios with dynamic obstacles and varying environments to evaluate the algorithms' performance. Furthermore, the study will explore the integration of both standard and adaptive fuzzy logic control systems as a means of enhancing the autonomous system's path following capabilities.

The Deep Q-learning offers to robotics a set of tools for the design [7]. To implement and evaluate the Deep Q-Learning and Dyna Q-learning algorithms in simulated environments, and design a fuzzy logic Control System to enable accurate path following the generated paths. To compare the performance of Deep Q-learning and Dyna Q-learning algorithms in terms of path planning efficiency to assess the effectiveness of the integrated fuzzy logic control system in improving path following accuracy. This study will utilize a simulated environment to collect data for evaluation. The dataset will include information on the environment characteristics, obstacles, generated paths, and the performance metrics of the Deep Q-learning, Dyna Q-learning, standard, and adaptive fuzzy logic control systems. Through this research, we aim to contribute to the advancement of autonomous systems' path planning capabilities by exploring and comparing the performance of Deep Q-learning and a slightly modified Dyna Q-learning algorithms. Furthermore, integrating fuzzy logic control systems can provide insights into enhancing path.

2. METHOD

Deep Q-learning algorithms and Dyna Q-learning algorithms for path planning are explained in this section, along with fuzzy logic and proportional integral derivative (PID) adaptive fuzzy logic for controlling the quadrotor.

2.1. Path planning methods

2.1.1. Deep Q-learning algorithm

The unmanned aerial vehicles (UAV's) 3-D trajectories have a remarkable effect on the performance of networks [8]. However, a number of intelligent algorithms have been put out to address the obstacle avoidance problem as a result of the quick development of computer technology and hardware [9]. The path planning, maximizes the quadcopter's operational potential while simultaneously ensuring the safety of its surroundings [10]. A carefully thought-out path reduces the quadcopter's energy consumption, increases its flight time, and improves its agility, allowing it to execute difficult maneuvers with accuracy.

A deep neural network, known as the Q-network, is employed in Deep Q-learning to approximate the Q-values. The Bellman equation, which encapsulates the relationship between the present state-action pair and the anticipated future rewards, is used to compute the target Q-values, and the network is trained to minimize the difference between these values and the predicted Q-values. The Bellman equation is given by (1):

$$Q(S,A) \leftarrow R + \gamma max_a Q(s',a) \quad (1)$$

An essential reinforcement learning approach is called Deep Q-learning, which is very important for training a deep neural network, or deep Q-network, to approximate the well-known Q-function [11]. The approach makes use of an experience replay buffer to train the Q-network. As it interacts with its surroundings during training, the agent records observed state-action-reward-next state transit ions in the replay buffer. The approach samples from a batch of transitions from the replay buffer and utilizes them to update the network's weights rather than updating the Q-network after each interaction. With this strategy, the learning process is stabilized and the correlations between subsequent updates are decreased.

Target network is another method used by the Deep Q-learning algorithm [12]. The weights from the primary network are periodically changed on this distinct replica of the Q-network. During training, the target network computes the target Q-values while the main network forecasts the Q-values. The technique overcomes the problem of the target values fluctuating continuously during learning, which can cause instability, by employing a distinct target network. In Deep Q-learning algorithm, the agent selects the highest Q-value and the random actions with a probability of epsilon exploration [13]. The agent's behavior is often gradually changed from exploration to exploitation by annealing the exploration rate epsilon over time.

The Deep Q-learning is a potent path planning system that fuses deep neural networks and reinforcement learning but in order to properly apply this technique in scenarios that resemble the complexity of the actual world, agents must overcome a challenging task [12]. Deep Q-learning achieves consistent and effective learning by using a target network, experience replay, and a neural network to approximate Q-values. Its use in path planning has produced encouraging results in a number of fields. The algorithm is [14]:

```
Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function Q with random weights
For episode = 1, $M$ do
  Initialise state $s_t$
    For $t = 1, T$ do
      With probability $\varepsilon$ select a random action $a_t$
      otherwise select $a_t = \max_a Q^*(s_t, a; \Theta)$
      execute action $a_t$ and observe reward $r_t$ and state $s_{t+1}$
      Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$
      Set $s_{t+1} = s_t$
      Sample random minibatch of transitions $(s_t, a_t, r_t, s_{t+1})$ from $\mathcal{D}$
      Set $yj = \begin{cases} rj & \text{for terminal } st+1 \\ \mathrm{rj} + \gamma \max_{a'} Q(st+1, a'; \Theta) & \text{for non-terminal } st+1 \end{cases}$

      Perform a gradient descent step on $(y_j - Q(s_t, a_j; \Theta))^2$
    end for
end for
```

a. Experimental setup

1) Simulation environment description

A 3D grid-based representation serves as the simulation environment for training the Deep Q-learning algorithm for quadcopter path planning. Utilizing an occupancy map (omap3D) that simulates the environment's barriers, it is put into practice. The grid's width, length, and height are supplied, and together they specify the environment's size. Because the occupancy map is generated randomly, it accurately depicts the environment's barriers for the agent to avoid.

The objective of the quadcopter agent is to arrive at the goal posture (goalPose), which is a predetermined target position. The goal stance depicts the place in the environment that is intended to be reached. The agent must develop the ability to choose the best routes from arbitrary starting points to the desired pose while avoiding collisions with the obstacles.

2) Training case studies and performance indicators

The agent interacts with the environment frequently throughout a number of episodes in the training settings. In each episode, the agent begins in a random location within the environment and moves through a series of steps to find its way to the desired pose. The agent's objective is to efficiently complete the goal pose while avoiding obstacles in order to maximize its cumulative reward.

Each episode features the agent acting in accordance with the learnt Q-values and the current condition. In order to gain experience and learn from the effects of its actions, the agent initially explores the world by acting arbitrarily. The agent eventually moves toward using the learnt Q-values to inform decisions as the training goes on. Several performance measures are employed to assess the effectiveness of the Deep Q-learning system for quadcopter path planning:

b. Episode reward

The agent's overall reward in each episode reveals if it was able to effectively navigate to the goal posture. A greater episode reward indicates that the agent is developing efficient methods for navigating barriers and getting to the destination. During training and evaluation, the agent's behavior is greatly influenced by the reward function. Each action made by the agent is given a numerical value, reflecting its usefulness or desirability in attaining the intended goal. The reward function affects the agent's decision-making in the context of quadcopter path planning by giving feedback on the effectiveness of its actions.

The reward function in the sample code consists of a number of parts that together make up the overall reward:

– Goal reached: when the agent reaches the goal pose, a positive reward of 100 is assigned. This encourages the agent to prioritize reaching the goal as it signifies successful completion of the task.
– Obstacle collision: if the agent collides with an obstacle, a negative reward of -100 is given. This penalizes the agent for making unsafe or invalid moves, discouraging it from colliding with obstacles.

- Map boundary violation: if the agent moves outside the boundaries of the map, a negative reward of -10 is assigned. This penalizes the agent for going beyond the allowed limits of the environment and encourages it to stay within the defined map boundaries.
- Time penalty: for each step taken by the agent, a small negative reward of -1 is given. This encourages the agent to find efficient paths and reach the goal in the minimum number of steps.

When these incentive elements are combined, the agent is guided toward safe and effective navigation while avoiding obstacles and completing the goal posture. The reward function influences the agent's behavior and directs it toward learning the best path planning strategies by allocating positive and negative incentives based on desired and unwanted activities, respectively. It's vital to remember that the reward function's design and tweaking can have a big impact on how well the agent learns and performs. The balance of rewards for the various components should be carefully considered in order to promote desired behaviors and deter undesirable ones. Additionally, domain-specific information and experience can be used to improve and tailor the reward function to meet the needs and restrictions of certain applications.

1) Number of steps

The agent's path planning effectiveness is shown by the number of steps it takes in each episode. A decreasing number of steps shows that the agent is improving its navigation strategy by learning to take more direct routes to the goal posture. Training time: to assess the learning algorithm's computational effectiveness, the amount of time spent training the deep Q-network is measured. A quicker training time indicates that the algorithm can learn quickly and converge to the best solutions in a fair amount of time. These performance indicators give information about the agent's path planning efficiency and learning progress in the simulated environment. Figures 1 and 2 represent successively a schematic depiction of deep reinforcement learning and simple schematic of Q-Learning: Q-Table.

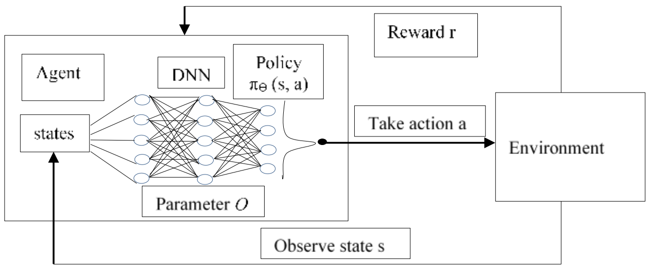
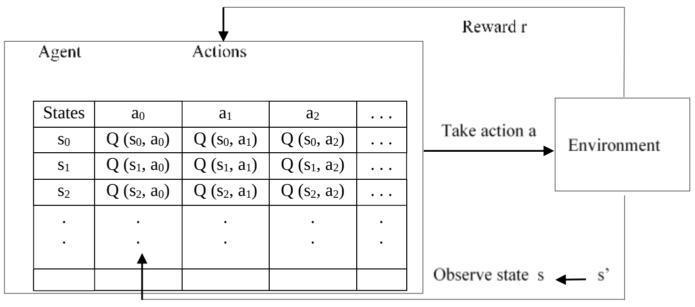


Figure 1. Schematic depiction of deep reinforcement learning [15]



States	a_0	a_1	a_2	...
s_0	Q (s_0, a_0)	Q (s_0, a_1)	Q (s_0, a_2)	...
s_1	Q (s_1, a_0)	Q (s_1, a_1)	Q (s_1, a_2)	...
s_2	Q (s_2, a_0)	Q (s_2, a_1)	Q (s_2, a_2)	...
. .	. .	. .	. .	. .

