



Research Article

Deep Reinforcement Learning-Based Fault-Tolerant Flight Control of a Quadcopter Using Full Rotor Thrust Vectoring After Complete Rotor Failure

Zairil A. Zaludin 

Department of Aerospace Engineering, Universiti Putra Malaysia, Serdang, Selangor, Malaysia

Article Information

Abstract

Article History

Received: 1 July 2026

Revised: 10 August 2026

Accepted: 30 August 2026

Published online: 12 September 2026

Keywords

Deep Deterministic Policy Gradient

Reinforcement Learning

Drone Fault Tolerance

Adaptive Thrust Vectoring

Pan, Tilt Rotor

Correspondence*

zairil_azhar@upm.edu.my

ORCID

Zairil A. Zaludin <https://orcid.org/0000-0001-9599-4084>

This article addresses the issue of rebalancing a quadcopter drone after a single rotor fails, employing an unconventional method of thrust vectoring the remaining rotors. To determine the complex control policy for managing the thrust vectoring components, the Reinforcement Learning method was used to train a Deep Deterministic Policy Gradient (DDPG) agent. The implementation of the control policy during simulation showed that the drone not only managed to avoid crashing after a single rotor failed, but it also remained airborne while maintaining an acceptable change in attitude and altitude, albeit at the cost of the drone moving backwards continuously. The agent's capability to propose a better control policy could be further improved by selecting appropriate reward functions, which should include restricting the motion of the drone in the X- and Y-axis directions.

© 2026 Centre for Research and Innovation (CRI). This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

I. INTRODUCTION

When a single rotor on a quadcopter drone (QD) suddenly malfunctions while flying, there is very little the drone operator can do to rescue the vehicle. Anyone who has had such an experience while flying a QD will testify that when such an incident occurs, all they can do is watch the drone helplessly go out of control and crash to the ground. It is not difficult to understand why QDs have a relatively small margin for recovery when a rotor malfunction occurs. Historically, this type of unmanned vehicle was designed to make the 'job' easier and cheaper for operators, instead of safer. These vehicles were not deliberately designed with sufficient safety features.

As the popularity of QDs increased worldwide, the norm was to follow the trend. Users around the world assume that their drones are safe to fly every single time. However, this is a wrong assumption. When a single rotor fails due to any reason, the inevitable conclusion has always been the same.

Wild [1] published the findings from a study involving drone accidents and incidents, which revealed that most unwanted events were due to technology issues, rather than human factors. This suggests that even if drone operators are competent or hold licenses from aviation authorities, when the drone malfunctions (due to technical failures such as those that lead to rotor stoppage), it must 'come down' somewhere uncontrollably and unpredictably. Unlike fixed-wing aircraft, drones cannot glide to descent; hence, the rate of descent and the orientation of the drone will be uncontrolled and rapid. The operator has no control over the landing or crashing location, nor the speed at which it reaches the ground. As highlighted by Wild [1], the current technology of drones poses a problem, and, unlike manned aircraft, drone technology is not certified or deemed airworthy from the aviation authorities' perspective. And yet, they are flying today in civilian airspace, registered or otherwise. The risk and danger associated with QDs create an issue of trust among the public when they see a QD flying above them. A study conducted by Khavas [2] showed that if

the public could be assured that ‘some’ safety features were incorporated to protect them from danger, their trust in these drones would increase. In that study, a simple test revealed the difference in trust levels of humans when watching a drone crash within a cage, compared to crashing into a wall. In aviation, this has also occurred many times. For instance, the primary flight control system for Boeing 777 incorporates several layers of redundancy [3].

Furthermore, a minimum of 2 human crew pilots are placed in the cockpit to oversee the overall safety of the aircraft. Naturally, as humans, we need assurance for our safety, thanks to millions of years of evolution of the part of the brain called the amygdala [4]. Until such ‘assurance’ is demonstrated, QDs will remain ‘dangerous’ in the public's eyes, and therefore, limit the growth of the industry as a whole. Previous work on fault tolerance control (FTC) for QD confirms there has been interest in making drones safer in the event of malfunctions. This field of study can, generally, be divided into these categories: faulty sensors, faulty actuators, damaged airframes, failure due to computer systems, and communication systems problems [5]–[7]. Of interest in the work presented in this article is the section on the faulty actuator, as a rotor on the drone serves as an actuator.

Guo [8] corroborated that unless standard-configuration QDs were over-actuated, the chances of recovering stability from a single rotor failure were slim. A typical method to over-actuate a QD was to implement a dual rotor system at each corner. However, such a technique altered the fundamental ‘quad’ nature of the drone and imposed additional weight and power requirements. Other authors studied the issue too, but by assuming that the failed rotor did not experience complete failure but rather a partial failure or ‘loss of effectiveness (LOE)’, where the rotor still provided some thrust, albeit limited due to the failure [9]–[11]. The problem with such an assumption is that, in reality, such failure during mid-flight is rarely quantifiable. Even if the rotor was still operating at some percentage of effectiveness, such as 50% or 70%, its reliability had been compromised. A safer option would be to completely stop that faulty rotor from continuing to operate by deactivating its power supply, thereby removing its contribution to the drone's flight dynamics entirely. Perhaps it is better to assume that the failure is total, meaning the rotor comes to a complete stop. From that point, a solution must be deduced to rescue the drone using whatever is still functioning. Other than the study on controllers for drones, which have lost rotor effectiveness, Chung and Son [12] implemented a control reconfiguration technique on a QD experiencing a single motor malfunction. However, the solution proposed sacrificed yaw angle and yaw rate. This, perhaps, temporarily prevented the drone from crashing to the ground. Sooner or later, the drone must land due to a depleted battery level. If the drone continued to yaw rapidly during landing, the impact created while touching down due to rapid yawing would likely damage the drone.

From these studies, the fundamental problem appears to be physics-related. For a drone to fly in a controlled manner, the key to achieving this feat lies in balancing the forces and moments acting on the drone. Therefore, something else must change in the way the forces and moments interact with the drone using the remaining rotors. In one study, Dehkordi and Danesh [13] proposed modifying the quadcopter configuration by designing a mechanism to move the motors along the drone's arm. By adjusting the origin of thrust and torque produced by individual motors, the forces and moments could be rebalanced to provide the unbalanced drone with a chance to regain dynamic stability. Airframe real-time adjustments have yielded some positive results in addressing complete rotor failure, as they can reconfigure the sum of moments and the sum of forces by adjusting the source of forces and the pivot point of moments, provided they are performed correctly. The concept of panning and tilting a remaining working rotor after a single rotor failed was introduced and studied too, to attempt to reconfigure the forces and moments [14], [15]. With the findings from those studies, a new possibility emerged. If such rebalancing were possible, the QD might even be able to continue flying with only three rotors working. If such potential could be incorporated into a QD, the safety of the drone would be significantly improved in the event of a single rotor failure.

The work presented in this article proposes an upgrade to the current configuration of a typical QD. The present drone's simplicity is also its biggest enemy. The small, compact size of the drone makes it so convenient to produce the results the operator wants, but due to its size and simplicity, very few safety features can be incorporated. In this article, the proposed change was to allow all remaining rotors to pan and tilt, and then to enable a separate controller to manage the pan and tilt angles as well as the throttle command input for each remaining rotor. To achieve such tasks and objectives, a Deep Deterministic Policy Gradient (DDPG) agent was used. Through reinforcement learning, the agent was trained to find the solution. The study shows that after implementing the control policy learned during the training, the drone successfully averted crashing to the ground using only the three rotors. The results also show that the drone was able to remain airborne without a significant change in roll, pitch, and yaw and their associated rates. When a single rotor stops during flight, a QD loses balance and inadvertently crashes. This article proposes a solution to improve the outcome of this scenario by allowing the remaining rotors to pan and tilt while changing the thrust command separately for each rotor, thereby rebalancing the drone. The proposed solution presented a control policy obtained through Reinforcement Learning, where a DDPG agent was trained to generate action signals for managing individual pan and tilt angles, as well as thrust commands for each remaining rotor. The aim of the proposed solution was to prevent the unbalanced drone from crashing by maintaining its airborne state with minimal changes in attitude and altitude.

A. Fault-Tolerant Flying Qualities Criteria

At the time of writing, there are no official flying quality criteria published by any aviation authority for unbalanced drones. Therefore, to assist in gauging the success or failure of the fault-tolerant controller, a few flying qualities (FQ) criteria are proposed below. These FQ criteria are used as targets to achieve when designing the control policy. The criteria below apply to a QD that experiences a single rotor failure during hovering.

1. The drone must not crash immediately after failure.
2. The altitude change must be within $\pm 1\text{m}$ of the altitude when failure occurs

3. The roll, pitch, and yaw changes must not exceed $\pm 45^\circ$ from level, respectively.

II. METHODOLOGY

To explain the concept proposed in this article, the drone's mathematical model and Simulink model representation are introduced first. Following this, the modifications made to the original Simulink model to incorporate the Reinforcement Learning controller are described.

A. When All Rotors Are Working Properly

An illustration of the QD tested is shown in [Figure 1](#) to assist in understanding the concept proposed in this article. The rotors are named as follows, with respect to the body axis.

