Reinforcement Learning for Aircraft Control and Solving Faults Scenarios

By

Lijing Zhou

MSc Aerial Robotics Dissertation

Department of Engineering Mathematics
UNIVERSITY OF BRISTOL

A MSc dissertation submitted to the University of Bristol
in accordance with the requirements of the degree of
MASTER OF SCIENCE IN AERIAL ROBOTICS in the
Faculty of Engineering.

September 12, 2022

Declaration of own work

I declare that the work in this MSc Aerial Robotics dissertation was carried out following the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific references in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others are indicated as such. Any views expressed in the dissertation are those of the author.

Lijing Zhou

August 23, 2022

Acknowledgement

In this thesis, I have come to understand how physical models and algorithms work in tandem. I have come to understand how we can use algorithms to solve the problems we encounter. With the completion of this thesis, my Masters' career has officially come to an end, and I really want to acknowledge a few people who have helped me tremendously during this short period of time. Firstly, I would like to thank Sebastian East, a professor at the University of Bristol, for his constant support, help, and guidance throughout the whole research process. He provided me with the direction of my research and ideas for solving several key problems. His help has fundamentally given me a deeper understanding of how to apply reinforcement learning to physical models. In the second place, I want to thank Liam Fletcher, the Ph.D. at the University of Bristol, who has given me crucial help with details and has actively given me ideas on how to solve problems when I have encountered them, and who has also helped me a lot with code. Finally, I want to thank Alexander Quessy, the Ph.D. at the University of Bristol, who taught me a lot about drone dynamics, and I learned a lot from his thesis. To this day, I still miss our meetings on Wednesdays and Thursdays. I miss the discussions that we have and the progress I made over and over again. I would like to thank you all sincerely for your help, and the time we spent together. This is all my thanks to you.

Abstract

There is growing interest in the ability of drones to cope with problems in the event of an emergency, such as high wind speed, a broken aileron, or an engine failure on a quadcopter. In reality, some drone malfunctions can't be avoided. Effective faults are small and can be fixed before more damage occurs. However, more failures occur during the flight, which will often affect the flight and mission of the UAV. And at the controls, it's hard to artificially repair and adjust the drone. The advanced system still needs improvement in different fault scenarios. Reinforcement Learning (RL), a form of machine learning, can generate controllers that are robust to changes in dynamics through methods such as domain randomization. In addition, RL has recently achieved a wide range of successes in many areas, including robot control. For example, many classical reinforcement learning algorithms have been applied to the control of drones. Now, people have developed deep learning theories and techniques to deal with the problems we encounter.

Number of words in the dissertation: 10046 words.

Contents

List of Tables

1 Introduction

1.1 Background and Motivation

Even the aviation industry's accident rate fell 36% between 2008 and 2019. However, In-flight loss of control is still the main reason leading to the 61% of flight accidents from 2009 to 2018 [1]. Therefore more attention has been paid to how to remain Unmanned Aerial Vehicles (UAV) stable in some changing conditions or extreme emergencies.

In addition, some index of the aircraft control model is not easy to adjust when the environment or the condition of the aircraft change.

Some deep reinforcement learning methods like Soft Actor-Critic (SAC), Trust Region Policy Optimization (TRPO), Proximal Policy Optimization (PPO), Deep Deterministic Policy Gradient (DDPG), Deep Q-Networks (DQN), World Models, and AlphaZero have been applied in our lives [2].

In this research, the project aims to apply reinforcement learning to build and adjust the dynamic model of aircraft to solve the problem that one of the rotors of the quadrotor has a rate of random output. Here is growing interest in the ability of drones to cope with problems in the event of an emergency, such as high wind speed, a broken aileron ,or an engine failure on a quadcopter. How to keep the UAV stable in an emergency, and how to control its parameters, such as rotors to adjust the speed of the rotors to maintain stability, are still a challenge to us.

In the first place, there will be a flight dynamics model of a quadrotor built-in simulation. The dynamics and theory of the drone will be modeled. In this heel model, there will be the dimensions of the drone, the weight, and the wind resistance coefficient of the drone in flight, etc. In this model, there will be 4 inputs, including the output power of the propellers of the drone, and 12 outputs, including information about the speed and position of the drone.

Secondly, the model will be tested as the normal input of the rotors and the results like the position and velocity information will be observed at the same time. There are 4 inputs of the output power of the propeller of the drone, different kinds of data combinations will be tested to see the 12 outputs, including the continuous change of position and speed of the drone.

Thirdly, the drone has some basic movement modes, including hovering i.e. stabilization in the air, ascent, and descent, left, right, back and forth movement, and rotation. The motion of the UAV is controlled by controlling the speed of the four propellers, which is the torque output of the four propellers. We traditionally use a PID controller to control the basic motion of the UAV. This research has tried to use reinforcement learning to create the motion models instead of a traditional PID controller. The models are trained repeatedly, and the number of training sessions is increased to check if the models meet the requirements.

Finally, flight faults will be modeled by modifying the simulator and the reinforcement learning will contribute to being robust to the fault. There are many scopes of experimentation, like the different types of faults of the rotor and their corresponding processing methods. The model of reinforcement learning can help the aircraft automatically learn to adjust their parameters to judge the environment and adapt to the new environment to remain the drone stable.

1.2 Aims and Objectives

- **Aims**: To investigate and exploit the reinforcement learning methods in the fight control of the UAV and the solution when the UAV meets faults.
- **Objectives**:
    1. Research the reinforcement learning theories and techniques and their application to aerial robotics.
    2. Choose and build an appropriate aircraft model in simulation. Modeling is based on rigorous dynamics equations for UAVs
    3. Using reinforcement learning to build a model for the basic control of the UAV.
    4. Come up with a simple reinforcement learning scenario such as flying straight and

remain a certain altitude. In this environment, we will train and test the performance of the UAV.

5. Use a baseline reinforcement learning algorithm/framework such as the Proximal Policy Optimization in Stable baselines and Monte-Carlo Learning to train a controller.
6. Initially train reinforcement learning controllers using simple domain randomization techniques like the control authority of elevator changes at each episode
7. Evaluate controllers in simulation, comparing and contrasting performance by the reward or variance.

2 Literature Review

2.1 Quadcopter Introduction

A quadrotor drone is, literally, a drone with four rotors. The four propellers are equidistant from the centre of mass of the drone. A quadrotor drone flies by means of the lift generated by the four rotors and by controlling the output power of the four rotors in order to hover, ascend and descend, fly backwards and forwards and rotate the drone. Based on their simple structure and dynamics, quadcopter drones are widely used in our daily live [3].

Compared to fixed-wing drones, quadrotor drones are fast, manoeuvrable, and can perform complex tasks in the air. The basic control principle of a quadrotor UAV is to control the speed of the UAV's rotors to vary the forces on the UAV and the UAV's own torque on its own axis, thereby controlling the position of the UAV and the UAV's attitude.

Quadcopter controller design has been investigated many times [4]. Here we try to control the UAV using reinforcement learning to control the speed of the UAV's rotors in the hope of achieving a basic control action. At the same time, we consider a situation where the propeller of a UAV fails, and the type of failure varies, and so does the control model derived from reinforcement learning.

2.2 Reinforcement Learning Background

Reinforcement learning is a framework of Machine Learning. Reinforcement Learning aims to make the actions with the given environment to achieve the max numerical reward signal [5].

Reinforcement learning refers to two important factors, the agent and the environment [6]. The agent will respond to each change in the environment. The action of the agent aims to maximize its cumulative reward. The agent relies on the feedback of the environment called reward and creates the most reward called to return for the next step. The agent always contains plenty of computational models composed of multiple processing layers [7].

The agent always observes the reward in the form of a real-valued vector, matrix, or higher-order tensor with a mathematical way to show the states. How to respond to the reward is called the policies thought as the core of reinforcement learning. We explain how to analyze the data and maximize the expected performance [8]. The optimal policy always creates the maximum expected return. In conclusion, the agent will take action according to the environment, and how to make the decision is based on how to achieve the maximum reward. The general description of the policy of returning the maximum reward can be expressed by the equation 2.1). The π is on behalf of the policy, and R means the return of the available policies.

$$\pi^* = \arg\max_{\pi} \mathbb{E}[R \mid \pi] \tag{2.1}$$

The figure below 2.1) will give a brief description of the simple iterative process between the agent and the environment.