Figure 2. Simple schematic of Q-Learning: Q-Table [16]

2.1.2. Dyna Q-learning algorithm

The Dyna-Q model is used to create more training data for the agent and simulate experiences. The agent can then use the simulated events to update its Q-table, allowing it to learn more quickly and decide more wisely. By providing fictitious experiences that the agent has never had, the model-based component aids with environment exploration as well [13], [17].

In Q-learning, the agent chooses its course of action based on the highest possible Q-values for a certain state. Through reinforcement learning, the Q-values are progressively improved. exploitation and exploration are combined. It uses a model of the environment to simulate future states in addition to updating Q-values, allowing the agent to explore and plan its course of action. The model-free method Q-learning directly learns the best course of action by changing Q-values in response to observed state-action-reward transitions. It is independent of an environment model. It keeps an environment model and employs it to simulate potential future situations. This improves planning abilities by enabling the agent to adjust Q-values through simulated experiences. Dyna-Q uses a model of the world to create simulated experiences, incorporating planning into the process. This enables more effective research and decision-making by taking conceivable future states into account [18].

Q-learning simply needs to update Q-values based on observed transitions; it often has reduced computational complexity [13]. Dyna Q-Learning Due to model updates and planning stages, Dyna-Q increases computational complexity. The computing demands may rise when hypothetical future situations are simulated and Q-values are updated based on simulated experiences [18]. The Dyna Q-learning's model-based planning for Quadcopter promotes quick decision-making [19]. This ability for real-time adaptation enables the quadcopter to instantly modify its trajectory in response to changing environmental factors, enhancing overall performance and responsiveness.

Dyna Q-learning presents a promising method for improving quadcopter path planning. It is particularly suited for the difficulties presented by quadcopter navigation in complex and dynamic situations since it can efficiently learn from both real experiences and simulated environments. The Q-learning reinforcement learning method seeks to identify the best course of action for an agent interacting with its surroundings [20]. The Q-values are iteratively updated according to the agent's experiences under the Q-learning update rule. In the Dyna-Q update, the agent uses an exploration approach to choose an action depending on its current Q-values. Second, using the Q-learning, the agent adjusts its Q-value for the prior state-action combination after seeing the outcome's state and reward. The Dyna update rule is given by (2):

$$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma max_a Q(s',a) - Q(S,A)] \quad (2)$$

a. Experimental setup
   1) Description of simulation environment

The simulation environment in this project aims to train a quadcopter agent to plan optimal paths in a 3D map with obstacles. The map is represented by a grid of dimensions mapWidth x mapLength x mapHeight. The agent's objective is to navigate from a start pose to a goal pose while avoiding obstacles. The environment provides information about the current state (pose) of the agent and allows it to take actions corresponding to different movement directions in 3D space.

   2) Training case studies and performance indicators

The agent's training is conducted using Dyna Q-learning, an algorithm that combines model-free reinforcement learning techniques. The training involves multiple episodes, each consisting of a sequence of steps. The performance of the agent is evaluated based on two key indicators:

– Total reward per episode

The cumulative reward obtained by the agent throughout each episode reflects its ability to navigate efficiently towards the goal pose while avoiding obstacles. A higher total reward indicates better performance.

– Total steps per episode

The total number of steps taken by the agent in each episode indicates the efficiency of the planned paths. A lower number of steps suggests that the agent has successfully learned to navigate through the environment more directly.

b. Assessment of Dyna Q-learning for planning quadcopter paths

The use of Dyna Q-learning in this work offers several advantages and unique features:

   1) Reward function customization

The reward function is tailored to encourage optimal behavior. Positive rewards are assigned for reaching the goal pose, while negative rewards are given for colliding with obstacles or deviating from the planned path. By fine-tuning the reward function, the agent learns to prioritize efficient and obstacle-free navigation.

   2) Extended action space

The agent is equipped with an expanded set of 26 actions, allowing for more precise and accurate movement in 3D space. This additional granularity enhances the agent's ability to navigate through complex environments and plan optimal paths.

   3) Deterministic environment

The environment is deterministic, meaning that the outcomes of actions are predictable. Exploiting this determinism, the Q-values are updated directly using the current state, next state, and reward information. Temporal-difference learning is not utilized, allowing the agent to converge to the optimal policy faster. By employing Dyna Q-learning with these modifications, the agent aims to learn an optimal policy.

4) Slight improvment to the leterature method

In this work, we made some slight modifications compared to previous methods of reinforcement learning. to the way of learning the first is buy starting the training of the agent in locations close to the goal pose to help find optimal solutions faster knowing that for any reinforcement learning algorithm in order to work or find the optimal behavior needs to get to the goal pose to obtain maximum reward thus an optimal policy obtained.

2.2. Control methods

2.2.1. Unmanned aerial vehicles fuzzy logic control

For the purpose of controlling complex nonlinear systems, it is practical to translate theoretical descriptions into automatic control strategies [21]. The high characteristics of a UAV include its capacity to conduct quick movements, take off and land vertically, and hover in a stable air state. However, because a quadrotor is an unstable and underactuated nonlinear system, developing a high-performance drone controller is a complex task [22]. Dynamic systems are characterized by exterior disturbances, unknown parameters, and complex nonlinearities [23]. Then it is suggested to develop a fuzzy logic controller to ensure the stability of the quadrotor.

a. Fuzzy rules

Fuzzy logic is usually expressed by linguistic rules showing in Figure 3 by the form:

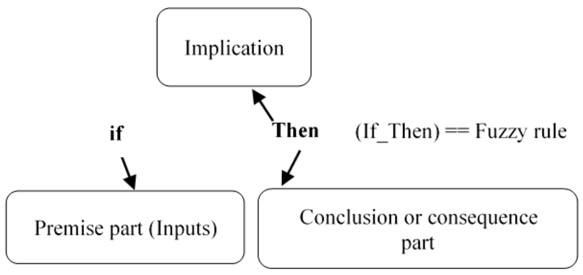


Figure 3. Linguistic fuzzy rule

The general form of the fuzzy rules is given as (3):

$$\text{If } X1 \text{ is } x1 \text{ and/or } X2 \text{ is } x2 \text{ and/or } \dots Xn \text{ is } xn \text{ then Y is y.} \quad (3)$$

The proposed method presents the advantages of the defuzzification and adaptive inference engines that have been fuzzified. Along with various other approaches, the fuzzy controller is evaluated subjectively and objectively, where the processing time is taken in consideration [24].

b. The quadrotor rule base

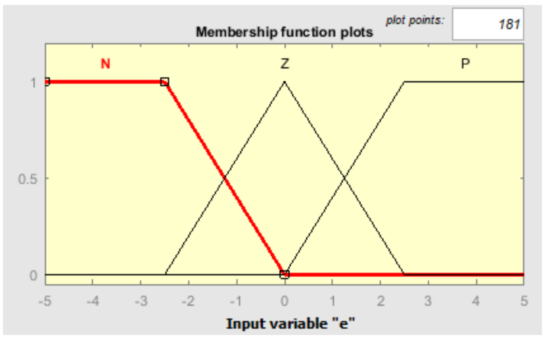
A rule base has been established for all six controllers, where the Table 1 represent the rules of each controller (roll, pitch, yaw, and *x*, *y*, *z* positions) ensuring that the rules are derived based on empirical evidence and careful analysis.

Table 1. Quadrotor rule base

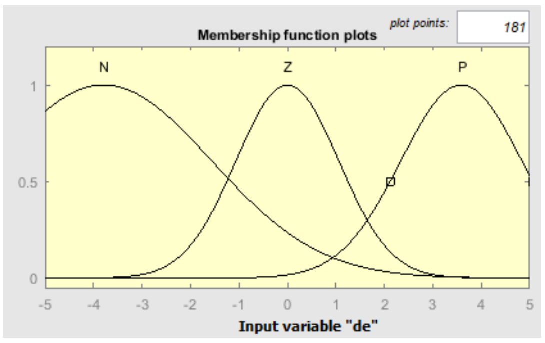
***dE* dt**	NB	N	Z	P	PB
N	GDM	GD	GD	S	GU
Z	GUM	GD	S	GU	GUM
P	GD	S	GU	GUM	GUM

where: N is negative; Z is zero; P is positive; GUM is go up much; GU is go up; S is stand; GDM is go down much; GD is go down; NB is negative big; and PB is positive big.

Triangular, trapezoid, and Gaussian membership functions are the most used for controlling UAVs. The input range for the system is set as [-4, 4], while the output variables have different ranges: [-12.22, 12.22] for U1[-3.05, 3.05] for U2 and U3, [-0.066, 0.066] for U4, and [-1, 1] for pitch and roll. The membership functions for each controller are defined using the "fuzzy" instruction in MATLAB, as illustrated in the accompanying Figures 4 and 5, where Figure 4(a) displays the error input and Figure 4(b) displays the derivative of error input for membership function and Figure 5 displays the output membership function.