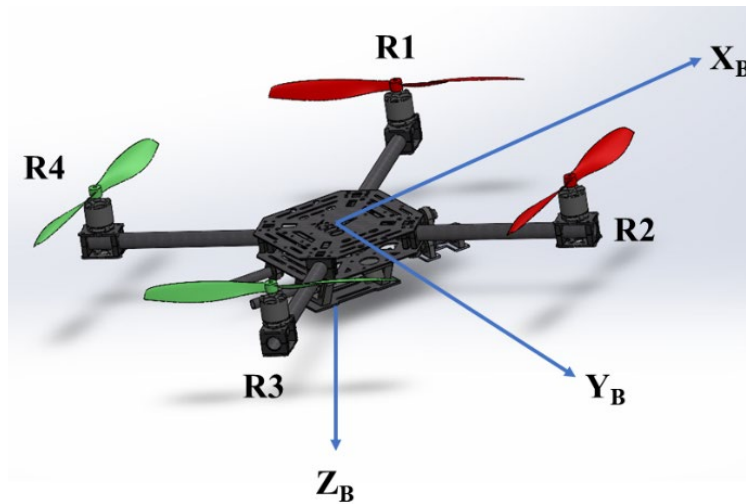


Fig.1 The Quadcopter Drone

The details of the mathematical model of the drone used for the work published in this article can be found in [16]. The concept consisted of allowing rotor R3 to PT after rotor R1 stopped. The idea was to counter the yaw produced by rotors R2 and R4, and simultaneously manage the drone's roll and

pitch. In this article, however, rotors R2 and R4, including rotor R3, are given the capability to PT and change thrust by changing the FFTS. This is illustrated in [Figure 2](#) and [Figure 3](#). It means that the orientation of the thrust vectors for these rotors no longer points downwards in the z-axis direction.

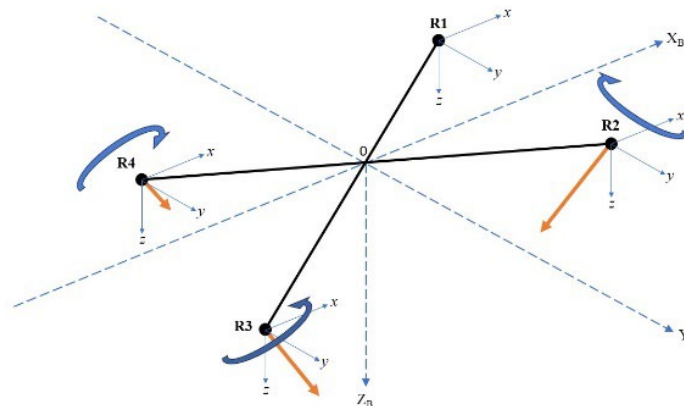


Fig.2 When R1 Failed and R2, R3 and R4 on Full Thrust Vectoring

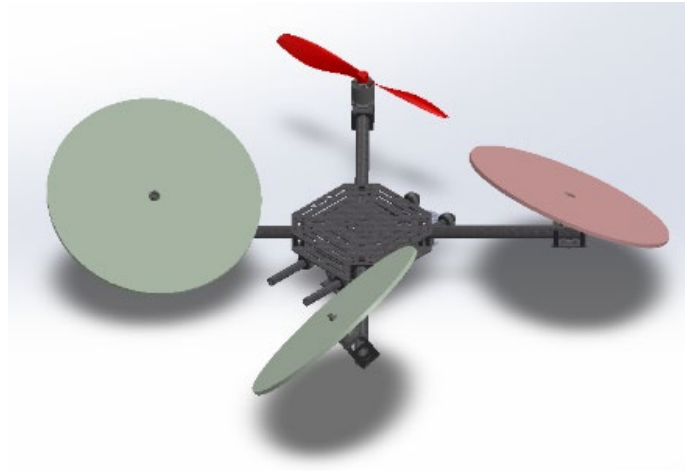


Fig.3 When R1 stops, and the other Rotors Thrust Vectoring

Figure 4 shows the thrust vector orientation for these rotors, where the pan (μ) and tilt (η) angles are defined. The components of the thrust vector in the X_B , Y_B , and Z_B directions can be resolved. Equations (1-3) propose the 'vector' to which the thrusts and torques for rotors R2, R3, and R4 act in the X, Y, and Z axis directions.

$$\text{Vector in } X - \text{direction: } \cos \eta \cdot \cos \mu \quad (1)$$

$$\text{Vector in } Y - \text{direction: } \cos \eta \cdot \sin \mu \quad (2)$$

$$\text{Vector in } Z - \text{direction: } \sin \eta \quad (3)$$

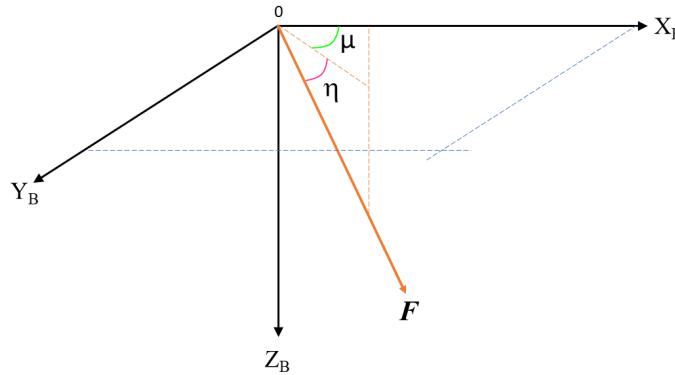


Fig.4 R3 Pan (μ) and Tilt (η) Thrust Vector Angles

B. Reinforcement Learning Strategy

The reinforcement learning approach is selected for this experiment given the agent's capability to learn from and adapt to an unfamiliar environment in which it operates [17], [18]. This is particularly beneficial in uncertain situations, such as when a QD encounters a rotor malfunction. Considering the intricacies and swift alterations in forces and moments that a damaged drone experiences, the RL method is deemed a promising strategy to produce favorable outcomes after sufficient training. The subsequent section details the key components of applying RL for adjusting FFTS, as well as the panning and tilting of rotors R2, R3, and R4 on the unsteady drone. The Deep Deterministic Policy Gradient (DDPG) agent is selected for this experiment. The following sections will discuss the environment, the reward

function, the termination function, and the agent's description.

C. The Observation Signals

In the unbalanced drone environment, the agent received nine observation signals constructed within the observation subsystem (see Figure 5). The signals were:

- Roll angle error, e_R
- Pitch angle error, e_P
- Yaw angle error, e_Y
- Roll rate, R_R
- Pitch rate, P_R
- Yaw rate, Y_R
- Descent rate, D_R
- Current altitude, C_A
- Altitude error, A_E

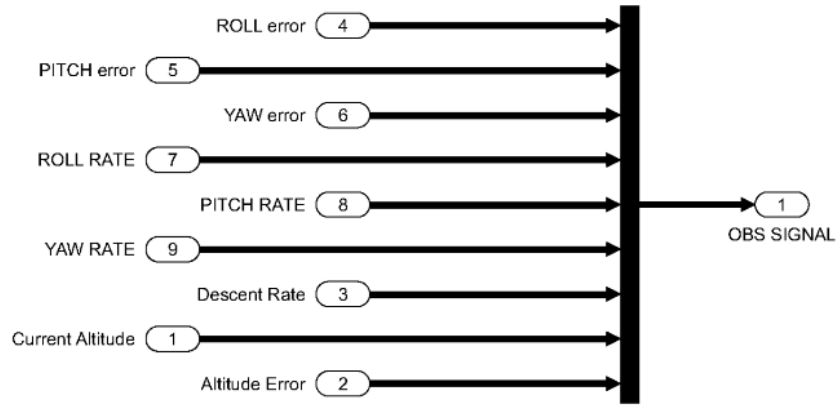


Fig.5 The Observation Block

The ‘error’ signals consist of the differences between the actual drone attitude angles and the commanded angles. For these experiments, all commanded angles are set to zero. The justification for this choice is that during a rotor failure incident, the operator's assumed decision is to immediately hover, which means maintaining zero roll, pitch, and yaw angles. All else within the environment remained the same, as discussed in [15].

D. The Reward Function

The following reward function, provided at every time step, t , is used to motivate the agent to achieve an optimal performance.

Figure 6 shows the implementation of the reward function in its Simulink block diagram. When R1 stops, the immediate commanded roll, pitch, and yaw given to the drone are zero degrees. However, as experience has shown, the drone will intuitively not be able to maintain this commanded attitude due to the remaining rotors still spinning. For this reason, in these simulations, the agent will be least rewarded for the difference between the commanded and the actual roll, pitch, and yaw angles.

Conversely, if the square of the roll error is small, $e_R^2 < 400$, then $H_t = 1$, rewarding the agent with 5 points. Otherwise, $H_t = 0$. This means that if the roll error is less than 20° , the agent will get the reward. A similar reward structure is applied if the square of the pitch error is small, $e_P^2 < 400$. Since many have reported the problem with yaw when a rotor failed, a greater reward is placed for the agent to attempt to reduce the uncontrollable yawing of the drone during this failure.

If the square of the yaw error is small, $e_Y^2 < 900$, then $Z_t = 1$, rewarding the agent with the highest mark of 8 points. Otherwise, $Z_t = 0$. This means that if the yaw error is less than

30° , the agent will get the reward. If the descent rate is higher than 1 m/s, the agent will be penalized by a small reward; however, if the descent rate is lower than or equal to 1 m/s, the agent will be rewarded with 10 marks.