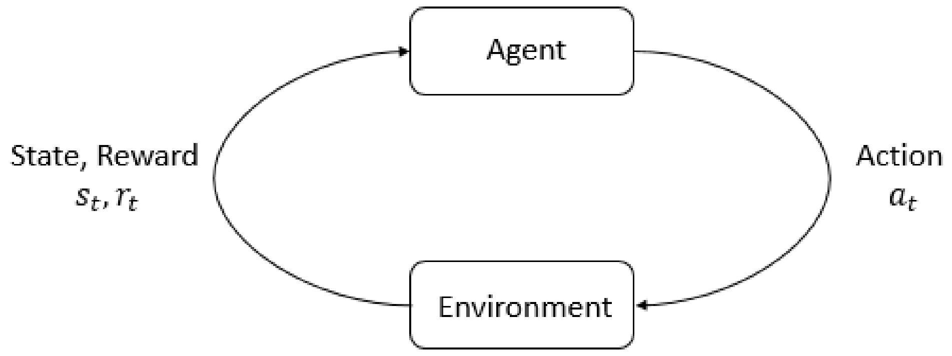


Figure 2.1: Agent-environment interaction loop.

There are some classic reinforcement learning methods like Deep Deterministic Policy Gradient (DDPG) [9] algorithm, which is based on the Deterministic Policy Gradient (DPG) algorithm [10].In 2015, Lillicrap [9] proposed the DDPG algorithm. After the theory of deep Q-learning, DDPG can be considered the improved algorithm of DQN. DDP algorithm describes that the agent uses the parametric probability distribution to make the action. The Deterministic Policy Gradient algorithm aims to adjust the policy parameters in the direction of the policy gradient during the high dimension and continuous conditions. The same objective is still to train a policy that aims to achieve the max cumulative reward [11]. The Deterministic Policy Gradient algorithm tends to use low-dimensional observations like Cartesian coordinates or joint angles with high s simplicity, and DDPG needs fewer steps of experience to train than the DPG. The Trust Region Policy Optimization(TRPO) and Proximal Policy Optimization(PPO) are different kinds of algorithms using the Policy Gradient to optimize the policy.

Some other deep reinforcement Learning methods like Soft Actor-Critic (SAC) [12] may show more advantages in other domains. The Soft Actor-Critic algorithm based on the environments with continuous action spaces is some Entropy-Regularized reinforcement learning [13]. Entropy is thought of as how random a random variable is in machine learning. Generally speaking, the high entropy is linked with the high opportunity that the events happened in the train events [13]. The agent chooses the reward during the continuous times is related to the entropy within the policy. The Soft Actor-Critic algorithm is based on how to use entropy to control and limit the Q function.

2.3 Deep Reinforcement Learning for Flight Control

The business jet aircraft requires a high ability of controller when the aircraft meets unknown fault scenarios [14]. In the presence of faults, how the aircraft can address the program, remain stale, and continue the initial task is still a complex challenge for us. People assume deep reinforcement learning can be applied to the dynamic fly model. The control model will collect, train and test enough data in simulation. The Soft Actor-Critic (SAC) is one of the deep reinforcement learning methods, and we apply it in the model [13].

The classic example is about the attitude control of the UAV [15]. The ideal controller uses the PID to control the movement of the UAV. The PID controller has performed well in most circumstances. However, when some unknown dynamics, like the change in wind speed, pressure, and loads, the UAV cannot reach the ideal state. Consequently, the project aims to use reinforcement learning methods to contribute to the precision and agility of the UAV.

2.4 Reinforcement Learning for Fault Tolerant Control of a Quad-Rotor UAV

According to Newton's second law, the research uses a non-linear dynamic equation of movement to show the condition of the UAV quadrotor. Then based on the different cases of faults, the research designs several PID controllers to control the quadrotor UAV [16].

The PID controller directly adjusts the velocity and the orientation of each propeller. According to the real situation, the UAV can make a judgment and switch to the pre-tuned PID controller. In addition, fault-tolerant control (FTC), is thought of as the ability of a system to continue its operation when the system faces different kinds of fault scenarios. The author first builds the quad-rotor UAV's dynamic model and creates the PID controllers based on a fault-tolerant control strategy to address the fault scenarios[17]. Four rotors are divided into two pairs for the example of a quad-rotor UAV. The first pair is on the x-axis and rotated clockwise with the second one located on the y-axis and rotating counterclockwise. In addition, the quad-rotor UAV needs to assume one direction as its heading.

The figure 2.2) belows show the basic structure of the quad-rotor UAV.

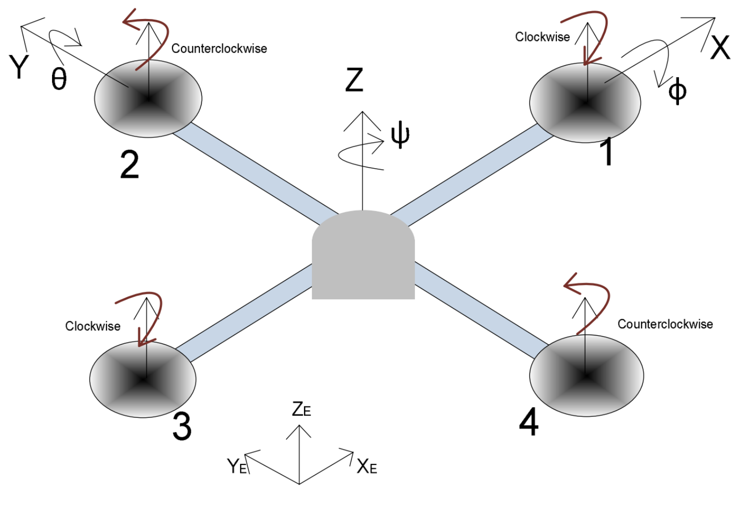


Figure 2.2: Quad-rotor UAV configuration[17].

The propeller produces a thrust and torque of the centre of the UAV, and the contrary torques will be influenced to zero. We assume the body frame of the UAV is (X, Y, Z) and the earth frame is (XE, YE, ZE). The primary action angles are roll (θ), pith (ϕ), and yaw (ψ). After a series of calculations, we can get the dynamic model of the quad-rotor UAV. The whole calculation is complex and is affected by the drag force, which is ignored in the whole progress. In fact, the environment, like the pressure and high wind speed, will cause unknown influences in the actual fight. Then we design the PID controller in terms of controlling the orientation of the propeller and the velocity of each motor [18].

The controller uses the yaw angle to control the orientation of the quad-rotor UAV when we check the heading of the UAV itself. The controller will control the movement of the x-axis and y-axis. The objective of the controller is to minimize the deviation. In addition, the speed controller will drive the quad-rotor UAV to a certain attitude and direction. For the part of the fault modelling, we assume that the rotor index is correct and creates several situations where the rotor encounters faults. When the fault scenarios happen, the main controller can identify the type of the fault and switch to the corresponding PID controller [19].

3 Quadcopter Drone Modeling

3.1 General Description

The quadrotor is an underactuated strongly coupled aircraft with six degrees of freedom and four inputs. The six degrees consist of the drone's position related to the earth X, Y, Z, and the drone's gesture, including the angles, roll (θ), pith (ϕ), and yaw (ψ). There are generally 2 types of drone results: cruciform and X-shaped. In either structure, we assume that the four rotors of the drone are symmetrically and evenly distributed at the four ends of its structural arms. When controlling the drone, the position and speed of the drone is controlled by varying the output of the four rotor blades of the drone.

The inputs of the dynamic model are 16 elements of the initial conditions including the X, Y, Z, roll angle(θ), pith angle(ϕ), yaw angles (ψ), liner velocity,U, V, W, and angle velocity P, Q, R, and the four rates of the max motor thrust. The final outputs will include 12 elements the force and torque and initial conditions in the next state.

A simple discrete-time dynamic system model is shown in the figure 3.1. .In a discrete-time dynamic system, in time t, xt in Rn is the state of a drone, and Ut in Rm means the input to the drone, the force and torque, and initial conditions in the next state. F is the drones' state transition. By the drones' state-transition and precious state and the current input, the Xt+1 in time t+1 (the next state) can be confirmed.

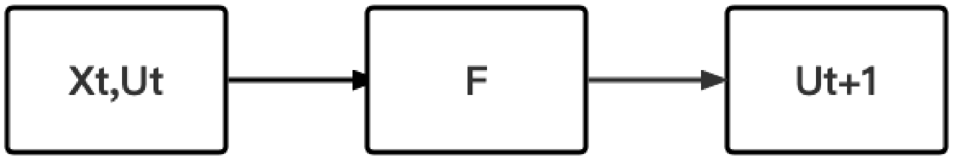


Figure 3.1: Discrete time dynamic system model.

The discrete-time dynamic system model created for this study is shown in the figure 3.2.