(a)



(b)

Figure 4. Membership functions for; (a) error input and (b) derivative of error input

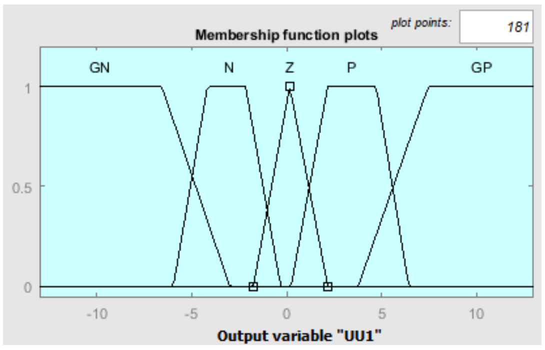


Figure 5. Output membership function

c. Roll controller

A rotating subsystem (roll, pitch, and yaw) and a translating subsystem (x, y, and z coordinates) could be used to control the movement of a quadrotor [25]. To handle the control input of a roll controller, an equation script is employed that makes use of the "evalfis" instruction. This script ensures that the controller operates within the defined range by implementing a saturation code. This code prevents any situation where the value exceeds the specified range, thereby maintaining control stability and reliability.

$$U2(k) = KP\theta e\theta(k) + KD\theta[e\theta(k) - e\theta(k-1)] + KI\theta[e\theta(k) - 2e\theta(k-1) + e\theta(k-2)] + evalfis(U3(k), [e(k), e(k-1)]) \quad (4)$$

Such that: $e\theta$ is the error between the referential roll angle and the desired roll angle.

$$e\theta\ (k) = \theta ref(k) - \theta(k) \quad (5)$$

d. Pitch controller

To manage the control input of a pitch controller, an equation script is utilized that incorporates the "evalfis" instruction. This script allows the pitch controller to operate within the specified range and includes a saturation code to prevent any conditions that could cause the value to exceed the defined range. The saturation code ensures that the pitch controller remains within the desired boundaries, ensuring stability and preventing any undesirable effects.

$$U3(k) = KP\varphi e\varphi\ (k) + KD\varphi[e\varphi\ (k) - ee\varphi\ (k-1)] + KI\varphi[e\varphi\ (k) - 2e\varphi\ (k-1) + e\varphi\ (k-2)] + evalfis(U3(k), [e(k), e(k-1)]) \quad (6)$$

where evalfis is the instruction of the FLC and $e\varphi$ is the error between the referential pitch angle and the desired pitch angle:

$$e\varphi(k) = \varphi ref(k) - \varphi(k) \quad (7)$$

e. Yaw controller

From (8) and (9) can be used to define the control input for yaw angle:

$$U4(k) = KP\psi e\psi(k) + KD\psi[e\psi(k) - e\psi(k-1)] \\ +KI\psi[e\psi(k) - 2e\psi(k-1) + e\psi(k-2)] + evalfis(U4(k),[e(k),e(k-1)]) \quad (8)$$

$$e\psi(k) = \psi ref(k) - \psi(k) \quad (9)$$

Such that: $e\psi$ is the error between the referential yaw angle and the desired yaw angle.

f. Altitude controller

The altitude control input is specified as the control input for the quadrotor's Z position can be defined as (10) and (11):

$$U1(k) = KPZeZ(k) + KDZ[eZ(k) - eZ(k-1)] \\ +KIZ[eZ(k) - 2eZ(k-1) + eZ(k-2)] + evalfis(U1(k),[e(k),e(k-1)]) \quad (10)$$

$$eZ\,Z(k) = Zref(k) - Z(k) \quad (11)$$

Such that: e_Z is the error between the referential Z position and the desired Z position.

g. X and Y position controller

While the quadrotor's ability to maintain a forward and level position is commendable, it is not flawless. In the presence of unexpected external forces, there is a possibility of slight roll or pitch angles being introduced, causing a deviation from the desired position. To address this issue, a fuzzy PID controller can be employed for each position, allowing for the adjustment of these angles based on the quadrotor's X and Y positions. By implementing this controller, the quadrotor can effectively counteract any deviations and maintain its desired position more accurately.

$$\theta ref(k) = KPXeX(k) + KDX[eX(k) - eX(k-1)] \\ +KIX[eX(k) - 2eX(k-1) + eX(k-2)] + evalfis(\theta ref(k),[e(k),e(k-1)]) \quad (12)$$

$$\varphi ref(k) = KPYeY(k) + KDY[eY(k) - eY(k-1)] + KIY[eY(k) - 2eY(k-1) \\ +eY(k-2)] + evalfis(\varphi ref(k),[e(k),e(k-1)]) \quad (13)$$

2.2.2. Proportional integral derivative adaptive fuzzy logic control

a. Principle

When using the PID control approach, tracking performance is comparatively lower since the PID controller is unable to alleviate the effects of external interference and the uncertainty of the system model, so it is required to adjust the PID's parameters in real time [26]. Adaptive control refers to a method employed to adjust and operate a system in real-time. In industrial applications, conventional PID controllers are commonly utilized. However, in robotic applications, traditional controllers often face challenges such as overshooting and oscillation around settling points. These issues can be addressed by incorporating fuzzy logic control. When developing an adaptive controller, the initial step involves understanding how fuzzy logic interacts with the parameters of a conventional PID controller (Figure 14), as well as with the error and error signal. The error and rate of change of error are fed into the fuzzy logic controller, which then produces adjustments in the values of Kp, Ki, and Kd (proportional, integral, and derivative gains). Using this technique, we insured, System's sustainability and robustness in the face of external wind disturbances when compared to the classical PID. The simulation results validate the optimization of the system parameters [27] of the PID controller in real time. Figure 6 shows a block diagram of the adaptive fuzzy PID control. In this paper, we have compared and evaluated deep learning methods in order to determine the suitable algorithm for distance measurement for autonomous drone controlled in real time [28].

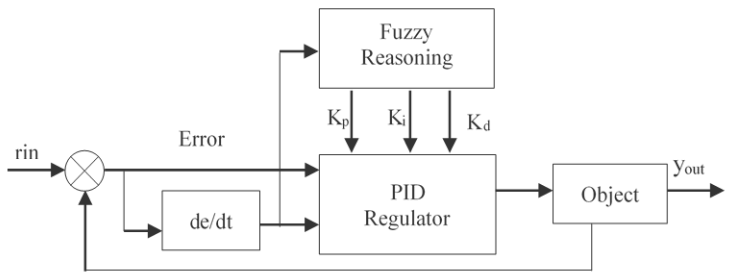


Figure 6. Adaptive fuzzy PID control

3. RESULTS AND DISCUSSION

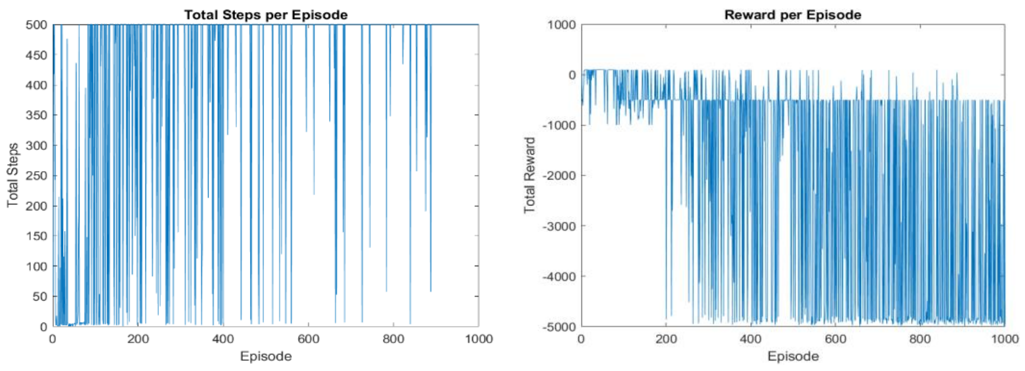
3.1. Path planning

3.1.1. Deep Q-learning algorithm

In this simulation, we set a 3D environment. For each episode, random starting points were chosen on the map, from which the agent initiated its navigation. The number of steps taken and the total reward accumulated during each episode were recorded. Additionally, the total time taken for the agent to complete the learning process was measured. The steps graph shown above tells us that the agent does not get to the goal point so often since the number of steps is always near its maximum. The reward graph shows that it gets negative quite often specially at the latest episodes which is completely inacceptable and quite the opposite of what was expected. This method trained the neural network 32,000 times and took nearly 16 hours.

The application of Deep Q-learning for UAV path planning in the 3D grid, where it follows a predetermined path to collect calculation tasks from smart devices posed significant challenges, leading to unsatisfactory results [29]. Deep reinforcement learning trains general-purpose neural network approach to overcome the constraint [30] using a reward-based system, the UAV learns to select optimal actions based on its current state. The neural network employed in the learning process faced difficulties in deriving patterns or relationships between the input and output, primarily due to the fixed Q-values of the occupied spaces set to 0, which remained unmodifiable throughout the learning phase.

The inability of the neural network to learn meaningful Q-values severely impacted the agent's decision-making capability and hindered the effectiveness of the Deep Q-learning approach. Consequently, the agent exhibited suboptimal path planning behavior and struggled to navigate the environment efficiently. The challenges encountered in this study underscore the difficulties of utilizing Deep Q-learning in complex, 3D grid world environments for UAV path planning. The unique characteristics of the environment, such as the fixed Q-values of occupied spaces, present obstacles that hinder the learning process and prevent the agent from acquiring an optimal policy. Further investigation and potential modifications to the Deep Q-learning approach, including reward shaping, exploration strategies, network architecture, or augmented inputs shown in Figure 7(a) represent total step (y-axe) per episode (x-axe) and Figure 7(b) represent reward (y-axe) per episode (x axe) (may be necessary to address these challenges and improve the results in similar scenarios).



(a) (b)

Figure 7. Response per episode for; (a) total steps and (b) rewards for dqn algorithm

In conclusion Deep Q-learning doesn’t fit a nonlinear model due to nature of neural networks that is why we moved to Dyna Q-learning that puts data in a tabular form. Figure 2 shows the simple schematic of Dyna Q-Learning and Figure 3 shows the simple schematic of Q-Learning, and explain how they differ where a Deep Q-learning that uses neural networks and underneath it a dyna that uses a table.

3.1.2. Dyna Q-learning algorithm

Each episode's total steps taken and accrued rewards were kept track off. Additionally, the amount of time needed for the agent to finish learning was calculated. The Bellman equation, which is significantly more straightforward, was used as the update rule for the q instead of the temporal differencing (TD) error. Figure 8(a) shows response for reward per episode and Figure 8(b) shows total steps per episode for Dyna_Q and Figure 9(a) shows response for reward per episode and Figure 9(b) shows total steps per episode in the improved Dyna_Q.