Because this experiment aims to explore the possibility of the unbalanced drone maintaining the same altitude as when the R1 rotor failure occurred, the highest reward of 15 marks is allocated to the agent if the drone maintains an altitude error within 0.5m. A small reward is allocated to the agent if the altitude error exceeds 0.5m. Note that a heavy penalty is also placed if the conditions for termination are met. If the simulation is terminated because a termination condition is met, then $F_t = 1$, penalizing the agent by 10 points. Otherwise, $F_t = 0$.

E. The Action Signals

The agent interacted with the environment using a total of nine action signals for the QD. The signals are as follows.

- The pan angle of rotor R2, μ_{R2} .
- The tilt angle of rotor R2, η_{R2} .
- The pan angle of rotor R3, μ_{R3} .
- The tilt angle of rotor R3, η_{R3} .
- The pan angle of rotor R4, μ_{R4} .
- The tilt angle of rotor R4, η_{R4} .
- The FFTS value for rotor R2, $R2_{FFTS}$.
- The FFTS value for rotor R3, $R3_{FFTS}$.
- The FFTS value for rotor R4, $R4_{FFTS}$.

These action signals are then directed to their respective destinations. Figure 7 shows the FFTS signals sent to replace the previous constant values of ‘36’ used when all four rotors were working properly. Figure 8 illustrates the pan and tilt signals for rotor R2. The same is done for signals produced for rotors R3 and R4.

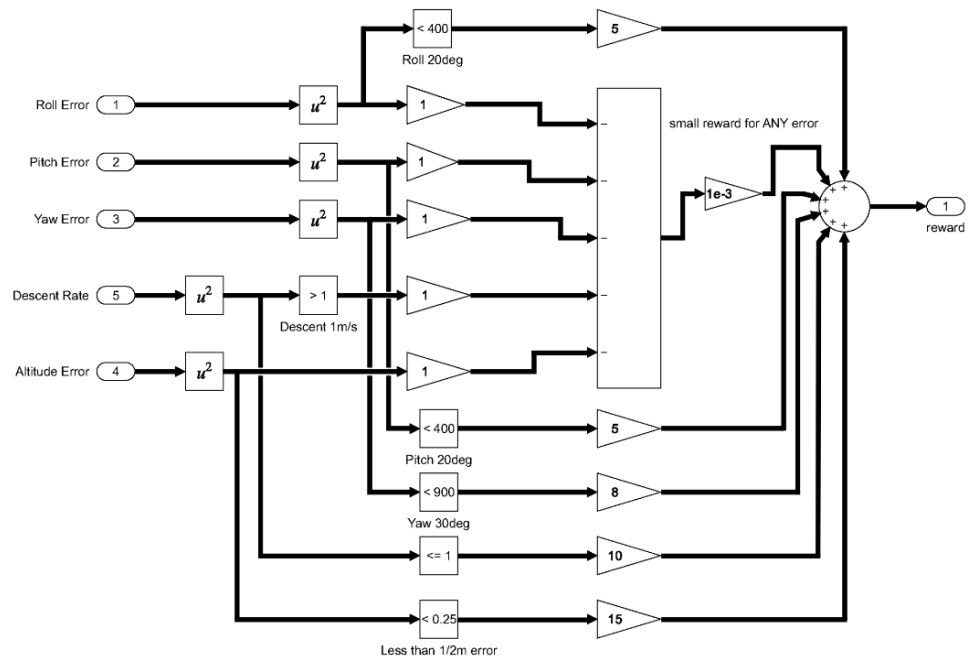


Fig.6 The Reward Block

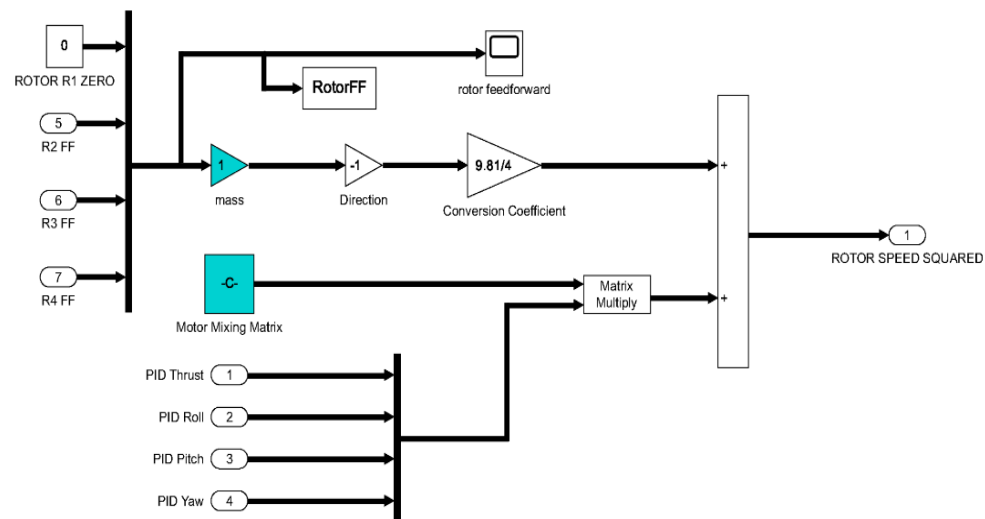


Fig.7 FFTS Signals for QD Received from the RL Agent

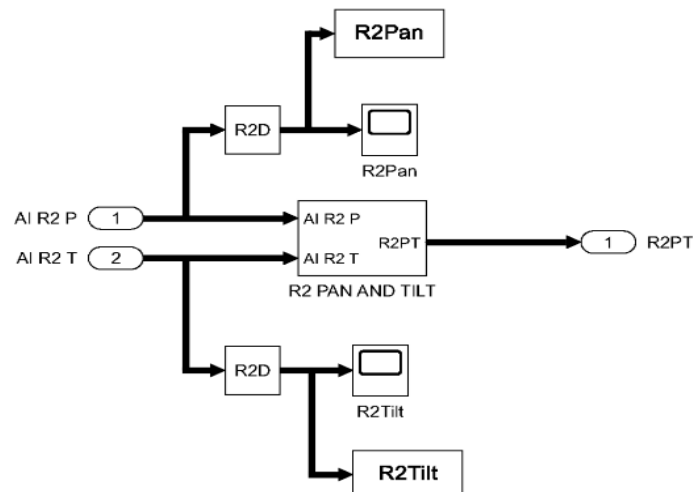


Fig.8 Pan and Tilt Signals for Rotor R2 Received from the RL Agent

F. The Termination Function

The agent's training is terminated when any of these four conditions are met.

If the magnitude of the roll error $e_R \geq 90deg$

If the magnitude of the pitch error $e_P \geq 90deg$

If the magnitude of the yaw error $e_Y \geq 90deg$

If the drone hits the ground

These conditions are incorporated into the Simulink model as a block diagram, as shown in Figure 9.

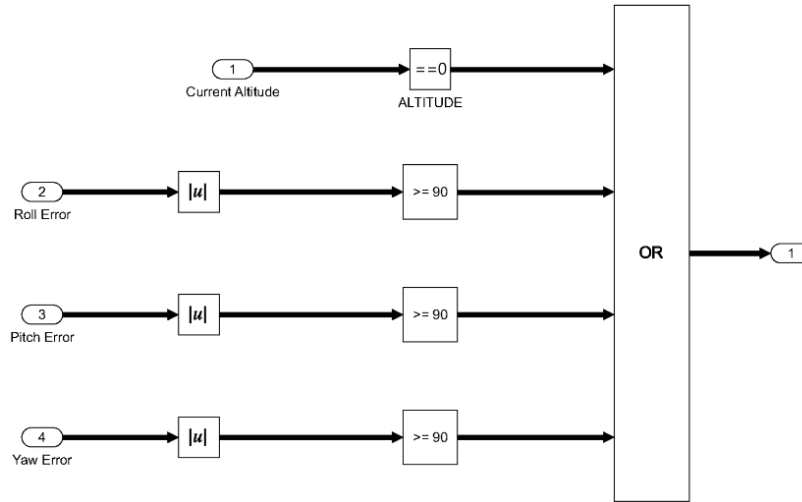


Fig.9 The Termination Block

G. The Agent

A Deep Deterministic Policy Gradient (DDPG) agent is selected to determine the control policy for rebalancing the disabled drone. The agent utilized a parametrized Q-value function approximator to assess the policy's value. The following training options were used. Training ran for 2000 episodes. Each episode lasted for at most 600 time steps, with a sample time of 0.1 seconds and a simulation duration of 60 seconds. No 'stop-training' instruction was set for the experiment if the agent received an average cumulative reward greater than a certain value. Instead, the instruction given to the agent was to stop the training session after 2000 episodes. The justification for this was to explore the results produced by the agent by allowing the agent to accumulate

as many reward points as possible within the allowed episodes.

H. Thrust And Torque Collection For Unbalanced Drone

Because of the new capabilities of rotors R2, R3, and R4, the components of thrusts and torques in X, Y, and Z directions for the QD had to be recalculated accordingly. Figure 10 and Figure 11 illustrate how the components are recalculated, regrouped, and then summed up to obtain the overall forces and moments experienced by the unbalanced drone. Lastly, the total values of forces and moments acting on the drone flight dynamics are directed to the 6DoF (Euler Angle) block to calculate the resultant flight dynamics for the unbalanced drone (see Figure 12, Figure 13 and Figure 14).