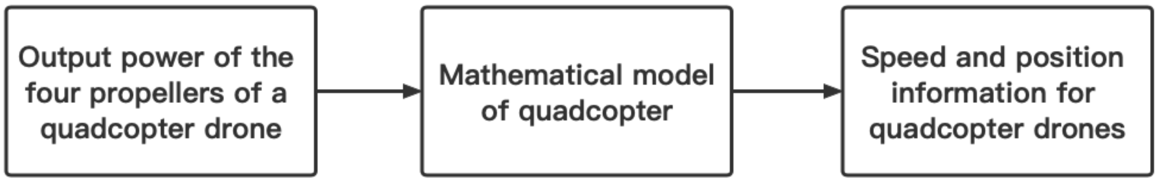

Output power of the four propellers of a quadcopter drone
Mathematical model of quadcopter
Speed and position information for quadcopter drones

Figure 3.2: Mathematical model of Quadcopter.

This model of the dynamics of a UAV is based on the model used in Alex Quessy's article [20]. This model is a close approximation of the ideal. It assumes that the UAV is symmetrical in the horizontal plane and that the rotational inertia of the UAV with respect to the z-axis is small and close to zero. Since no wind field is introduced, the forces on the UAV's blades due to the naturally occurring inhomogeneous wind in the environment are neglected. The rotational moment of the UAV's individual blades against the z-axis is also neglected, resulting in the source of the UAV's moment being the lift forces generated by the UAV's four blades.

3.2 Principle Basis for Quadcopter Modelling

3.2.1 The Newton-Euler Equation

According to the Newton-Euler equation, the rigid body motion = the translation of the center of mass + the rotation around the center of mass.The formula is shown below

$$\vec{F}_i = m\vec{a} = \frac{d\vec{v}}{dt} \tag{3.1}$$

where, F is the external force on the rigid body, a is the acceleration on the rigid body, and V is the velocity of the rigid body(see Equations 3.1).

The rotation around the center of mass is described by the Euler equation:

$$M = Jw^{'} + w \times Jw \tag{3.2}$$

M is the external torque of the rigid body; J is 3×3 inertia matrix; w is the angular velocity of the rigid body3.2.

The sum of the torques acting on the rigid body provides the body with the angular acceleration to allow it to rotate.

In this model, we define that the total force consists of the pulling force produced by propellers, the drag forces of the wind and gusts, and the drag forces of the velocity. The action of the drone can be divided into two parts, the translation process, and the spinning process.

3.2.2 Time Domain Differential Iteration

When the velocity and displacement at time t are given, it can assume a tiny time dt, and by the idea of time domain differential iteration to find the displacement information at the next time t+dt. Similarly, when the velocity and acceleration at this moment are known, it can iterate to find the velocity at the next moment. The principle of the formula is listed below:

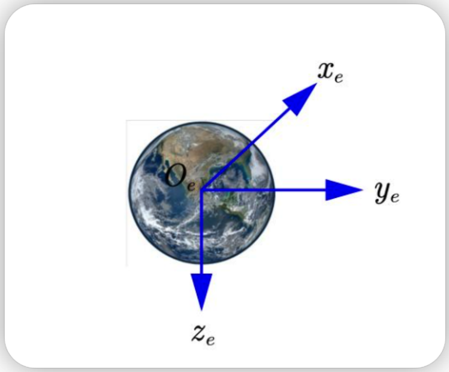
$$X_{t+dt} = X_t + V_t * dt \qquad V_{t+dt} = V_t + a_t * dt \tag{3.3}$$

where xis the displacement, v is the velocity, and a is acceleration 3.3).

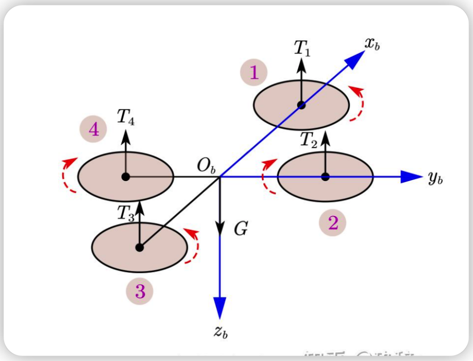
3.3 Mathematical Model of a Quadcopter

3.3.1 Coordinate System

Before establishing the model, it requires to define the coordinate system representing the vector in advance. This paper will use two coordinate systems: the inertial coordinate system (static coordinate system), which is the Earth coordinate system, and the non-inertial coordinate system (moving coordinate system), which is the body coordinate system. In the figure below, $O_e x_e y_e z_e$ in the figure3.3(a) is the earth coordinate system, $O_b x_b y_b z_b$ in the figure 3.3(b) is the body coordinate system.



(a) Earth coordinate system.



(b) Body coordinate system.

Figure 3.3: Coordinate Systems.

3.3.2 Rotation Matrix

From the different coordinate systems, the rotation matrix is linked with transforming the vector expressed in the body coordinate system into the earth coordinate system. The purpose of the transformation vector expression of rotation matrix is to express the position and velocity of the quadrotor in the earth coordinate system and the posture changes like the roll, pitch, and yaw angles and their change rates, tension and torque of the quadrotor in the body coordinate system.

$$R_b^e = \begin{bmatrix} cos\theta cos\psi & cos\psi sin\theta sin\phi - sin\psi cos\phi & cos\psi sin\theta cos\phi + sin\psi sin\psi \\ cos\theta sin\psi & sin\psi sin\theta sin\phi + cos\psi cos\phi & sin\psi sin\theta cos\phi - cos\psi sin\psi \\ -sin\theta & sin\phi cos\theta & cos\phi cos\theta \end{bmatrix} \tag{3.4}$$

where b on the lower right of R_b^e represents the body coordinate system, and e on the upper right represents the earth coordinate system. θ, ϕ, and ψ represent the roll Angle rotated around the X-axis, the pitch Angle rotated around the Y-axis and the yaw Angle rotated around the Z-axis, respectively 3.4).

3.3.3 Translation Process

According to the Newton-Euler equation:

$$F_{all} = m \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{3.5}$$

, the external force on the rigid body on the three axes can be calculated.

The real drone can be divided into 5 parts, the center part of the drone and 4 rotors. In the ideal condition, all the forces can be concentrated on the center of the mass. Then all the forces will be discussed and transformed from the body coordinate system into the earth coordinate system.

In the first place, the force of the mass(G) equals to m*g and its direction points to the center of the earth(Z_e).

$$The\ force\ of\ gravity = \begin{bmatrix} 0 \\ 0 \\ m*g \end{bmatrix} \tag{3.6}$$

Secondly, the main force is the tension that is generated by the rotors. The forces of the four motors are all in the Z-axis direction in the body coordinate system (Z_e) is F_{rotor_e}.

$$F_{rotor\_e} = \begin{bmatrix} 0 \\ 0 \\ F1 + F2 + F3 + f4 \end{bmatrix} \tag{3.7}$$

To transform the force into the earth coordinate system, the force will multiply the Rotation Matrix(R) to get the F_{rotor_b}, as the force into the earth coordinate system.

$$F_{rotor\_b} = R_b^e F_{rotor\_e} \tag{3.8}$$

Thirdly, the drag forces are divided into the wind drag force and gust drag force, and the drag of the velocity. The former is related to the environment as F_{wind}. The drag of the velocity as F_{drag}

is related to the velocity of the drone, which means the different Vx, Vy, and Vz will get the drag forces in three directions.

$$F_{wind} = \begin{bmatrix} F_{wind\_x} \\ F_{wind\_y} \\ F_{wind\_z} \end{bmatrix} \tag{3.9}$$

$$F_{drag} = \begin{bmatrix} F_{drag\_x} \\ F_{drag\_y} \\ F_{drag\_z} \end{bmatrix} = \begin{bmatrix} k_{drag\_x} V_x \\ k_{drag\_y} V_y \\ k_{drag\_z} V_z \end{bmatrix} \tag{3.10}$$

In conclusion, the resultant force equals can be calculated.

$$F_{all} = G + F_{rotor\_b} + F_{wind} + F_{drag} \tag{3.11}$$

According to Newton's second law, linear acceleration can be calculated. And by the information of the last state of velocity and position and differential iteration, the velocity and position in the next state can be calculated. So the X, Y, Z, Vx, Vy, and Vz can be confirmed.