We first start by sampling random points in the entire map with equally distributed probabilities, it took the agent 202.3 seconds to complete the learning we started training the agent in the points closer to the goal point and then we keep getting farther until we cover the entire map. This method took the agent 144.0879 seconds to complete learning. The method of training the agent in a smaller environment with in the neighborhood of the goal point took the agent 144.0879 seconds to learn, which is quite a significant improvement over the standard method. It is 29% faster to be precise. In the Figures 4 and 5, we see that the agent takes an obstacle clean path and manages to get to the goal point successfully and does not take a very long path (Figure 10) which is exactly the goal of this work. The final model tested to generate paths from 2 different start points, where Figure 10(a) represent the occupancy map for the first Start point and Figure 10(b) represent the occupancy map for the second start point.

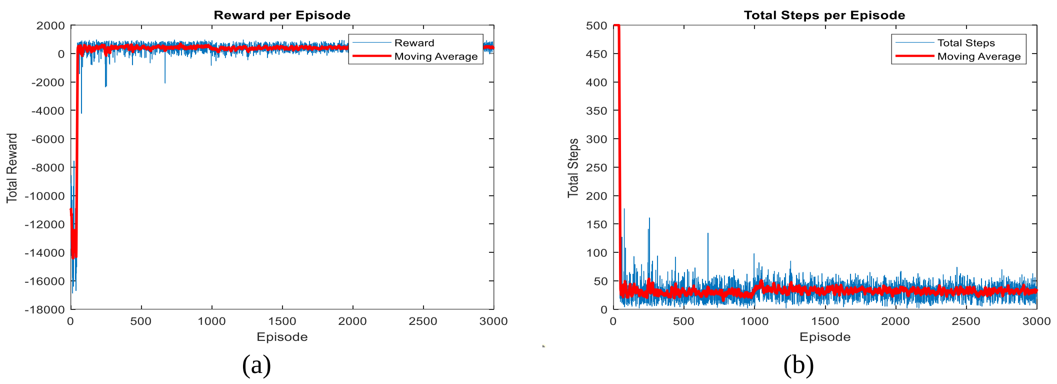


Figure 8. Display response for; (a) reward and (b) total steps per episode for Dyna_Q

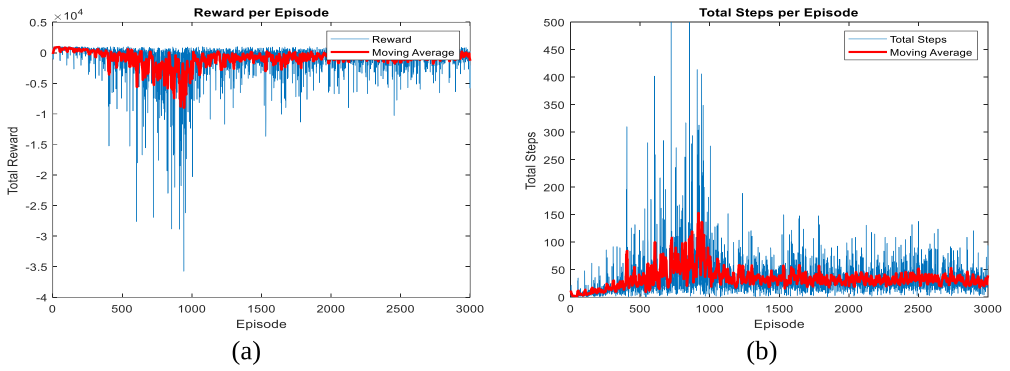


Figure 9. Display response for; (a) reward and (b) total steps per episode in the improved Dyna_Q

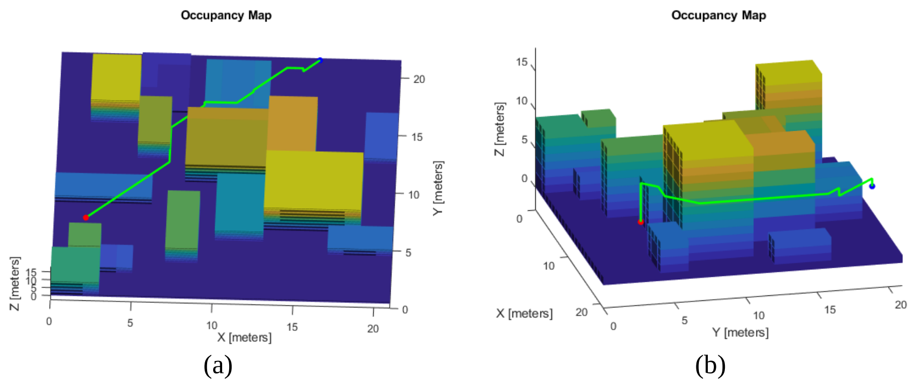


Figure 10. Occupancy map for; (a) first start point and (b) second start point to generated path

3.2. Control methods

3.2.1. Unmanned aerial vehicles fuzzy logic control

The drone's objective is to arrive at the terminal destination safely [31], so in this design, we used a pre-defined UAV model which has its control laws already calculated. The fuzzy logic controller than was added after the PID controller, so it compensates any error or miss-calculation from the PID. This control method is a combination of traditional PID controller with fuzzy logic [26]. A generated path from our path planning model was used as the input for the fuzzy logic control system. After running the simulation, obtained the results shown in Figures 11 and 12, where Figure 10 shows the control inputs for path 1.

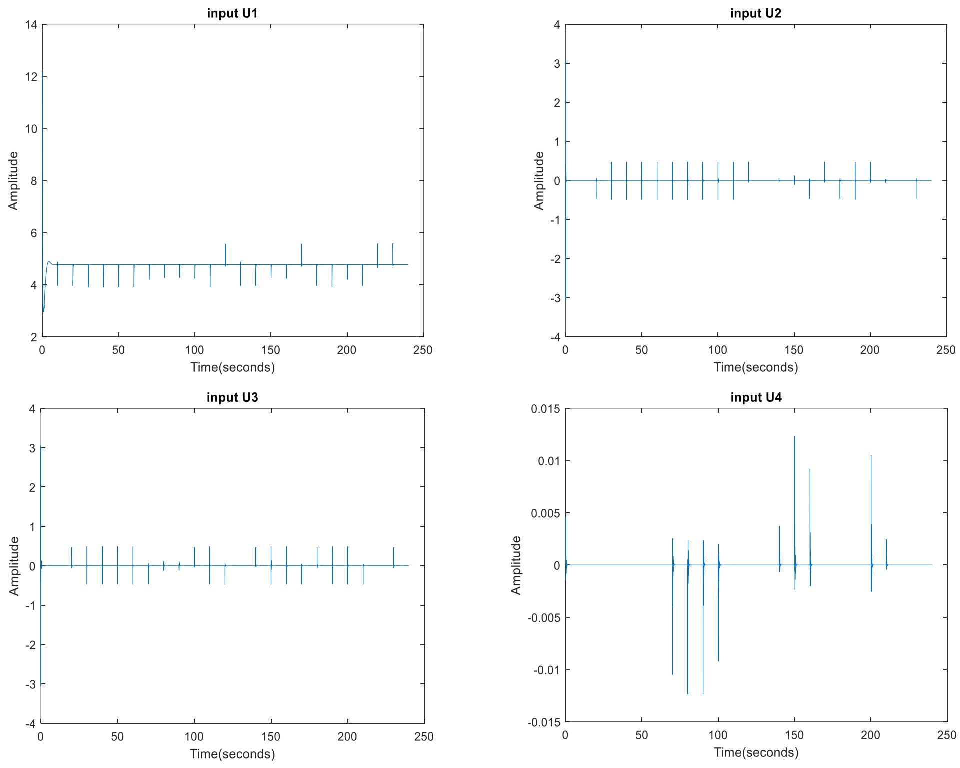


Figure 11. Control inputs for path 1

A resilient controller is the fuzzy logic controller. We are extremely convinced that this controller is more than capable of handling the task because it demonstrated its performance throughout our experiment and provided some incredibly adequate performance figures. It also successfully followed the generated paths, which gives us even more confidence. Figure 12(a) (in Appendix) is an illustration of the global trajectory in (x, y, and z axes) and Figure 12(b) (in Appendix) shows the fuzzy PID responses of the quadrotor in 3D for path 1. After running the simulation for the second generated path we obtained the graphs shown in Figure 13 (in Appendix) that represent control inputs for path 2 and Figure 14(a) (in Appendix) is an illustration of the global trajectory in (x, y, and z axes) and Figure 14(b) (in Appendix) shows fuzzy PID responses of the quadrotor in 3D for path 2.

The Table 2 shows that there is no steady-state, and the settling time sits just under 6 seconds for the first path and a little more than 4 seconds in the second path. Pitch and Yaw angles exhibit no steady-state error. Note that the settling time for X and Y positions is 0. This is due to the nature of the path and has nothing to do with the performance of the system.

Table 2. The results for the fuzzy controllers of the UAV

	Settling time first, (second) path	Steady state error
X position	0 Sec, (0 Sec)	0
Y position	0 Sec, (0 Sec)	0
Z position	5.8 Sec, (4.22 Sec)	0

3.2.2. Proportional integral derivative adaptive fuzzy logic control

For the simulation, the parameters of each PID control are adapted by a fuzzy controller, and the results are shown by the Figures 15 to 21. The table shows that there is no steady-state, and the settling time for the z position sits just under 5.6 seconds for the first generated path and there is hardly any settling time for the X and Y positions. Which means that the adaptive fuzzy proved to be a better alternative than the standard fuzzy logic control system.