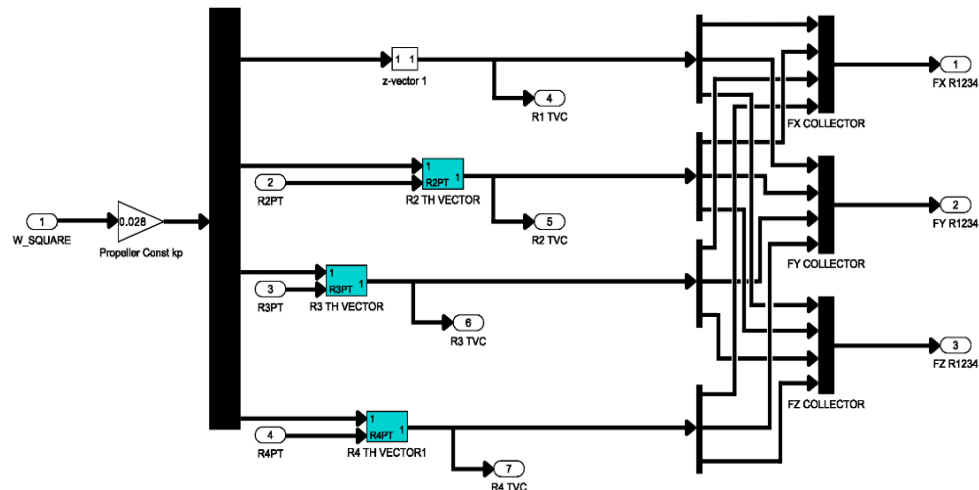


Fig.10 Rotor Thrust Components Recalculation

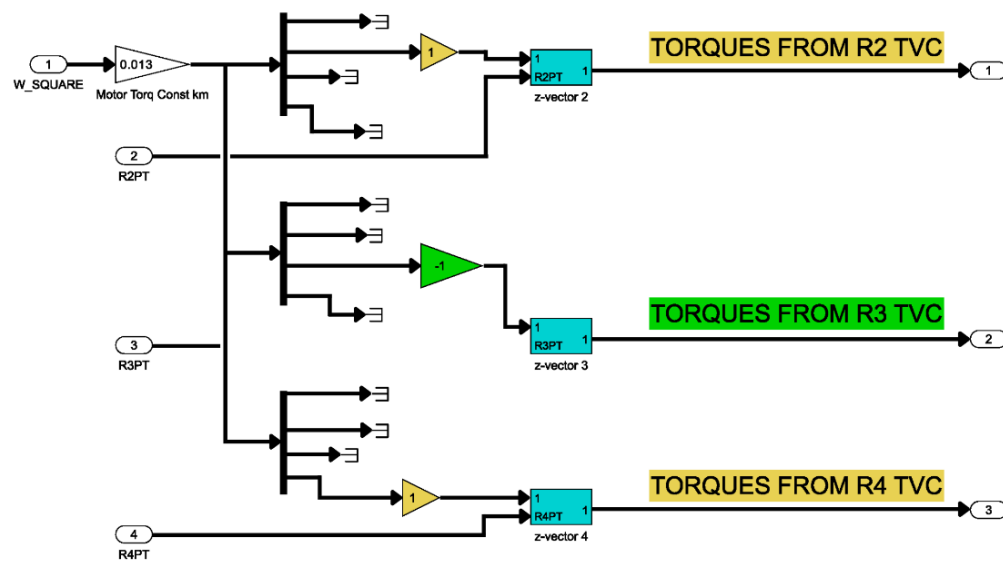


Fig.11 Rotor Torque Components Recalculation

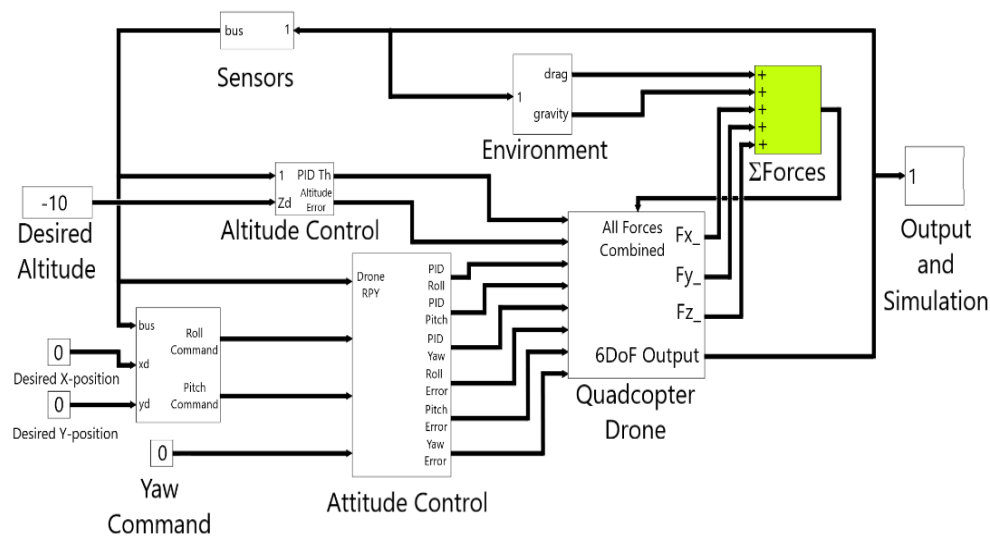


Fig.12 Recalculation of the Sum of Forces for QD

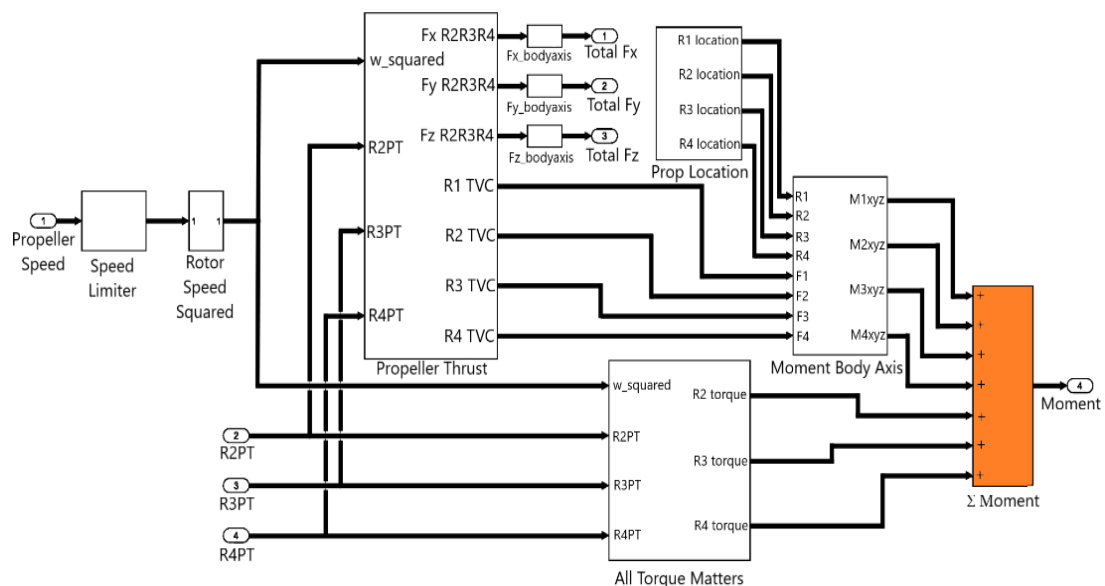


Fig.13 Recalculation of the Sum of Moments for QD

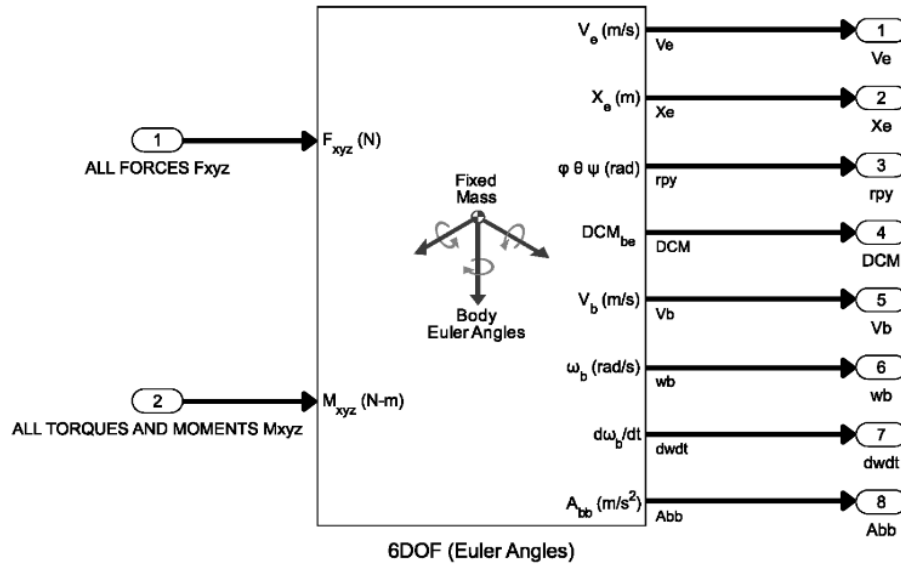


Fig.14 Inserting Sum of Forces and Moments into the 6DoF (Euler Angles) Block

III. RESULTS AND DISCUSSION

It is worth mentioning at this point that, by convention, the drone's altitude above ground, plotted by MATLAB, has negative values. However, in the next section, to assist in visualizing the drone's altitude trajectory, the 'sign' for altitude has been intentionally changed to positive to better visualize the drone's position above ground. The sign change has also been made to the drone's vertical velocity. These sign changes do not alter the overall results of the simulation and are made solely to aid visualization and explanation.