If we consider the wind field, the velocity of the drone relative to the ground is subtracted from the wind speed of the wind potential field.

$$V_{real} = \begin{bmatrix} V_x - V_{wind\_x} \\ V_y - V_{wind\_y} \\ V_z - V_{wind\_z} \end{bmatrix} \tag{3.12}$$

3.3.4 Spinning Process

Four propellers will produce four torques. These four torques will control the rotations of the drone itself. Referring to the rotation of a rigid body and according to the Euler equation, the angular accelerations of roll, pitch, and yaw can be confirmed. And by the information of the last state of angular velocity and angular position, and differential iteration, the angular velocity, and angular position in the next state can be calculated. So the roll angle(θ), pith angle(ϕ), yaw angles (ψ), and angle velocity Wx, Wy, and Wz can be confirmed. In the case of a "ten" aircraft, it is usually considered the nose direction as the positive direction of the X axis, and the torque of the X, Y, and Z axes as τ.

In the case of "X" aircraft, it is usually considered the angle bisector of the angle between the two spirals and the centre of mass as the positive direction of the X axis. In this research, the model is designed based on the "X" aircraft.

$$\tau = \begin{bmatrix} L(F2 - F4) \\ L(F1 - F3) \\ kb(F1 - F2 + F3 - F4) \end{bmatrix} \tag{3.13}$$

where L is the distance from the center of the airframe to each propeller, and KD is a coefficient 3.13).

The inertia matrix is I.

$$I = \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \tag{3.14}$$

where J means teh inertia for 3 axis 3.14).

Because of the symmetry, it is assumed that J_{xx} is approximately equal to J_{yy}. The α can be calculated.

$$\tau = I * \alpha \tag{3.15}$$

However, in this model, the drag force formed by the individual propellers of the UAV due to their rotation is ignored as it is ideal. In the real state, a single spiral pulp has a number of spiral blades, each of which is subjected to a drag force in the opposite direction to the movement of the spiral blades. Due to the symmetry of the spiral blades, the drag moments on the X and Y axes are cancelled out, but the moments on the Z axis are superimposed. If the actual situation is considered, this part of the moment needs to be taken into account. Nevertheless, this moment is relatively small compared to the moment generated by the lift of the spiral pulp.

4 Quadcopter Model Test

4.1 General Description

The model will be tested as the 4 normal inputs of the rotors given and the results like the position and velocity information will be observed. The 4 inputs are the output power of the propeller of the drone, the 12 outputs include information about the speed and position of the drone.

Different combinations of rotational power of the propellers will affect the action of the drone. Several common outputs will be tested, then 4 rotors with an output of 0, an output of normal values, an output of is drone remains stable at the same value, and an output of different values.

Once the model had been created, it can be tested with the output of the drone with some different power outputs from the drone propellers.

The 12 outputs are the X, Y, Z, roll angle(θ), pith angle(ϕ), yaw angles (ψ), liner velocity(u,v,w), and angle velocity (p,q,r).

Before the test, the drone will reset the initial conditions with all input and output equalling to 0, and the initial condition will be set like the size and mass of the drone and rotor, the max motor thrust, the body drag coefficient, and the simulation frequency and time. It is assumed that the initial position of the drone is (0, 0, 0), but at this point, the drone is not on the ground, the drone is just in the air, and the position of the drone in the air is (0, 0, 0). The drone can still move downwards. For example, when the drone moves to (0, 0, -10), it means that the drone has moved vertically downwards by 10m. The distance from the ground to the drone is assumed to be infinite, i.e. there are no spatial constraints and no collisions between the drone and the ground in this model.

In this case, the max thrust of the rotor is 140 N/m. The simulation frequency is 100 Hz and the simulation time is 10s, which means that the code will iterate 1000 times.

4.2 Test 1: Zero Output of the Rotors

4.2.1 Input Settings

In test 1, I set the output of the four propellers to 0. This means that the drone will initially be subjected to the force of gravity only. The drone will move downwards. As the speed of the drone increases, the drone is also subject to an increased drag.

Percentage of input of thrust of the rotor	rotor 0	rotor 1	rotor 2	rotor 3
Percentage of input of max thrust	0	0	0	0

Table 4.1: Test 1: Inputs of the rotors .

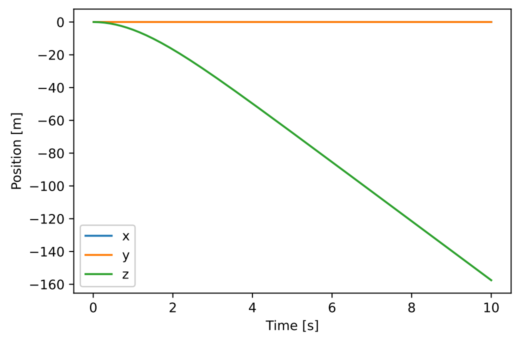
4.2.2 Data Analysis

These figures 4.1 record the position and velocity data of the drone. In test 1, the output of the four propellers is set to 0. This means that the drone will initially be subjected to the force of gravity only. The drone will move downwards. From the figures 4.1, it can be seen that the drone wants to down-accelerate at the beginning, the speed of the drone is increasing, while the acceleration of the drone is decreasing, and eventually the speed of the drone tends to stabilise and the drone does a uniform motion. When the drone receives only gravity, the drone will accelerate downwards. As the speed increases, the drag on the drone will increase and the forces on the drone will remain balanced. And the acceleration of the drone will decrease to zero and the drone will eventually move downwards at a constant speed.

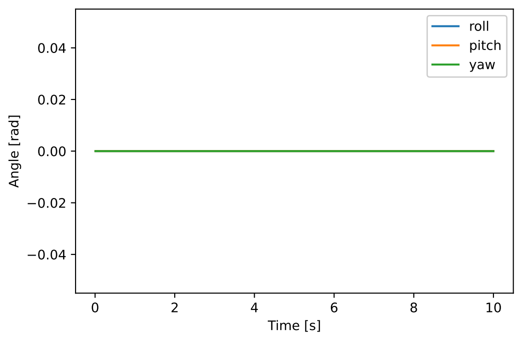
4.3 Test 2: Full Output of the Rotors

4.3.1 Input settings

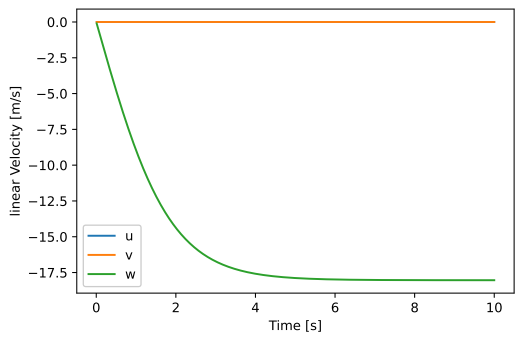
In test 2, the output of the four propellers will be 1, which means each rotor has 100% power output. The quadcopter will rise in full force.



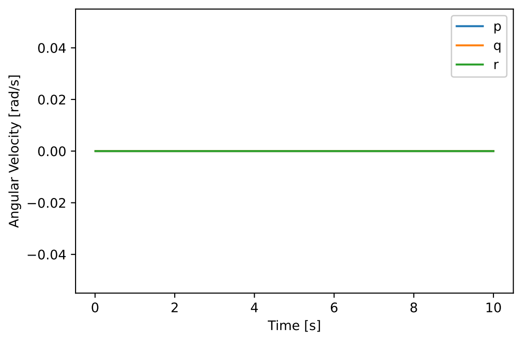
(a) Position:X, Y, Z.



(b) Angle Position: Roll, Pitch, Yaw angle:



(c) [Linear Velocity: U, V, W.



(d) Rotation Rate: P, Q, R

Figure 4.1: Position and Velocity Data of Test 1

Percentage of input of thrust of the rotor	rotor 0	rotor 1	rotor 2	rotor 3
Percentage of input of max thrust	1	1	1	1

Table 4.2: Test 2: Inputs of the rotors .

4.3.2 Data Analysis

These figures 4.2 record the position and velocity data of the drone.

From the figures 4.2, it can be seen that the drone makes an opposite motion to test1, with the drone accelerating upwards with decreasing acceleration, and finally, the drone makes an upward motion with a constant acceleration.

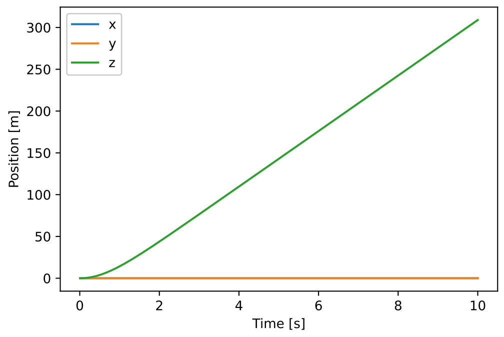
As the propellers of the quadrotor drone are all coming upwards to output lift, and as the four propellers have equal output, the drone pulp will not perform a rotational movement, so the drone will do an accelerated movement upwards. However, as the speed of the drone increases, the drag force received by the drone in the vertical direction gradually increases and eventually equilibrates with gravity, and the drone will maintain a constant speed of motion.