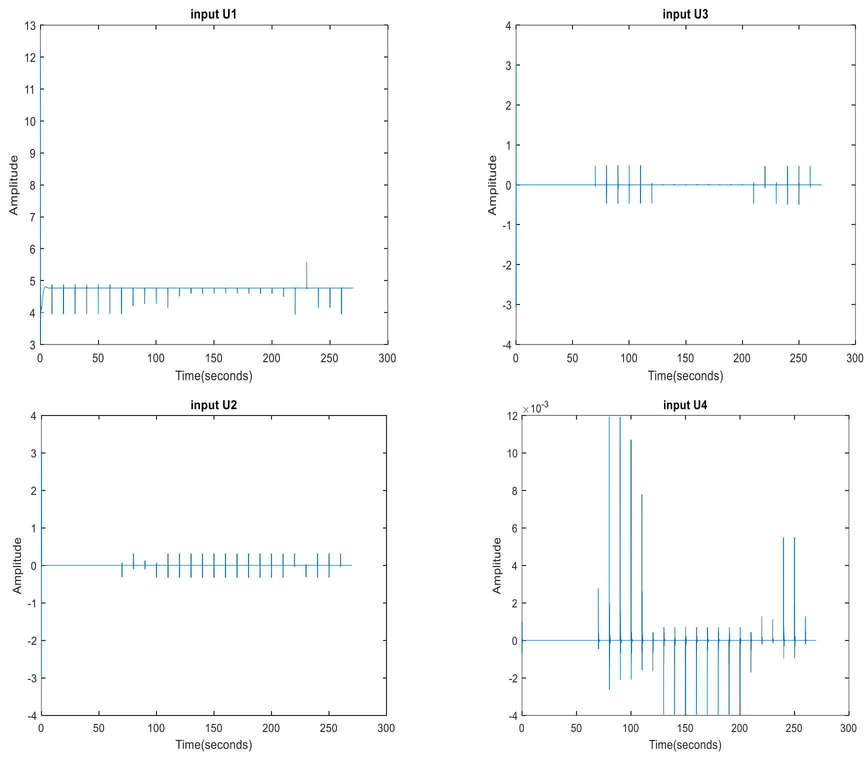


Figure 15. Adaptive fuzzy logic control inputs

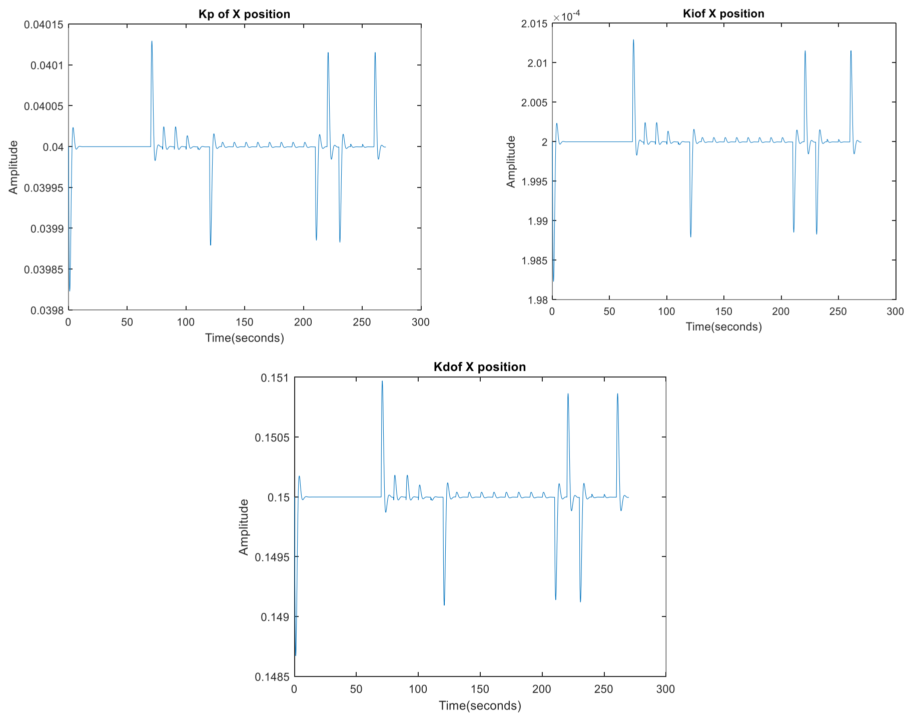


Figure 16. PID parameters of x position

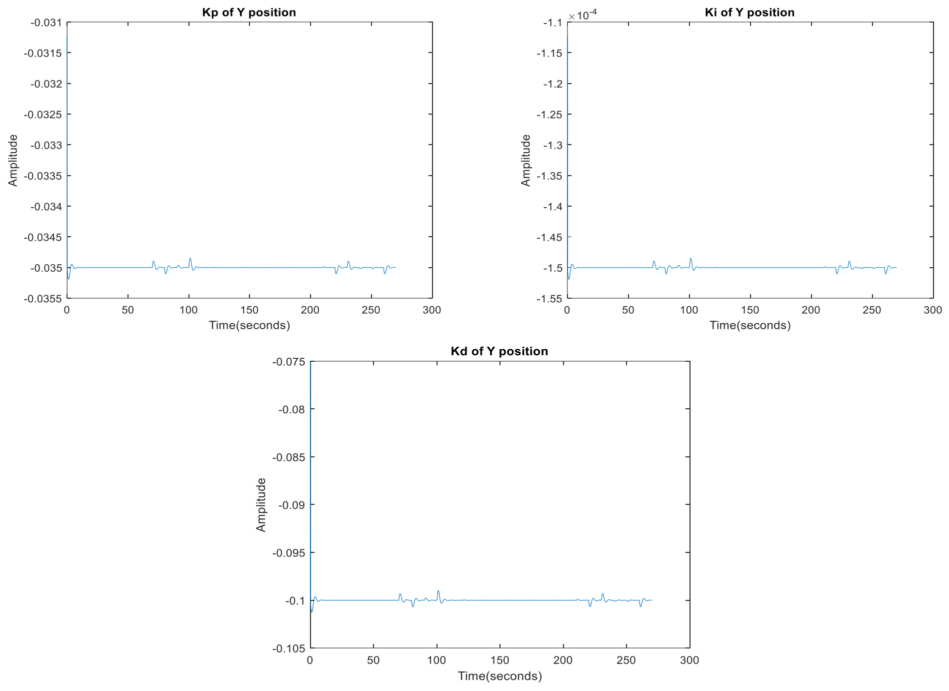


Figure 17. PID parameters of y position

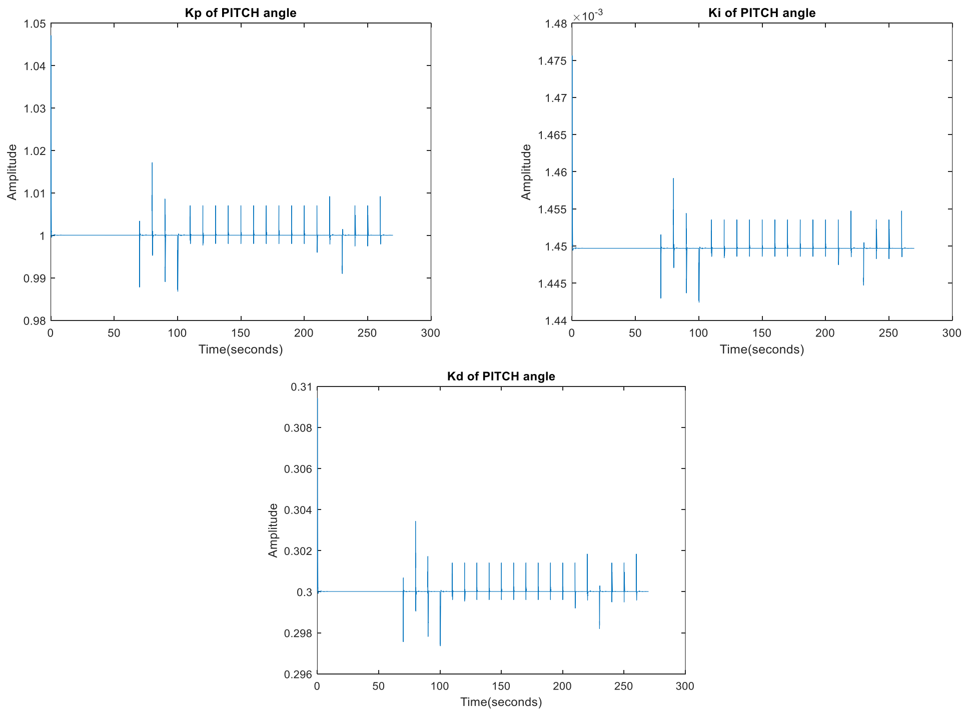


Figure 18. PID parameters of pitch angle

Figure 15 illustrate the control inputs (time in x-axe and amplitude in y-axe) of the quadrotor for adaptive fuzzy logic controller and Figures 16 to 20 illustrate successively the PID parameters for (x, y and z position and pitch, roll and yaw, where the (time in x-axe and amplitude in y-axe). Figure 21(a) is an illustration of the global trajectory of the quadrotor in (x, y, and z axes), and Figure 21(b) shows the adaptive fuzzy response successively of (pitch, roll and yaw angle and x, y, and z positions) of the quadrotor in 3-D.