A. Drone Responses When All Rotors Were Working And When R1 Failed

To better understand the effectiveness of the proposed solution, the responses from the drone before a single rotor failure occurred are presented first in this section. Then, the responses of the drone when R1 failed are shown. Figure 15(a) shows the performance of the drone when all rotors were working correctly while hovering at 10m. The position of the drone remained the same at 10 meters above ground level, as indicated by the Z-axis position. The X and Y positions of the drone remained zero, as expected. When a problem with a rotor on a drone occurred during hovering, both the altitude and the attitude changed. The equilibrium of forces and moments was upset. Figure 15(b) shows what happened to the drone when rotor R1 completely stopped while hovering at 10m. Due to the failure, the drone began to lose altitude. The drone hit the ground in 1.86 seconds.

Figure 16(a) shows the drone's attitude during hovering. There was no visible rolling, pitching, or yawing displayed by the drone when all rotors were working properly. But when rotor R1 failed, the orientation of the drone changed, as shown in Figure 16(b). The drone rolled over to the port side

while the pitch and yaw showed large amplitude oscillations before hitting the ground. Before impact, the final roll angle registered at -10deg, the pitch angle at -14deg, and the yaw angle at -3deg.

Figure 17(a) shows the rates of roll, pitch, and yaw when all rotors were working properly during hovering. As expected, no visible change was observed. Figure 17(b) shows the roll, pitch, and yaw rates when rotor R1 stopped. All rates increased over time. During impact, the study revealed that the roll rate was -276deg/s, the pitch rate was -289deg/s, and the yaw rate was -137deg/s. At those high angular speeds, it was likely that the drone would suffer from some physical damage due to the impact. These simulation results are consistent with real-life experiences reported by operators during a single rotor failure incident.

Lastly, when all rotors were working correctly during hovering, all four rotors showed the same rotor speed, as shown in Figure 18(a). Figure 18(b) shows the rotor speed R1 equal to zero, simulating rotor failure. But due to this failed rotor, the PD controllers attempted to regain stability using rotors R2, R3, and R4. This is shown as erratic rotor speeds as the controllers attempted to maintain attitude and altitude. The observations made when all four rotors were operational confirmed that the drone had achieved an equilibrium state, suggesting that the sum of forces and the sum of moments were zero during hovering. When a single rotor failed, the changes in attitude and altitude illustrated what needed to be improved, should there be a way to rescue the drone. Ideally, the method should improve attitude changes, which in turn would enhance their rates. Preferably, the solution restores the drone's hovering capability or at least satisfies the FQ criteria mentioned earlier. The results in this section will be used again later to compare the effectiveness of the solutions produced by the DDPG agent.

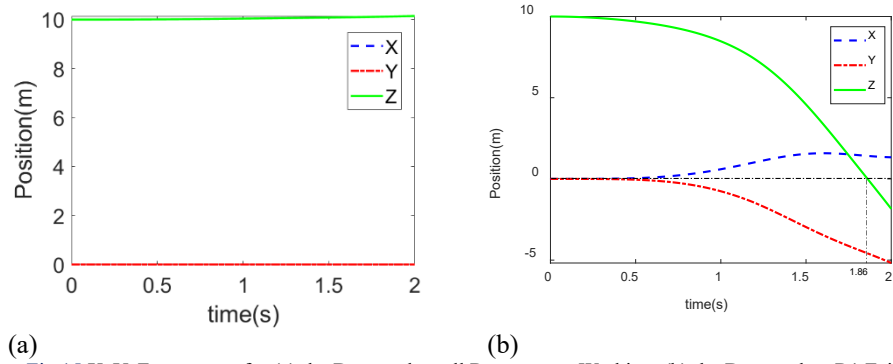


Fig.15 X, Y, Z responses for (a) the Drone when all Rotors were Working, (b) the Drone when R1 Failed

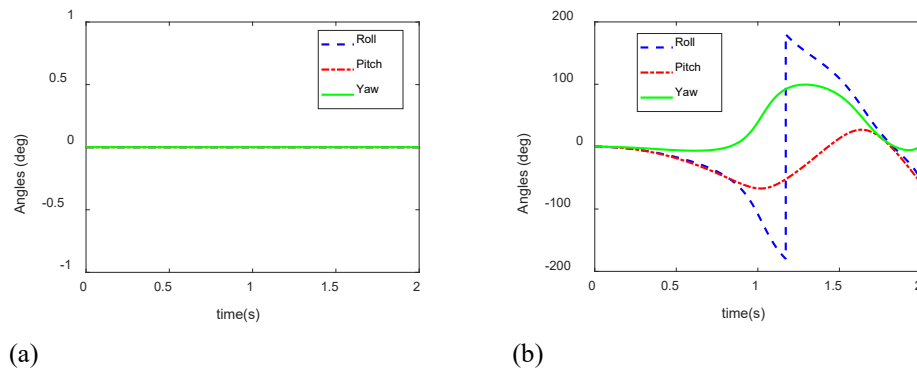


Fig.16 Roll, Pitch, and Yaw Responses for (a) the Drone when all Rotors were Working, (b) the Drone when R1 Failed

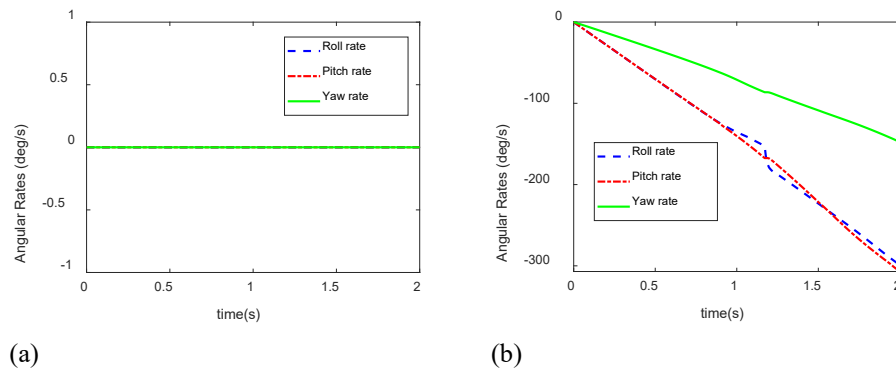


Fig.17 Roll, Pitch, and Yaw Rate Responses For (A) the Drone When all Rotors were Working, (B) the Drone When R1 Failed.

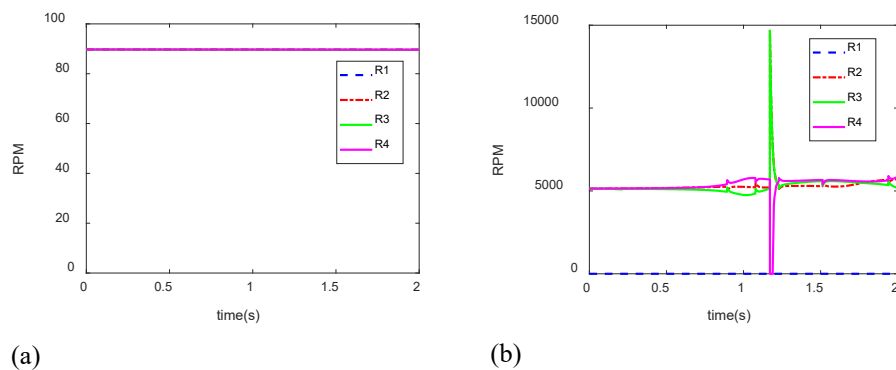


Fig.18 Rotor Speed Responses for (a) the Drone when all Rotors were Working, (b) the Drone when R1 Failed.