4.4 Test 3: Proximity Hover Test

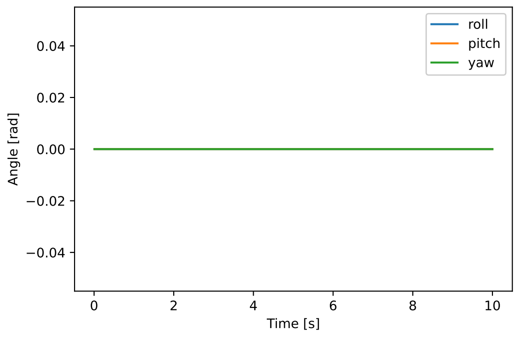
The power output of the propeller has been derived when the drone remains to hover by simple estimation. But this value is in fact an approximation and the drone will make a small movement at any time. In the next research, a reinforcement learning model to derive more accurate values for keeping the drone in balance.

Percentage of input of thrust of the rotor	rotor 0	rotor 1	rotor 2	rotor 3
Percentage of input of max thrust	0.227	0.227	0.227	0.227

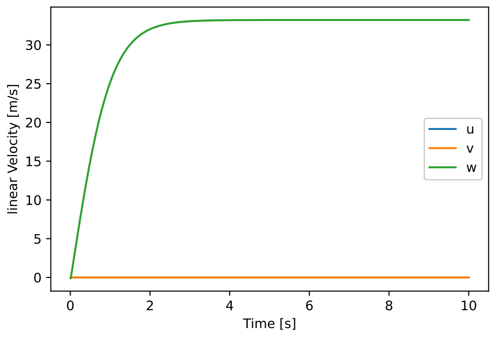
Table 4.3: Test 3: Inputs of the rotors .



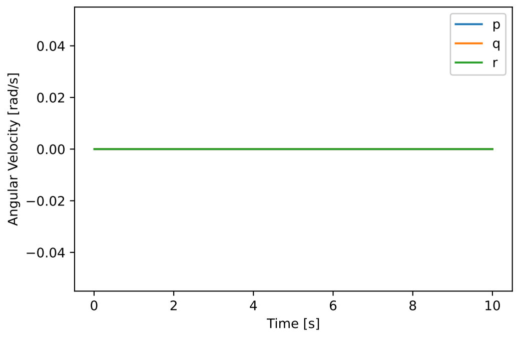
(a) Position:X, Y, Z.



(b) Angle Position: Roll, Pitch, Yaw angle:

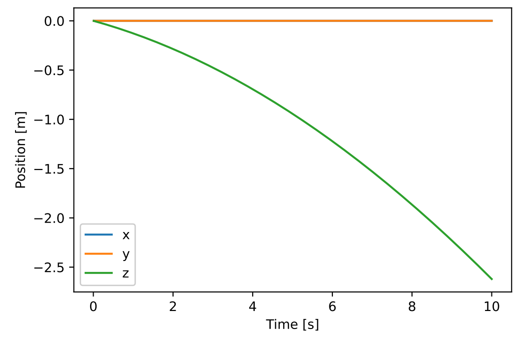


(c) [Linear Velocity: U, V, W.

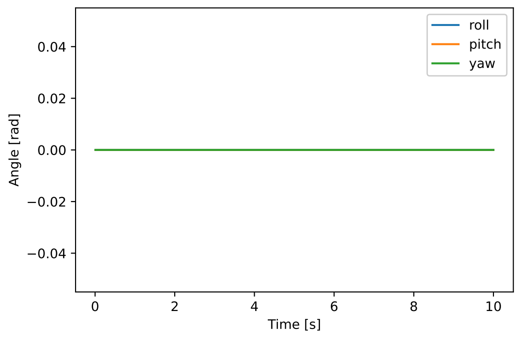


(d) Rotation Rate: P, Q, R

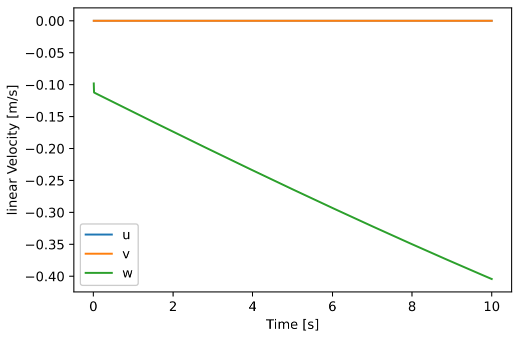
Figure 4.2: Position and Velocity Data of Test 2



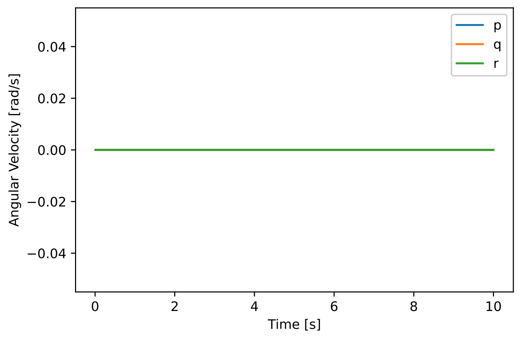
(a) Position:X, Y, Z.



(b) Angle Position: Roll, Pitch, Yaw angle:



(c) [Linear Velocity: U, V, W.



(d) Rotation Rate: P, Q, R

Figure 4.3: Position and Velocity Data of Test 3

4.4.1 Data Analysis

These figures 4.3 record the position and velocity data of the drone.

As can be seen from the figures 4.3 , the drone remains largely stable. When we give the spirals a power output of 22.7 % the lift generated by the spiral slurry is essentially equal to that of gravity. We can see from the data that the drone makes a relatively small movement relative to test2, close to hovering

4.5 Test 4: Random Action

4.5.1 Input Settings

By giving the four propellers different outputs, it can be observed with a change in both the position and attitude of the drone. However, if the speed of the drone's propeller varies greatly, the drone will make a very fast counter-rotating movement. In reality, the body of the drone will then be subjected to a great deal of stress. This huge stress will destroy the structure of the drone and cause huge and permanent damage to the structure of the drone. Therefore, based on the rotor speed that is calculated in test3 to remain the UAV stable, it makes a small adjustment to the speed of each UAV propeller.

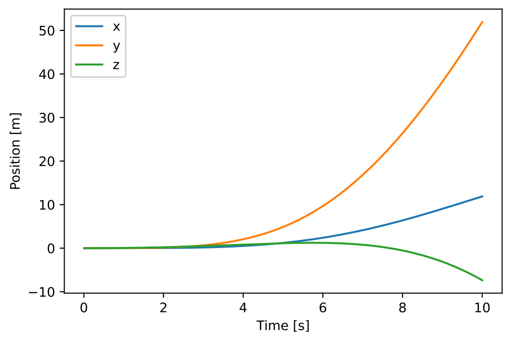
Percentage of input of thrust of the rotor	rotor 0	rotor 1	rotor 2	rotor 3
Percentage of input of max thrust	0.227+0.002	0.227+0.0044	0.227+0.003	0.227+0.005

Table 4.4: Test 4: Inputs of the rotors .

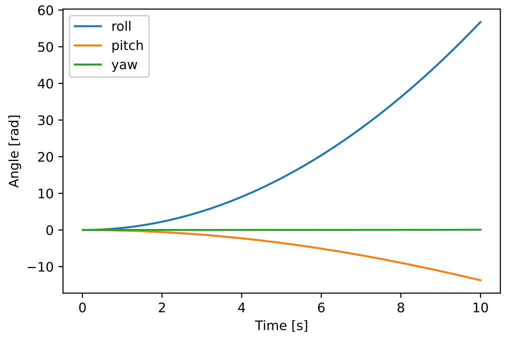
4.5.2 Data Analysis

These figures 4.4 record the position and velocity data of the drone.

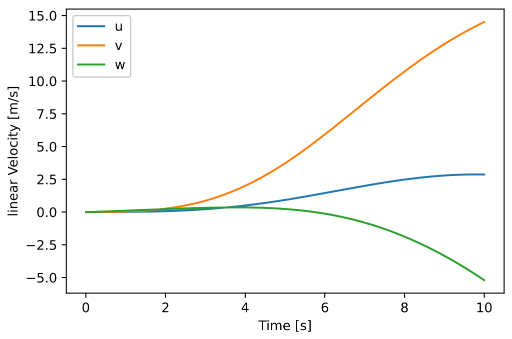
By observation of the figures 4.4, the position of the X, Y, and Z axis of the drone and the roll pitch yaw angle of the drone have changed. However, because the power output of the drone's four propellers is only very slightly different, the drone does not deflect very much and the drone moves very gently. The drone made a very graceful curving movement in the air.