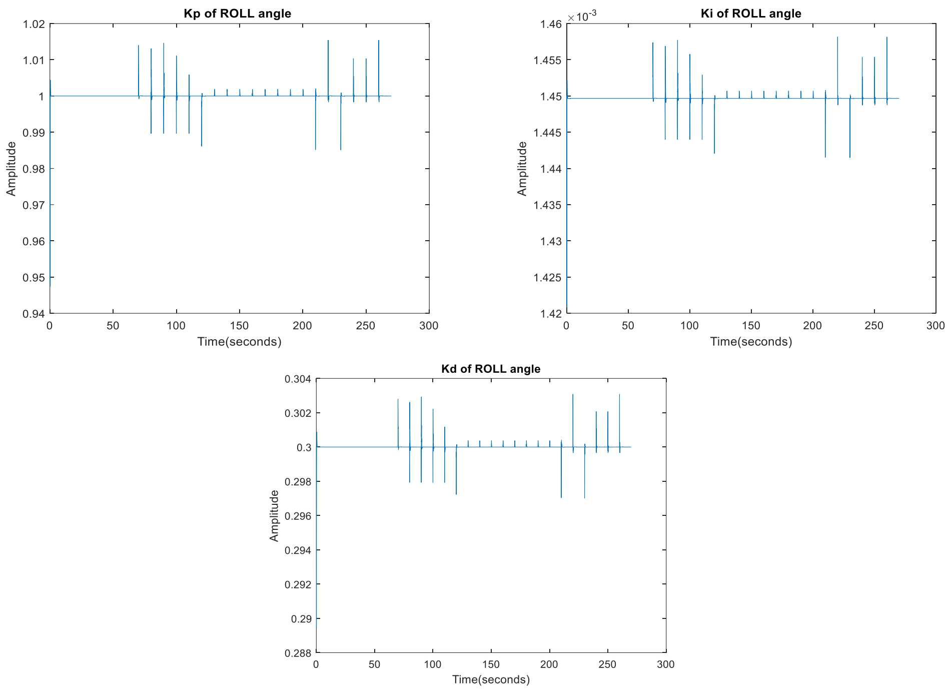


Figure 19. PID parameters of roll angle

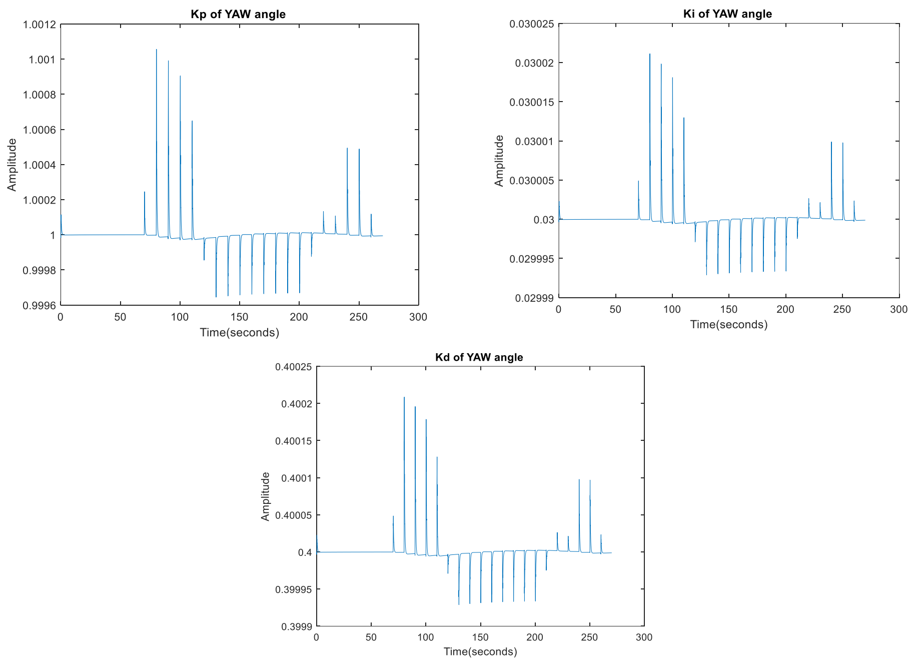
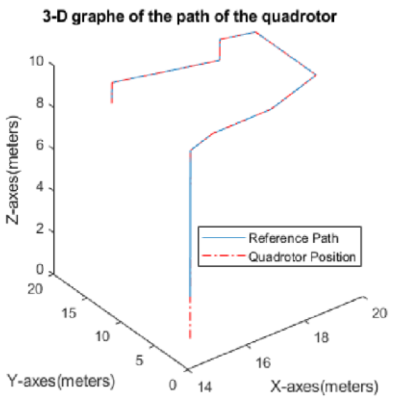
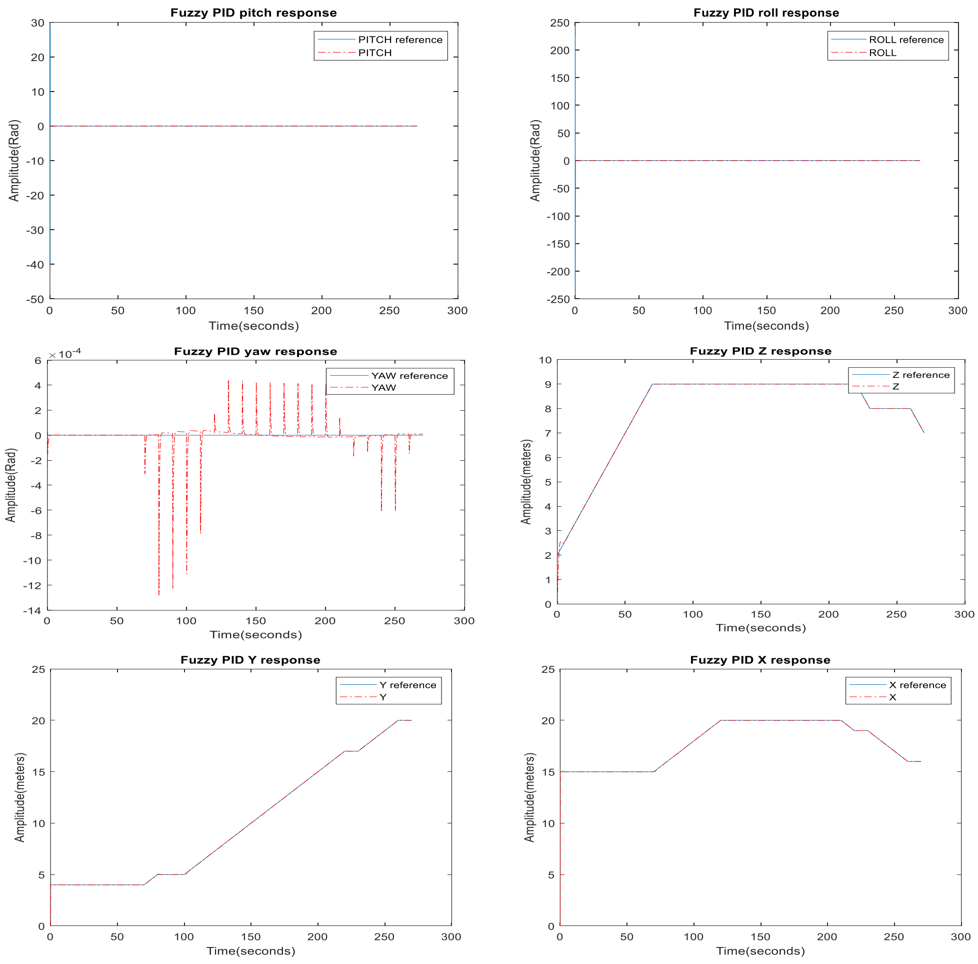


Figure 20. PID parameters of yaw angle



(a)



(b)

Figure 21. Global trajectory; (a) adaptive fuzzy response and (b) for the UAV in 3-D

Table 3 shows that there is no steady-state, and the settling time for the z position sits just under 5.6 seconds for the first generated path and there is hardly any settling time for the X and Y positions. Which means that the adaptive fuzzy proved to be a better alternative than the standard fuzzy logic control system.

Table 3. The discussion of the adaptive fuzzy controllers for the UAV

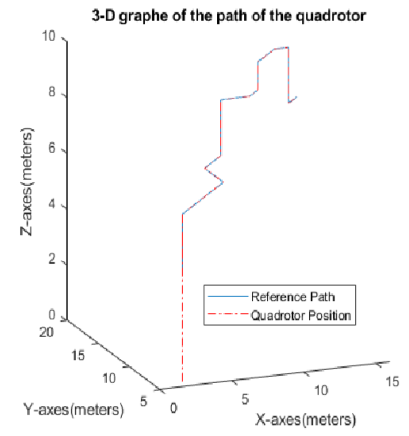
	Settling time	Steady state error
X position	0 Sec	0
Y position	0 Sec	0
Z position	5.55 sec	0

4. CONCLUSION

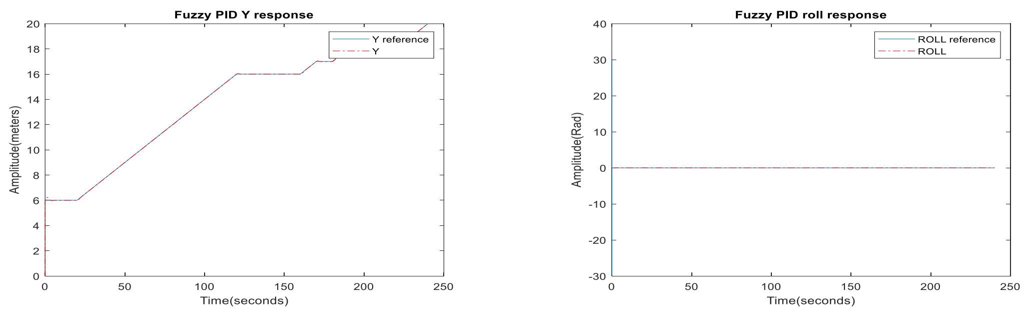
The objective of this research was to evaluate the suitability of Deep Q-learning and Dyna Q-learning algorithms for UAV path planning, along with the integration of a fuzzy logic control system for precise path following. The findings of this study indicate that Deep Q-learning is not well-suited for UAV path planning due to the absence of clear patterns or relations in the data. However, Dyna Q-learning, with customized modifications, showed promise in learning an optimal policy quicker than the standard Dyna algorithm. Additionally, the integrated fuzzy logic control system proved to be highly effective in ensuring precise path following by the UAV, and its more advanced version "Adaptive Fuzzy Logic" does always have the edge over the standard one when it comes to overall performance and versatility. The results suggest that when dealing with complex and unpredictable environments, such as UAV path planning, Deep Q-learning may not be the most suitable algorithm. The lack of discernible patterns in the data poses challenges for the neural network to learn accurate Q-values. However, Dyna Q-learning, with bespoke modifications, overcame these challenges and demonstrated improved performance in learning optimal policies.