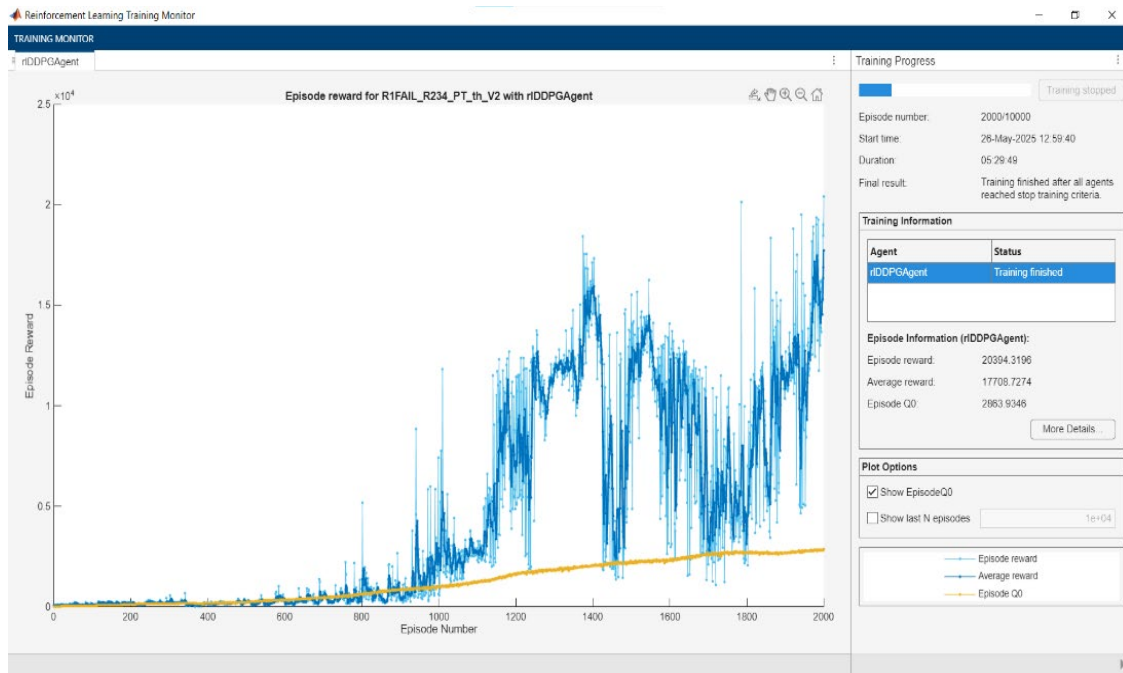


Fig.19 DDPG Agent Training Lasting 2000 Episodes

B. Drone Responses When RL Failed And Reinforcement Learning Agent Controlled Thrust Vectoring

The RL agent training ran for 2000 episodes. As mentioned in the previous section, no ‘stop-training’ command was programmed to stop the training if the agent received a certain value of reward. The intention of the training was to let the agent accumulate as much experience as possible by claiming rewards. Figure 19 shows a snapshot of MATLAB’s Training Monitor after completing 2000 training episodes. To interpret the plot, Figure 20 was included, which contained the Episode Q0 and Average Reward trajectory over Episode Number. By studying Figure 20, the agent demonstrated a healthy learning process in the first 1000 episodes, with the

critic's expectation aligning with the accumulation of average rewards received. The agent's exploration of the problem to find the right solution for the unbalanced drone appeared to be on the right path. From the 1000th episode, the average reward increased significantly compared to the critic's expectation. Although this seemed to suggest that the training should proceed to exploitation, this research did not proceed in that direction at this time. Instead, it allowed the training of the agent to continue for another 1000 episodes, enabling the agent to gain as much experience as possible to solve the problem. The final Episode Q0 value obtained was 2853.9. The Episode Reward and Average Reward values were 20394.3 and 17708.7, respectively.

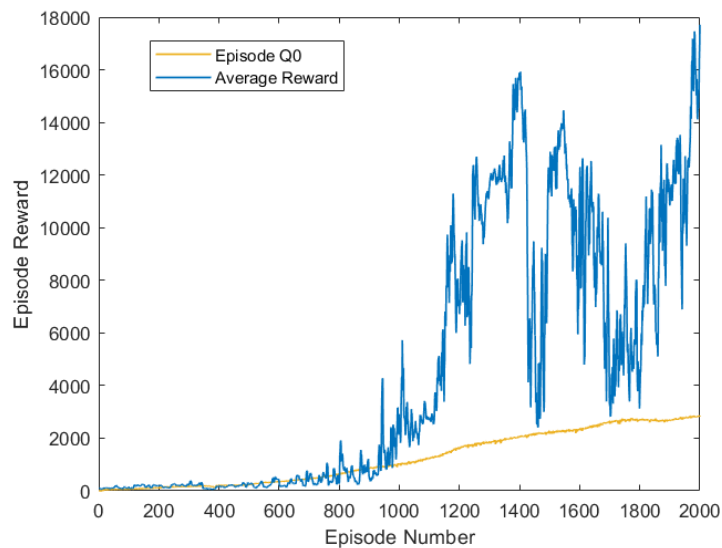


Fig.20 Plotting Episode Q0 and Average Reward over Episode Number

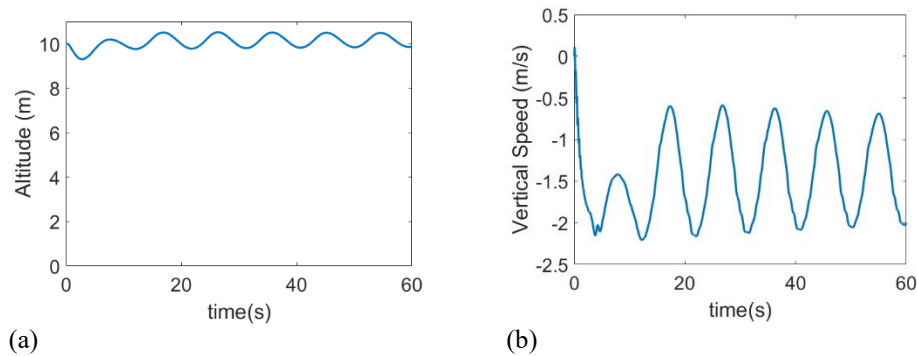


Fig.21 (a) Altitude and (b) Vertical Speed Responses Over Time

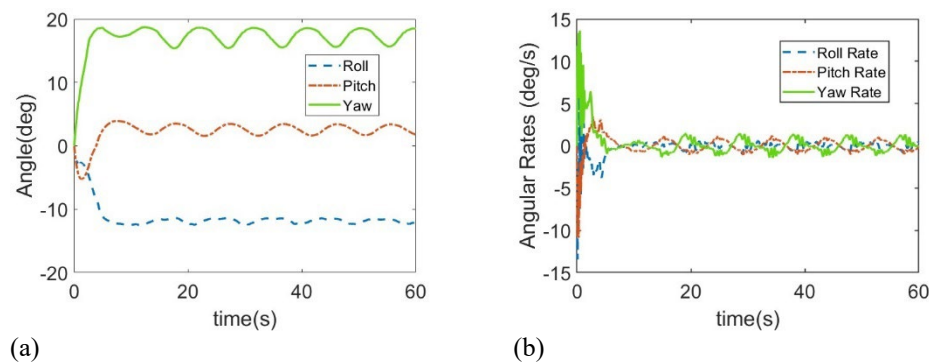


Fig.22 (a) RPY and (b) RPY Rate Responses Over Time

Shown next are plots of the drone responses with respect to time, after rotor R1 completely stopped, leaving only rotors R2, R3, and R4 thrust vectoring using the control policy obtained through reinforcement learning. When the RL controllers assisted, it was observed that the drone did not crash, as shown in Figure 21(a). In fact, the drone showed relatively small changes in altitude from the assigned hovering altitude. The lowest altitude registered by the unbalanced drone was 9.3m, and the highest altitude registered after the failure was 10.5m. That's a mere fluctuation of ± 5 to 7%. This is a significant improvement from what the drone displayed in Figure 15(b). More importantly, this shows that the unbalanced drone has satisfied the FQ criteria requirements (i) and (ii) mentioned in Section 3 in this article. It is also interesting to observe the vertical speed response after the RL controller participated in the rescue. After the failure, the drone dropped with a vertical speed of 2.15 m/s, but the speed decreased quickly and oscillated for the rest of the flight. The minimum drop speed recorded in the oscillation was 0.6m/s.

When the drone was hovering before rotor R1 stopped, the roll, pitch, and yaw angles were zero. After the failure occurred, the RL agent adjusted the pan and tilt of rotors R2, R3, and R4 to regain stability for the drone. In addition, the feedforward thrusts for R2, R3, and R4 were also adjusted. The effect on the drone's roll, pitch, and yaw can be seen in Figure 22(a). The unbalanced drone rolled slightly to the port

side and remained there at approximately 11.3 degrees. The drone pitched forward as expected since the loss of thrust from R1 caused a sudden imbalance. However, this was corrected by the RL controller. The pitching settled in a small oscillation with a mean value of 2.3 degrees, which is very small. One of the most notorious effects of a single rotor failure, as reported by many authors, is the uncontrolled yawing of the drone. None of that was seen on this rebalanced drone. Figure 22(a) shows that the drone was affected by the sudden loss of rotor R1 by yawing immediately to the right. This is expected due to the torque imbalance. However, the maximum yawing recorded was only 18.7 degrees, after which the yaw oscillated and settled into a small oscillation with an amplitude of approximately 17 degrees. These observations suggest that the FQ requirements (iii), (iv), and (v) have been satisfied, too.

Figure 22(b) shows the rates of roll, pitch, and yaw after R1 stopped. Initially, following the failure, a significant spike in roll rate, pitch rate, and yaw rate was observed. Among the roll, pitch, and yaw rates, the highest recorded was the yaw rate, but at only 13.6 deg/s, which is relatively small. The mean value recorded for yaw rate throughout the flight was only 0.3deg/s. The pitch rate had an equally small mean value of 0.01 deg/s, and the roll rate mean value was only -0.18deg/s. These results suggest that the QD no longer behaved erratically in a rapid manner.