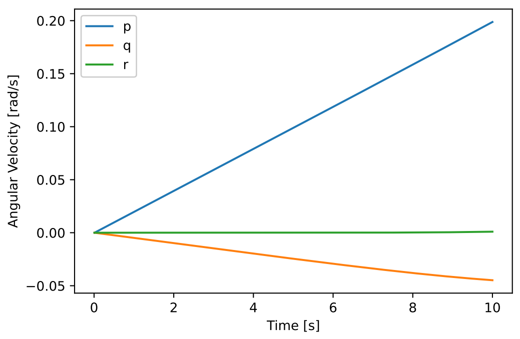
(a) Position:X, Y, Z.



(b) Angle Position: Roll, Pitch, Yaw angle:



(c) Linear Velocity: U, V, W.



(d) Rotation Rate: P, Q, R

Figure 4.4: Position and Velocity Data of Test 4

5 Reinforcement Learning

5.1 Basic Concepts of Reinforcement Learning

When a machine, a human or a robot, is in a changing environment, it has to perform certain tasks. The challenge is how it accomplishes this task. The purpose of reinforcement learning is to make it better at this task. Consequently, reinforcement learning is used to develop a strategy to help the person or the robot complete the task. In other words, the strategy determines the agent's every decision.

5.1.1 Agent-Environment Interaction

When a task is imitated in reinforcement learning, some factors, such as agent, state, action, agent, and policy, are related to the process. Let us assume a navigation problem as a classic example.

The agent means the robot or the machine. In a way, it refers to the main body of the operator. During the navigation problem, the agent refers to the car itself.

The state can be obvious as an environment state, including the agent and the environment that the agent can feel. For example, during the navigation problem, the state refers to the car's location, speed and direction, map and traffic light conditions, and surrounding vehicles.

The action means what the car wants to do, like acceleration, slowing down, lane choice, direction decisions, and path planning.

The policy is a π function. What policy means is that when the state is observed, what action should the car take? A policy means to make decisions based on the observed state to control agent movement. The policy function π is mathematically defined as a probability density function as the probability density of making an action A in a given state S.

The reward R influences the result. When an agent makes an action, the game will give a reward, which generally needs to be defined by us. The definition of reward is straightforward to affect the result of reinforcement learning. For example, during the navigation problem, the reward can be the time, the length of the path, or the accuracy of reaching the target point.

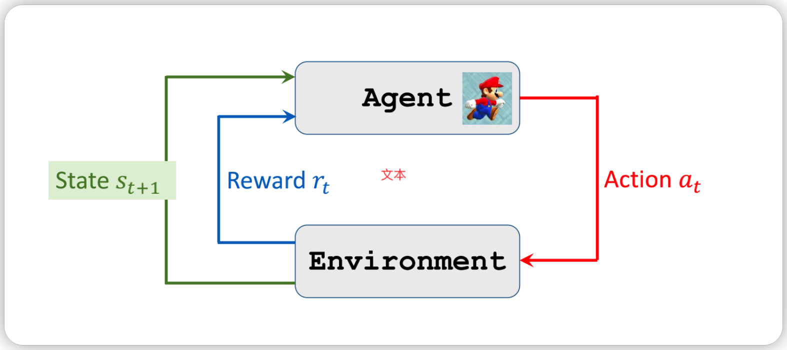


Figure 5.1: Agent-environment interaction loop [21].

In the current state, the agent makes action, and the environment gives a new state and returns a reward. This process is called state transition in the figure 5.1.

5.1.2 Randomness of Reinforcement Learning

It is generally accepted that there are two sources of randomness in reinforcement learning.

The first randomness is based on the actions because the action function is randomly sampled according to the policy function π. The policy π function is used to control the agent. Given the current state S, the agent's action A is randomly sampled according to the probability output by the Policy function π, such as the currently observed state S. The policy function will describe to us what the probability of each action is.

Another randomness is state transition. This randomness is based on the uncertainty of the environment. Assuming that the agent has made an action, the environment will generate the next state called S'. The S' comes from the action and the environment, like the car speed and change of location, and the position and speed of surrounding cars.

5.1.3 Return of the Reinforcement Learning

The return is translated as the cumulative future reward. We call the return at t time Ut. The definition of return is to add up all the rewards at T time until the last reward at the end of the game.

Discounted return adds the time factor into the return. The return is also linked with time.

Discounted return Ut is just a random variable, and at time T, you don't know what Ut is, so let's say you flip a coin, heads are 1, tails are 0, and at time T, you haven't flipped the coin yet, you don't know whether you got a one or a 0.

Ut is a random variable that depends on all the actions and states. Since Ut is a random variable, I don't know what Ut is at time T. The current situation can't be evaluated now.

Ut can be expected to drop inside the randomness of the product, which is a real number. For example, although you don't know what you're gonna get before tossing a coin, the probability of each is clear to half if reverse by 1 to 0 to get positive expectations is 0.5, also obtained the expectations will be a number of random variables Ut o as Q π. This expectation is how of, the expectation of Ut all future state S and all of the action of A function, A future state of action A and S has randomness, the action of A probability density function is the policy function of PI (A | S), state S probability density function is state transfer function p (S | 'S, A), Expectation is the future action, the future state, you take these random variables, and you integrate them, and you multiply all the random variables except for St and At, and the random variables that you multiply are A(t+1), A(t+2)... And S (t+1), S (t+2)... Such as state. Qπ is called an action-value function. Qπ is related to the current state and Action (St and At). Because the rest of the actions and values are multiplied, but St and At are not, St and At are treated as observed values, not as random variables.

The value of Qπ depends on St and At, and the function Qπ depends on the policy function π . The policy function is used in the integration, and if the policy function is different, the integral will be different Qπ .

Qπ tells us whether it is good or bad to do at time T in the state St if we use policy π . Given policy π , Q π will rate all the actions in the current state, and then we know which actions are good and which are bad.

Through continuous training, the agent performs the best actions and avoids the bad ones in the expectation of getting the final reward for this action in this environment.

5.2 Reinforcement Learning of Quadcopter

Reinforcement learning is used to create models to control the UAV. The basic UAV movement patterns such as hovering, i.e. stabilization, ascent and descent in the air, and the advanced solution of faults scenarios. Modifying the environment to model flight failures, there are many scopes of experimentation, like the different types of the fault of the rotor and their corresponding processing methods. The model of reinforcement learning will contribute to aircraft remaining the drone stable in a new environment.

Our goal is to build a reinforcement learning model to control drones. The reinforcement learning model will be trained for different action spaces and reward for a different requirements like the example in the figure 5.2.

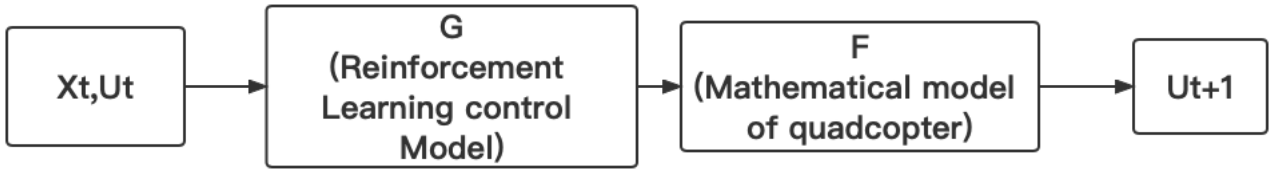


Figure 5.2: Reinforcement Learning Model.

To create a model to control a drone through reinforcement learning, some basic factors can be set as follows.

- The agent: The quadcopter.
- The state: The 12 outputs of the model (the drone's position related to the earth X,Y,Z, and the drones gesture, including of the angles, roll (θ), pith (ϕ), and yaw (ψ),liner velocity,Vx,Vy,Vz, and angle velocity Wx,Wy,Wz,)
- The action: The four rate of the max motor thrust (Percentage of output power).

- The environment: The different fault scenarios of rotor:1, rotor 0 has 0 output. 2, rotor 0 has the output that change with time. 3, rotor 0 has the random output.
- The return: the return is linked to the reward. The quadcopter will run in 10s and the return will be the sum of the rewards.
- The reward: The reward will be designed according to the requirement of the different type of task.

5.3 Proximal Policy Optimization (PPO) Algorithm

5.3.1 Model-Free Algorithm

The proximal Policy Optimization (PPO) Algorithm s is a classical reinforcement learning algorithm. It is a type of model-free algorithm, which means that the agent does not have access to the environment model. The agent cannot understand the model of the environment and can only create the model through experience gained through continuous experimentation.