Based on the findings, it is recommended to further explore and refine the customized Dyna Q-learning algorithm for UAV path planning. Additionally, the integration of adaptive fuzzy logic control systems should be considered for other autonomous systems requiring precise path following. Further research could focus on optimizing the customization of the Dyna algorithm and investigating alternative reinforcement learning algorithms that may better suit the UAV path planning problem. By identifying the limitations of Deep Q-learning for UAV path planning, showcasing the effectiveness of customized Dyna Q-learning, and highlighting the applicability of fuzzy and adaptive fuzzy logic control systems, this study contributes to the advancement of autonomous systems in complex environments. The insights gained from this research can guide further developments in path planning algorithms and enhance the capabilities of UAVs and other autonomous systems.

APPENDIX

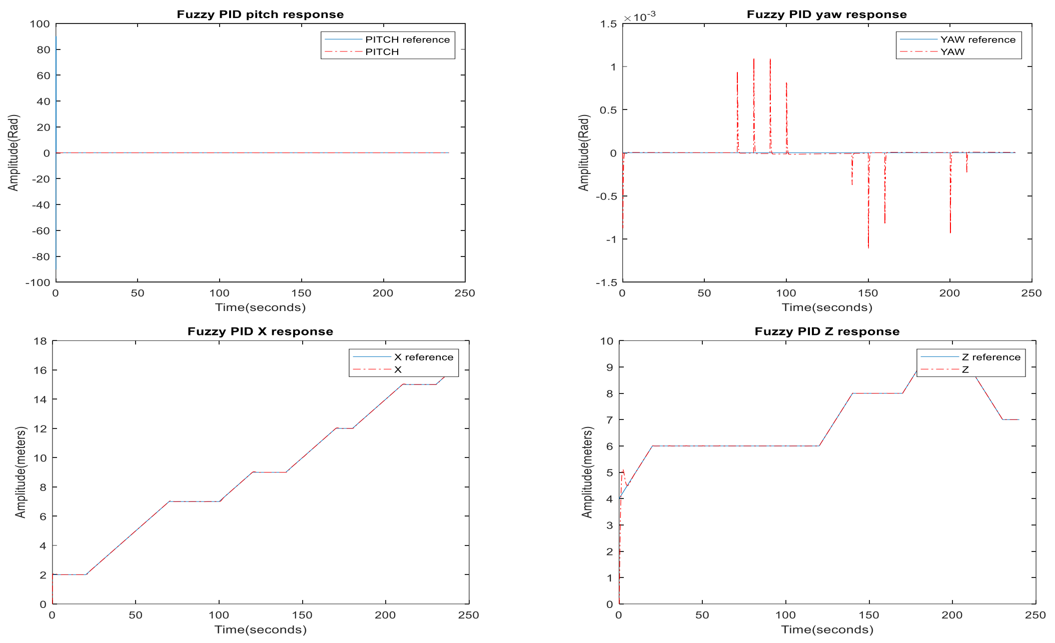


(a)



(b)

Figure 12. Global trajectory; (a) fuzzy PID responses and (b) of the quadrotor in 3D for path 1



(b)

Figure 12. Global trajectory; (b) of the quadrotor in 3D for path 1 *(continued)*

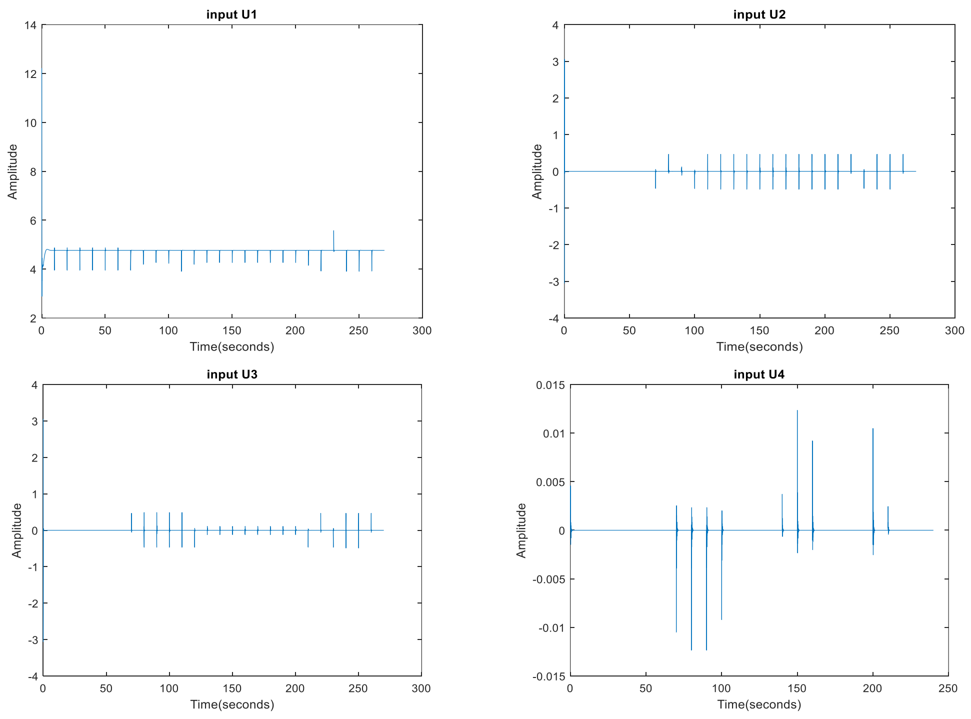
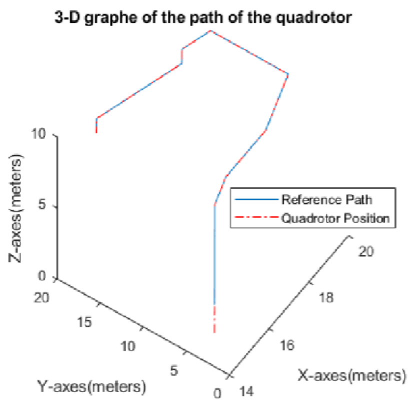
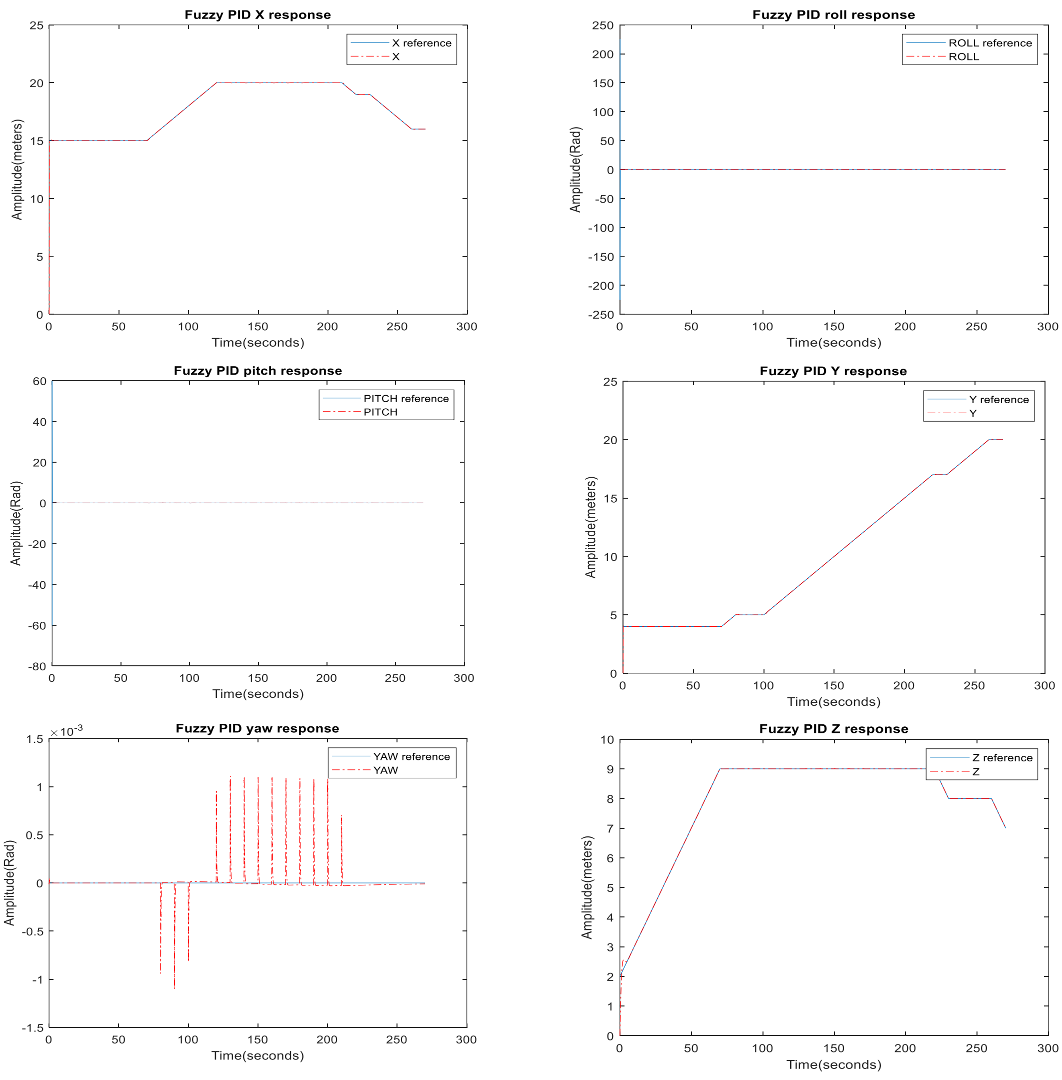


Figure 13. Control inputs for path 2



(a)



(b)

Figure 14. Global trajectory; (a) fuzzy PID responses and (b) of the quadrotor in 3D for path 2

REFERENCES

[1] N. V. Varghese and Q. H. Mahmoud, "A survey of multi-task deep reinforcement learning," *Electronics (Switzerland)*, vol. 9, no. 9, pp. 1–21, Aug. 2020, doi: 10.3390/electronics9091363.