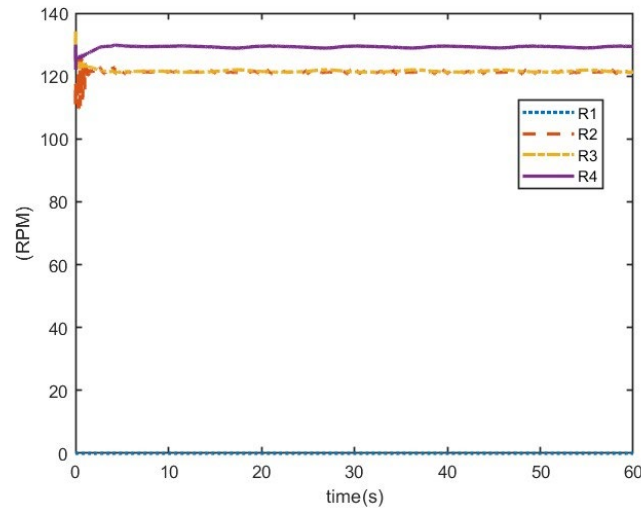


Fig.23 Rotor Speeds Regulation After R1 Failed

Figure 23 shows the speed of the rotors for this unbalanced drone. Rotor R1 was not spinning, but rotors R2, R3, and R4 were working. The agent proposed an interesting set of speeds for the rotors that were still working. R4 spun faster than R2 and R3, spinning mostly at 129rpm, but regulated, throughout the flight. Although R2 and R3 appeared to spin at the same speeds, they were not when observed closely.

Although both spun at different regulated speeds, the average speeds for both were approximately 121rpm throughout the flight. From Figure 23, the results confirmed that the QD was flying with three working rotors only. Next, the orientations of the rotors R2, R3, and R4, along with their respective FFTS, will be discussed.

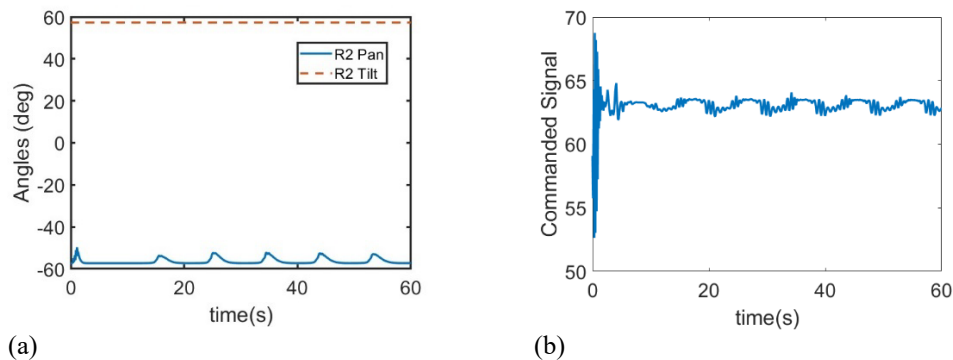


Fig.24 (a) R2 Pan and Tilt Angles, and (b) R2 FFTS Signal from RL Agent

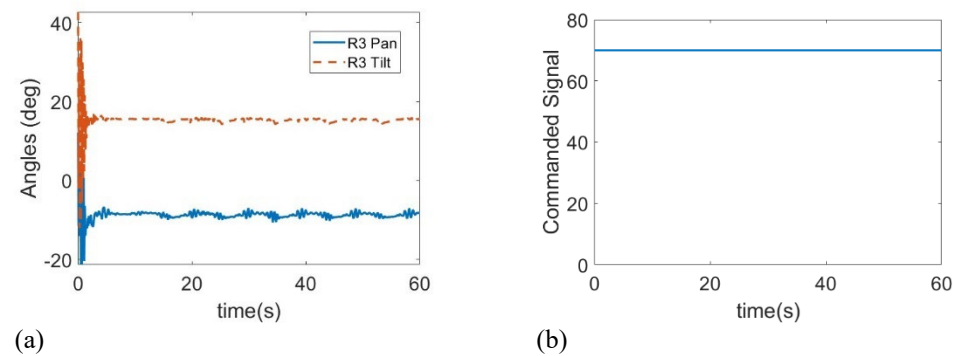


Fig.25 (a) R3 Pan and Tilt Angles, and (b) R3 FFTS Signal from RL Agent

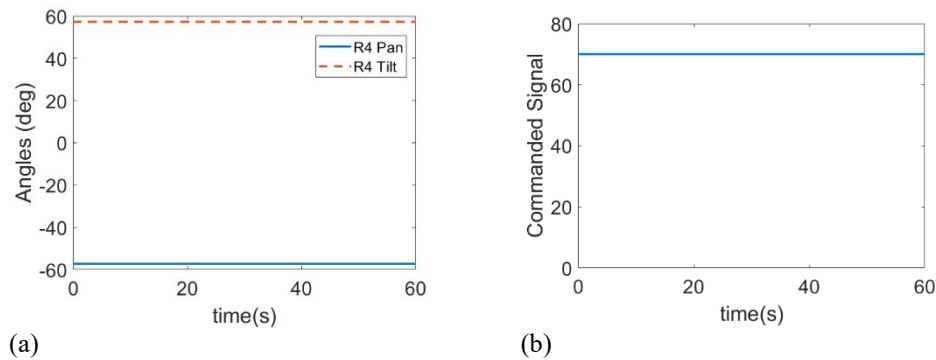


Fig.26 (a) R4 Pan and Tilt Angles, and (b) R4 FFTS Signal from RL Agent

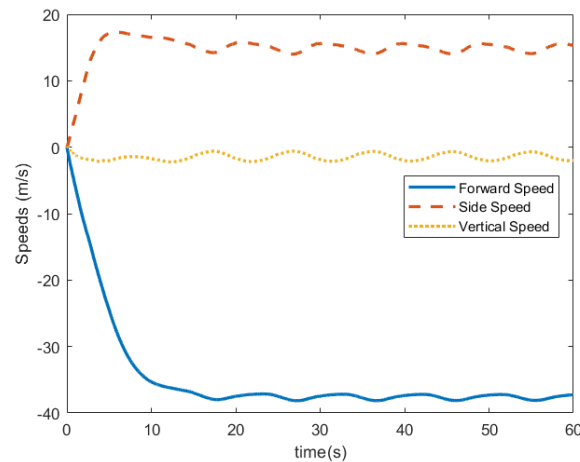


Fig.27 Forward, Side, and Vertical Speeds for the Drone with Full Thrust Vector Capabilities

The RL agent was tasked with finding the corrective actions to maintain altitude and the attitude of the unbalanced drone by producing nine action signals. Each remaining working rotor had access to 3 action signals from the agent. For rotor R2, the agent could change the pan and tilt angles, which in turn affected the rotor thrust angles and, consequently, the torque angles. The agent was also allowed to change the feedforward throttle setting for rotor R2 in its effort to maintain altitude and attitude. The same was done by the agent for rotors R3 and R4. Figure 24(a) and (b) show the results obtained from the DDPG agent for rotor R2. Figure 25(a) and (b) show the actions from rotor R3. Figure 26(a) and (b) show the actions from rotor R4. It is worth mentioning that it was the collective effort of all three rotors that enabled the unbalanced drone to perform such a feat successfully. At this stage of the experiment, no actuator dynamics have been incorporated into the model to simulate the servos for these PT capabilities. The information obtained serves as a guide for selecting suitable actuators for future flight testing.

Figure 27 describes the speeds of the drone in the X, Y, and Z directions while the drone attempted to maintain altitude

and attitude stability. The drone moved backwards the moment rotor R1 failed. This made sense intuitively since the aft rotors R3 and R4 would have too much thrust, and the forward rotor R2 alone could not compensate for the loss of rotor R1 to maintain balance. This is confirmed by Figure 22(a). Although the initial pitch down shown was most likely the immediate reaction from the drone after R1 thrust was lost, the drone continued to pitch up soon after to maintain balance at 10m altitude. The agent adjusted the rotors, causing the drone to move backward at an almost steady speed of 37.3 m/s. The drone did not move in a straight line backwards, though. It had a side speed of approximately 15.4 m/s. The vertical speed response in Figure 27 has been discussed previously in Figure 21(b).

To appreciate the distance travelled by the unbalanced drone while the agent attempted to maintain altitude and attitude, Figure 28 plots the 3D trajectory throughout the simulation time. In 60 seconds, the drone traveled 2.3 km backward and 225.1 m to the right from the point where rotor R1 completely stopped. The final altitude recorded at the end of the simulation was 9.8m, short of 20cm from the commanded hovering altitude. Not once did the drone touch the ground.