What this situation leads to is that when we create the model in the project, the bias in the model will have a negative effect. The model will have a high reward and performance during the training process, but it will show a disappointing result when this model with high reward during training is applied in the real environment.

5.3.2 Policy Optimization

Policy Optimization will directly optimize our policy to maximize returns. The goal aims to maximize the expected return $J(\pi_\theta)$ when the stochastic parameterization of the policy π_θ in the table 5.1.

$$J(\pi_\theta) = \tau \sim \pi_\theta R(\tau) \tag{5.1}$$

Gradient ascent is applied to optimize the strategy like the gradient of policy performance:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_\theta)|_{\theta_k} \tag{5.2}$$

where $\nabla_\theta J(\pi_\theta)$ is the policy gradient in the table 5.2.

PPO as the most welcome Reinforcement Learning method aims to take the biggest possible improvement step to improve the gradient ascent by the current data, without stepping so far to cause faults performance [22].By using a range of first-order methods to adjust the gradient.

5.3.3 Proximal Policy Optimization Theory

Based on the Policy gradient methods by computing an estimator of the policy gradient and plugging it into a stochastic gradient ascent algorithm. The PPO aims to use the Trust Region Method by adjusting the penalty coefficient.

PPO applies an adaptive Kullback–Leibler divergence penalty to control the change of the policy at each iteration to create an objective function [22]. PPO-Clip relies on specialized clipping in the objective function to remove incentives for the new policy to get far from the old policy.

$$L^{CLIP}(\theta) = \hat{E}[min(r_t\hat{A}_t, clip(rt(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)] \tag{5.3}$$

where θ is the policy parameter, $\hat{E}_t$ denotes the empirical expectation over timesteps, r_t is the ratio of the probability under the new and old policies, respectively, $\hat{A}_t$ is the estimated advantage at time t and ε is a hyperparameter, usually 0.1 or 0.2 [22].

In supervised learning, it is easy to implement a cost function, perform gradient descent on it, and be very confident of obtaining excellent results with relatively little hyperparameter tuning. However, the path to success with reinforcement learning is not as obvious - the algorithm has many difficult to debug moving parts and requires a lot of tuning work to get good results. PPO balances ease of implementation, sample complexity, and ease of tuning by trying to compute updates at each step that minimise the cost function while ensuring relatively small deviations from the previous strategy [23].

The PPO applies an adaptive Kullback-Leibler divergence penalty to control the policy change at each iteration to create an objective function [22].

This algorithm does a Trust Region update and simplifies the algorithm by removing the KL penalty and the need to make adaptive updates. The PPO is most used in continuous action spaces.

6 Basic Control Model of Reinforcement Learning

6.1 General Description of Basic Action of Quadcopter

The drone has some basic movement patterns, including hovering, ascending and descending, moving left, right, back, and forth. Here this research tries to build controllers with reinforcement learning algorithms to allow the drone to hover, rise to a certain height and remain stable, and move a certain distance to the right and remain relatively stable.

6.2 Basic action: Stabilisation (Remain stable at (0, 0, 0))

6.2.1 Model Building

When a drone is required to be stable, it is assumed that the four rotors of the drone produce the same speed to counteract each other's torque and that the lift of the drone is in balance with the force of gravity. At this point, the output power of the four rotors of the UAV is action space in the table 6.1. This is modelled by a reinforcement learning algorithm (PPO), and in the computational model, the exact output power of the rotors is obtained. In section 4.3, the power has been initially calculated in order to require the UAV to keep the drone in balance. When the torque output from the drone's four propellers is 22.7% of the maximum torque, the five-man aircraft remains largely stable. However, there is still a small upward movement of the drone at this point. The reinforcement learning model will give a more accurate value, and the drone will remain more stable.

Percentage of input of thrust of the rotor	rotor 0	rotor 1	rotor 2	rotor 3
Percentage of input of max thrust	action[0]	action[0]	action[0]	action[0]

Table 6.1: Stabilisation Model: Inputs of the rotors .

6.2.2 Model Test

In the computing model, the change of position and velocity data of the UAV in 10s are obtained in the figures 6.1). And the precise output power of the rotor is obtained from the action at stabilization.

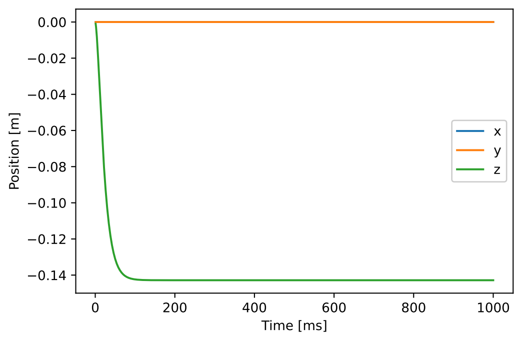
6.2.3 Analysis

Observing the data shows that the drone remains largely stable at 10s. At first, there is a certain acceleration of the drone due to gravity, but afterwards, the drone soon remains stable through reinforcement learning of the model. At this point, the action equals 0.2277321, which is close to our previous initial calculation. Therefore, when the drone's output is this value, it will hold its stability well and hover in its original position.

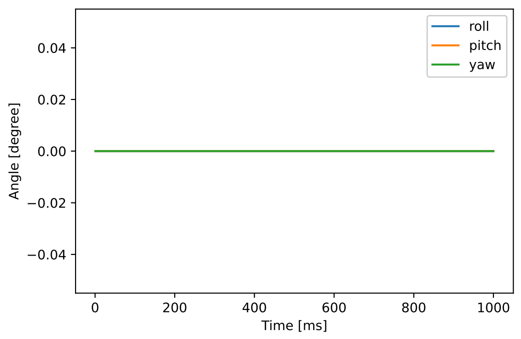
6.3 Basic Action: Ascent (Go up to (0, 10, 0) and remain stable)

6.3.1 Model Building

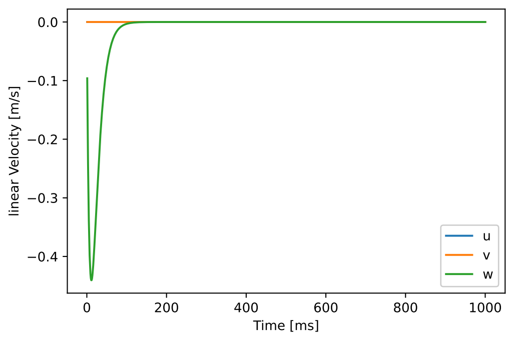
It is required that the four rotors of the drone produce the same speed to counteract each other's torque and that the lift of the drone is in balance with the force of gravity when the UAV reaches the target position. The output power of the four rotors of the UAV is action space in the table 6.2. A reinforcement learning algorithm models this, and in the computational model, the exact output power of the rotors is obtained.



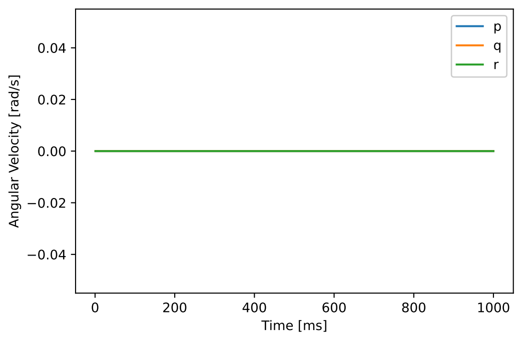
(a) Position:X, Y, Z.



(b) Angle Position: Roll, Pitch, Yaw angle.



(c) [Linear Velocity: U, V, W.



(d) Rotation Rate: P, Q, R.

Figure 6.1: Position and Velocity Data of Stabilisation Model.

Percentage of input of thrust of the rotor	rotor 0	rotor 1	rotor 2	rotor 3
Percentage of input of max thrust	action[0]	action[0]	action[0]	action[0]

Table 6.2: Ascent Model: Inputs of the rotors .

6.3.2 Model Test

In the computing model, the position and velocity data of the UAV in 10s are obtained in figures 6.2). And the stable output power of the rotor is obtained when the drone is at stabilisztion.

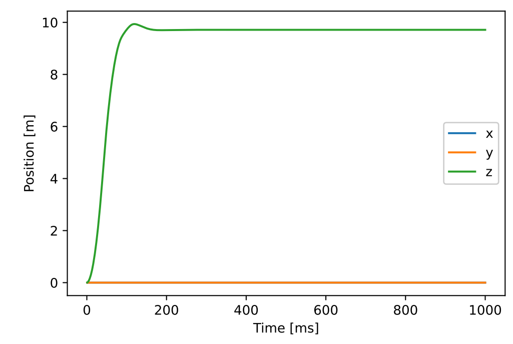
6.3.3 Analysis

The figures 6.2 show that the UAV starts with an accelerated ascent at a very high speed and then decelerates as the UAV approaches a height of 10m, and the speed of the UAV gradually becomes zero and the UAV gradually maintains equilibrium close to 10m. The drone rises to a height of nearly 10m with the reinforcement learning model and has good stability at this height. Finally, the UAV remains stable in (0,0, 9.71369). In fact, the drone was off by 0.28631 at the target, which is within acceptable limits.