[2] G. T. Tu and J. G. Juang, "Path planning and obstacle avoidance based on reinforcement learning for UAV application," in *Proceedings of 2021 International Conference on System Science and Engineering, ICSSE 2021*, IEEE, Aug. 2021, pp. 352–355,

doi: 10.1109/ICSSE52999.2021.9537945.
[3] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous Drone Racing with Deep Reinforcement Learning," in *IEEE International Conference on Intelligent Robots and Systems*, IEEE, Sep. 2021, pp. 1205–1212, doi: 10.1109/IROS51168.2021.9636053.
[4] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, vol. 30, no. 1, pp. 2094–2100, Mar. 2016, doi: 10.1609/aaai.v30i1.10295.
[5] Y. Bi, Y. Wu, and C. Hua, "Deep Reinforcement Learning Based Multi-User Anti-Jamming Strategy," in *IEEE International Conference on Communications*, IEEE, May 2019, pp. 1–6, doi: 10.1109/ICC.2019.8761848.
[6] J. Huang, Q. Tan, J. Ma, and L. Han, "Path Planning Method Using Dyna-Q Algorithm under Complex Urban Environment," in *Proceedings - 2022 Chinese Automation Congress, CAC 2022*, IEEE, Nov. 2022, pp. 6776–6781, doi: 10.1109/CAC57257.2022.10054800.
[7] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013, doi: 10.1177/0278364913495721.
[8] A. H. Arani, M. M. Azari, P. Hu, Y. Zhu, H. Yanikomeroglu, and S. Safavi-Naeini, "Reinforcement Learning for Energy-Efficient Trajectory Design of UAVs," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 9060–9070, Jun. 2022, doi: 10.1109/JIOT.2021.3118322.
[9] J. Li, X. Xiong, Y. Yan, and Y. Yang, "A Survey of Indoor UAV Obstacle Avoidance Research," *IEEE Access*, vol. 11, pp. 51861–51891, 2023, doi: 10.1109/ACCESS.2023.3262668.
[10] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv, 150-02971*, 2015.
[11] A. Ramaswamy and E. Hullermeier, "Deep Q-Learning: Theoretical Insights from an Asymptotic Analysis," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 2, pp. 139–151, Apr. 2022, doi: 10.1109/TAI.2021.3111142.
[12] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.
[13] F. N. Zohedi, M. S. M. Aras, H. A. Kasdirin, and N. B. Nordin, "New lambda tuning approach of single input fuzzy logic using gradient descent algorithm and particle swarm optimization," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 25, no. 3, pp. 1344–1355, Mar. 2022, doi: 10.11591/ijeecs.v25.i3.pp1344-1355.
[14] A. T. Humod and N. M. Ameen, "Robust nonlinear pd controller for ship steering autopilot system based on particle swarm optimization technique," *IAES International Journal of Artificial Intelligence*, vol. 9, no. 4, pp. 662–669, Dec. 2020, doi: 10.11591/ijai.v9.i4.pp662-669.
[15] A. Oppermann, "How AI Teach Themselves Through Deep Reinforcement Learning," builtIn, Oct. 21, 2021. [Online]. Available: https://builtin.com/machine-learning/deep-reinforcement-learning.
[16] "20. Q-Table – EN – Deep Learning Bible," Wikidocs, https://wikidocs.net/174536, 2023.
[17] S. P. Singh and R. S. Sutton, "Reinforcement learning with replacing eligibility traces," *Machine Learning*, vol. 22, no. 1–3, pp. 123–158, 1996, doi: 10.1007/BF00114726.
[18] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *ACM SIGART Bulletin*, vol. 2, no. 4, pp. 160–163, Jul. 1991, doi: 10.1145/122344.122377.
[19] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, 2011, pp. 465–472, doi: 10.5555/3104482.3104541.
[20] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, May 1992, doi: 10.1007/bf00992698.
[21] M. Mekhanet, L. Mokrani, A. Ameur, and Y. Attia, "Adaptive Fuzzy Gain of Power System Stabilizer to Improve the Global Stability," *Bulletin of Electrical Engineering and Informatics*, vol. 5, no. 4, pp. 421–429, Dec. 2016, doi: 10.11591/eei.v5i4.576.
[22] M. A. M. Basri and A. Noordin, "Optimal backstepping control of quadrotor uav using gravitational search optimization algorithm," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 5, pp. 1819–1826, Oct. 2020, doi: 10.11591/eei.v9i5.2159.
[23] M. A. A. Ghany and M. A. Shamseldin, "Fuzzy type two self-tuning technique of single neuron PID controller for brushless DC motor based on a COVID-19 optimization," *International Journal of Power Electronics and Drive Systems*, vol. 14, no. 1, pp. 562–576, Mar. 2023, doi: 10.11591/ijpeds.v14.i1.pp562-576.
[24] A. A. Baker and Y. Y. Ghadi, "Autonomous system to control a mobile robot," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 4, pp. 1711–1717, Aug. 2020, doi: 10.11591/eei.v9i4.2380.
[25] A. Saibi, H. Belaidi, R. Boushaki, R. Z. Eddine, and A. Hafid, "Enhanced backstepping control for disturbances rejection in quadrotors," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 6, pp. 3201–3216, Dec. 2022, doi: 10.11591/eei.v11i6.3997.
[26] C. Li, Y. Wang, and X. Yang, "Adaptive fuzzy control of a quadrotor using disturbance observer," *Aerospace Science and Technology*, vol. 128, p. 107784, Sep. 2022, doi: 10.1016/j.ast.2022.107784.
[27] A. Elbatal, A. M. Youssef, and M. M. Elkhatib, "Smart aerosonde uav longitudinal flight control system based on genetic algorithm," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 5, pp. 2433–2441, Oct. 2021, doi: 10.11591/eei.v10i5.2342.
[28] A. I. Arrahmah, R. Rahmania, and D. E. Saputra, "Comparison between convolutional neural network and K-nearest neighbours object detection for autonomous drone," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 4, pp. 2303–2312, Aug. 2022, doi: 10.11591/eei.v11i4.3784.
[29] F. Song *et al.*, "Evolutionary Multi-Objective Reinforcement Learning Based Trajectory Control and Task Offloading in UAV-Assisted Mobile Edge Computing," *IEEE Transactions on Mobile Computing*, vol. 22, no. 12, pp. 7387–7405, 2023, doi: 10.1109/TMC.2022.3208457.
[30] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proceedings - IEEE International Conference on Robotics and Automation*, IEEE, May 2017, pp. 3389–3396, doi: 10.1109/ICRA.2017.7989385.
[31] X. Han, J. Wang, J. Xue, and Q. Zhang, "Intelligent Decision-Making for 3-Dimensional Dynamic Obstacle Avoidance of UAV Based on Deep Reinforcement Learning," in *2019 11th International Conference on Wireless Communications and Signal Processing, WCSP 2019*, IEEE, Oct. 2019, pp. 1–6, doi: 10.1109/WCSP.2019.8928110.

BIOGRAPHIES OF AUTHORS

Yasmine Zamoum received the Engineer degree in Electromechanical Engineering from the university of MHamed Bougara of Boumerdes in 2002, she received the Magister degree in Industrial Electrical Equipment from Faculty of Hydrocarbon and Chemistry from the university of MHamed Bougara Boumerdes in 2010. Currently, she has been an assistant professor at Faculty of Hydrocarbon and Chemistry since 2019 in the Department of Automation and Electrification of Industrial Processes. Her main research interests include: control strategies for drones, renewable energy, and electrical machines. She can be contacted at email: y.zamoum@univ-boumerdes.dz.

Karim Baiche received his Magister's degree in Applied Automation from the University of M'Hamed Bougara Boumerdes in 1998 and received his Ph.D. from the University of M'Hamed Bougara Boumerdes, Algeria in 2014. He is currently an Associate Professor at the University of M'Hamed Bougara Boumerdes, Algeria. His main current research interests include: signal processing, system diagnostics and evolutionary computation, and metaheuristics. He can be contacted at email: kbaiche@univ-boumerdes.dz.

Youcef Benkeddad received his Bachelor's degree in Electrical and Electronics Engineering from the Institute of Electrical and Electronic Engineering at University M'hamed Bougara of Boumerdes, Algeria in June 2021. Subsequently, he pursued his Master's degree in Automatic Control at the same institution, completing it in 2023. Currently, he is continuing his studies at the University of Palermo, Italy., where he is pursuing a Master's degree in Electronics. He can be contacted at email: benkeddadyoucef@gmail.com.

Brahim Bouzida received his Bachelor's degree in Electrical and Electronics Engineering from the Institute of Electrical and Electronic Engineering at University M'hamed Bougara of Boumerdes, Algeria in June 2021. Subsequently, he pursued his Master's degree in Automatic Control at the same institution, completing it in 2023. Currently, , he is continuing his studies at the University of Genoa Italy, where he is pursuing a Master's degree in Strategos. He can be contacted at email: adelbouzida@gmail.com.

Razika Boushaki is a Professor in Electrical Engineering at the University of Boumerdes (Algeria) in the Institute of Electrical and Electronic Engineering since 2003. She obtained Engineer Diploma in 1995, magister Diploma in 2003 at University of Boumerdes and Doctorate degree in June 2013, in Electrical Engineering. She is member in the research laboratory since 2009. She introduced several practical automation systems in industry between 1999 and 2003. Currently, she is Prof. at Institute of Electrical and Electronic Engineering, University M'hamed Bougara of Boumerdes, Algeria. She can be contacted at email: r.boushaki@univ-boumerdes.dz and boushakiraz@yahoo.fr.