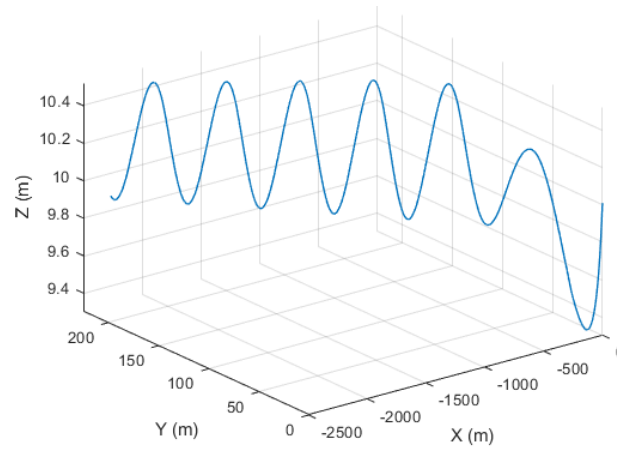


Fig.28 Travel Distance of the Unbalanced Drone with Full Thrust Vectoring Capabilities

TABLE I PARAMETERS UTILIZED WHEN TESTING DRONES WITH ACTIVE PAN AND TILT OF REMAINING ROTORS AFTER R1 FAILED

Parameters	Original QD Settings	QD with RL Agent
Feedforward (Throttle) setting:	[36 36 36 36]	Active
Commanded altitude:	10m	10m
Altitude Proportional (P), Derivative (D) gains:	P = 1, D = 2	P = 1, D = 2
Commanded (X, Y) location:	(0,0)	(0,0)
Commanded roll angles:	0deg	0deg
Roll Proportional, Derivative gains:	P = 1, D = 2	P = 20, D = 30
Commanded pitch angle:	0deg	0deg
Pitch Proportional, Derivative gains:	P = 1, D = 2	P = 20, D = 30
Commanded yaw angle:	0deg	0deg
Yaw Proportional, Derivative gains:	P = 1, D = 2	P = 60, D = 10
R2, R3, R4 Pan and Tilt angles:	Fixed: $\mu = 0$, $\eta = 90\text{deg}$	$\mu = \text{active}$, $\eta = \text{active}$

TABLE II RESULTS FROM EXPERIMENTS WHEN THE R1 ROTOR COMPLETELY STOPPED AT 10M ALTITUDE

Parameters	QD without RL	QD with RL
Time to reach ground (s):	1.85	Never
Roll	Rolled Over	-12.5 (11.8s)
Pitch	-70	-5.3(1.4s)
Yaw	103	18.7 (12.4s)
Angle During Impact (deg):		
Roll	10	None
Pitch	7	None
Yaw	6	None
Maximum Attitude Rates (deg/s):		
Roll Rate	-258	-13.4 (0.1s)
Pitch Rate	-271	-10.8 (0.3s)
Yaw Rate	-128	13.6 (0.5s)
Attitude Rates at Touch Down (deg/s):		
Roll Rate	-258	None
Pitch Rate	-271	None
Yaw Rate	-128	None
Vertical Speed During Impact (m/s):	14	None
Final Landing Location (m):	(1.4, -4.3, 0.0)	(-2267.1 225.1 -9.8)

An explanation follows why the drone drifted so far backward instead of hovering at a point. When the agent was trained, a set of reward functions was used by the agent to accumulate as many rewards as possible. For this experiment, the agent was encouraged through rewards if the actions taken resulted in a small change in altitude between the actual altitude and the commanded altitude. Conversely, a penalty was imposed if there was any error between actual altitude and the commanded altitude by rewarding little. The same was done for roll, pitch, and yaw, along with their respective rates. However, no reward was designed to ‘entice’ the agent to remain at X and Y locations. In other words, the agent was encouraged to maintain Z-location, but not X- nor Y-locations. The agent therefore decided to balance the drone without imposing any restrictions on itself, allowing it to move in both X and Y directions while collecting as many rewards as possible. This was the reason why the agent and the drone did not limit the trajectory in the X and Y directions.

IV. CONCLUSIONS AND FUTURE WORK

The results from the study revealed that a hovering quadcopter drone can manage attitude and altitude changes by utilizing the remaining three rotors when a single rotor stops, provided the rotors can be panned and tilted, and the throttle setting can be adjusted individually. The study used the Reinforcement Learning method to train a DDPG agent to produce a control policy to manage the action signals. The suggested control policy produced an unexpected outcome, however. To rebalance the forces and moments, the drone moved backward continuously. Upon investigation, it was found that the agent was not restricted from moving in the X- and Y-axis directions (forward-backwards, port-starboard). This restriction could be set in the reward function, which will be studied in the future. Future studies will also include exploitation exercises to improve the critic’s expectations of the agent’s actions.

Declaration of Conflicting Interests

The author declares no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author received no financial support for the research, authorship, and/or publication of this article.

Use of Artificial Intelligence (AI)-Assisted Technology for Manuscript Preparation

The author(s) utilised generative AI technologies solely to improve language clarity and refine grammar. No images were generated, manipulated, or altered using AI-assisted technologies.

REFERENCES

- [1] G. Wild, J. Murray, and G. Baxter, “Exploring civil drone accidents and incidents to help prevent potential air disasters,” *Aerospace*, vol. 3, no. 3, pp. 1–11, 2016, doi: <https://doi.org/10.3390/aerospace3030022>.
- [2] Z. R. Khavas, A. Majdi, S. Ahmadzadeh, and P. Robinette, “Human trust after drone failure: Study of the effects of drone type and failure type on human-drone trust,” in 2023 20th International Conference on Ubiquitous Robots (UR), Honolulu, Hawaii, 2023, pp. 685–692, doi: <https://doi.org/10.1109/UR57808.2023.10202489>.
- [3] Y. C. Yeh, “Triple-triple redundant 777 primary flight computer,” in 1996 IEEE Aerospace Applications Conference Proceedings, Aspen, CO, USA, 1996, vol. 1, pp. 293–307, doi: <https://doi.org/10.1109/AERO.1996.495891>.
- [4] C. M. Pittman, *Taming Your Amygdala: Brain-Based Strategies to Quiet the Anxious Mind*. Eau Claire, WI, USA: PESI Publishing, Inc., 2022.
- [5] G. K. Foulas and G. C. Karras, “A survey on fault diagnosis and fault-tolerant control methods for unmanned aerial vehicles,” *Machines*, vol. 9, no. 9, pp. 1–34, 2021, doi: <https://doi.org/10.3390/machines9090197>.
- [6] H. Shraim, A. Awada, and R. Youness, “A survey on quadrotors: Configurations, modeling and identification, control, collision avoidance, fault diagnosis and tolerant control,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 33, no. 7, pp. 14–33, 2018, doi: <https://doi.org/10.1109/MAES.2018.160246>.
- [7] M. Saied, H. Shraim, and C. Francis, “A review on recent development of multirotor UAV fault-tolerant control systems,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 39, no. 9, pp. 146–180, 2024, doi: <https://doi.org/10.1109/MAES.2023.3327697>.
- [8] J. Guo, J. Qi, and C. Wu, “Robust fault diagnosis and fault-tolerant control for nonlinear quadrotor unmanned aerial vehicle system with unknown actuator faults,” *International Journal of Advanced Robotic Systems*, vol. 18, no. 2, 2021, doi: <https://doi.org/10.1177/17298814211002>.
- [9] S. Mallavalli and A. Fekih, “Adaptive fault tolerant control design for actuator fault mitigation in quadrotor UAVs,” in 2018 IEEE Conference on Control Technology and Applications (CCTA), Copenhagen, Denmark, 2018, doi: <https://doi.org/10.1109/CCTA.2018.8511427>.
- [10] S. Mallavalli and A. Fekih, “An SMC-based fault tolerant control design for a class of underactuated unmanned aerial vehicles,” in 2018 4th International Conference on Control, Automation and Robotics (ICCAR), Auckland, New Zealand, 2018, doi: <https://doi.org/10.1109/ICCAR.2018.8384661>.
- [11] L. Xulin and Y. Guo, “Fault tolerant control of a quadrotor UAV using control allocation,” in 2018 Chinese Control and Decision Conference (CCDC), Shenyang, China, 2018, doi: <https://doi.org/10.1109/CCDC.2018.8407422>.
- [12] W. Chung and H. Son, “Fault-tolerant control of multirotor UAVs by control variable elimination,” *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 5, pp. 2513–2522, 2020, doi: <https://doi.org/10.1109/TMECH.2020.2982436>.
- [13] M. D. Dehkordi and M. Danesh, “Fuzzy control allocation of a positionable rotor quadrotor based on log-barrier optimization and propulsion system fault toleration,” *IET Control Theory & Applications*, vol. 18, no. 9, pp. 1176–1190, 2024, doi: <https://doi.org/10.1049/cth2.12653>.
- [14] Z. Zaludin, “Drone fault tolerance strategy by allowing remaining rotor to pan and tilt,” *International Journal of Advanced Mechatronic Systems*, vol. 11, no. 2, pp. 102–110, 2024, doi: <https://doi.org/10.1504/IJAMECHS.2024.139180>.
- [15] Z. Zaludin, “Enhancing drone performance: Using reinforcement learning for active pan and tilt rotor fault tolerance strategy,” *International Journal of Advanced Mechatronic Systems*, vol. 12, no. 2, pp. 132–146, 2025, doi: <https://doi.org/10.1504/IJAMECHS.2025.145731>.
- [16] Z. Zaludin, “Fault tolerance conceptual strategy for a quadcopter drone with rotor failure,” *Asian Review of Mechanical Engineering*, vol. 12, no. 2, pp. 1–14, 2023.
- [17] MathWorks, “Reinforcement Learning Toolbox: User’s Guide (R2024b),” accessed Dec. 27, 2024. [Online]. Available: <http://www.mathworks.com>.
- [18] A. Nandy and M. Biswas, *Reinforcement Learning: With Open AI, TensorFlow and Keras Using Python*. Berkeley, CA: Apress, 2017.