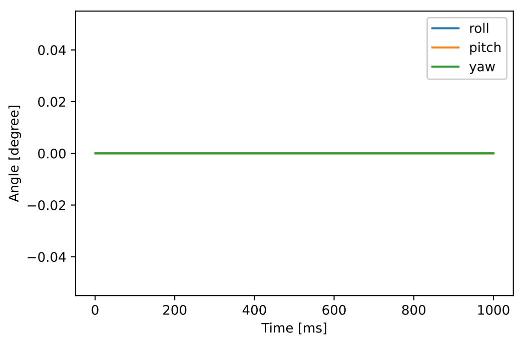
6.4 Basic Action: Horizontal Movement (Move to (0, 10, 0) and remain stable

6.4.1 Model Building

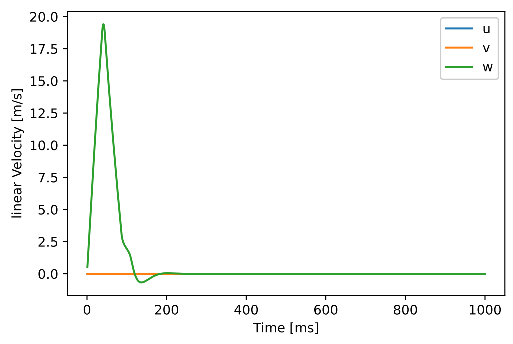
In order to control the horizontal movement of the drone, it is required the two adjacent drone propellers of the quadcopter to maintain the same speed and make them different from their counterpart propellers. For example, if the research wants the drone to move to the right, for an X-shaped drone, we need to keep rotor0 and rotor1 at a lower speed and rotor2 and rotor3 at a higher speed. Consequently, the drone will move to the right. At this point, the yaw and pitch angles of the drone will not change due to the symmetry of the drone.



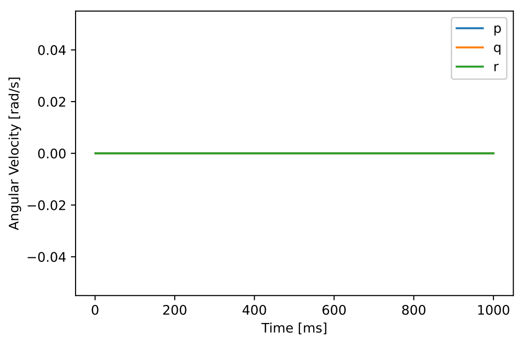
(a) Position:X, Y, Z.



(b) Angle Position: Roll, Pitch, Yaw angle.



(c) [Linear Velocity: U, V, W.



(d) Rotation Rate: P, Q, R.

Figure 6.2: Position and Velocity Data of Ascent Model.

Percentage of input of thrust of the rotor	rotor 0	rotor 1	rotor 2	rotor 3
Percentage of input of max thrust	action[0]	action[0]	action[1]	action[1]

Table 6.3: Horizontal Movement Model: Inputs of the rotors .

6.4.2 Model Test

In the computing model, the position and velocity data of the UAV in 10s are obtained in figure 6.3). And the precise output power of the rotor can be observed from the action at stabilization.

6.4.3 Analysis

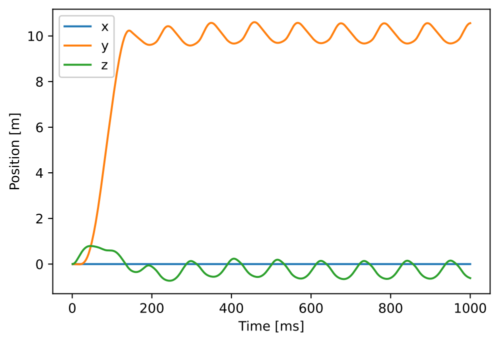
The figures 6.1 show that the drone starts with a very high-speed acceleration to the left and that the drone also has a very high deflection speed and roll angle. The drone then reaches a point near (0, 10, 0), where the drone makes a regular deflection.

Although the drone reached its intended position and drone continued to oscillate out of the target. However, we found by the change in the attitude angle of the drone that the drone performed a flip. This is because the model of the drone allows the drone's paddles to provide downward force, which means that the drone's paddles are allowed to provide downward and upward velocity. So in the theoretical model, this is acceptable.

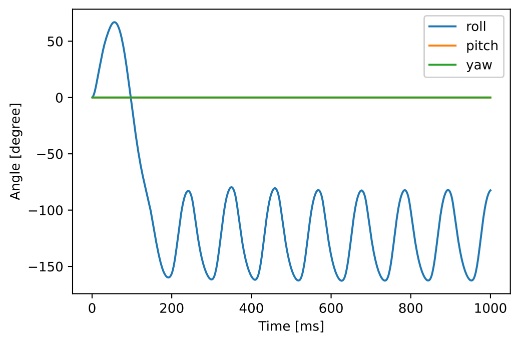
6.5 Results

Three models were built by reinforcement learning to control the hovering, upward and horizontal movements of the UAV. The first 2 of these models maintained good superiority and achieved satisfactory results. The third model, however, basically meets our requirements but still has some shortcomings. As the horizontal movement of the UAV involves coordinated changes in the position and angle of the UAV, the resulting model with reinforcement learning is also more complex. In contrast to conventional PID control, which is designed for each stage, reinforcement learning is actually performed on the whole idea of stages.

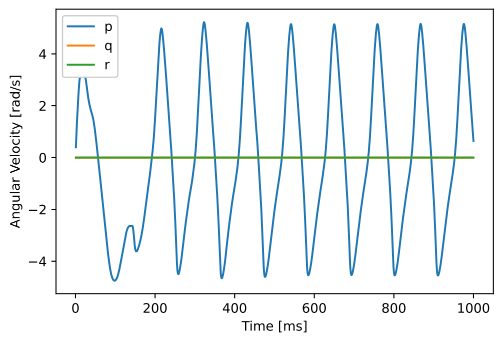
It may be possible to improve the performance of the resulting model by optimising the reward

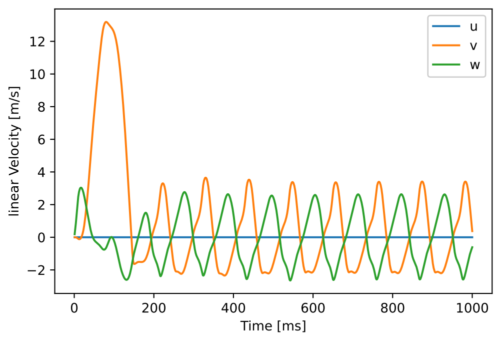


(a) Position:X, Y, Z.



(b) Angle Position: Roll, Pitch, Yaw angle.





Horizontal Movement

(c) [Linear Velocity: U, V, W.

(d) Rotation Rate: P, Q, R.

Figure 6.3: Position and Velocity Data of Horizontal Movement Model.

function, planning the task cut-off conditions, improving the number of times the model is trained and designing the whole task in different simpler stages, and using reinforcement learning to design the model for each stage.

The Gaussian function is used as the computational function for the reward function. The curve of the Gaussian function is continuous and smooth. And it does not have very drastic abrupt changes. However, in the model under study, the drones do not actually fit perfectly under certain conditions. For example, the Gaussian function is symmetrical, and when section2 assumes that the drone reaches (0, 0, 10), the drone's reward calculated by the Gaussian function will be maximised. In practice, however, the drone does not move from (0, 0, 0) to (0, 0, 10) in a symmetric manner. If a higher degree of accuracy is required, the reward function can be adapted or designed to suit the actual situation.

7 Control Model of Faults Scenarios Built by Reinforcement Learning

7.1 Faults Scenarios

Drones inevitably encounter a number of faults scenarios. In particular, the propeller blades of drones are very fragile. When a drone collides with something in the air, such as a bird, a leaf, or a piece of rubbish in the air, the drone's propeller blades are the first to receive damage. The first to receive damage is the drone's propeller blades. In this case, the output of the drone's propeller blades will be affected. Here we assume three scenarios: the output of one of the drone's propeller blades is 0. The output of one of the drone's propeller blades varies regularly with time. The output of a drone's propeller blade is an unpredictable random number.

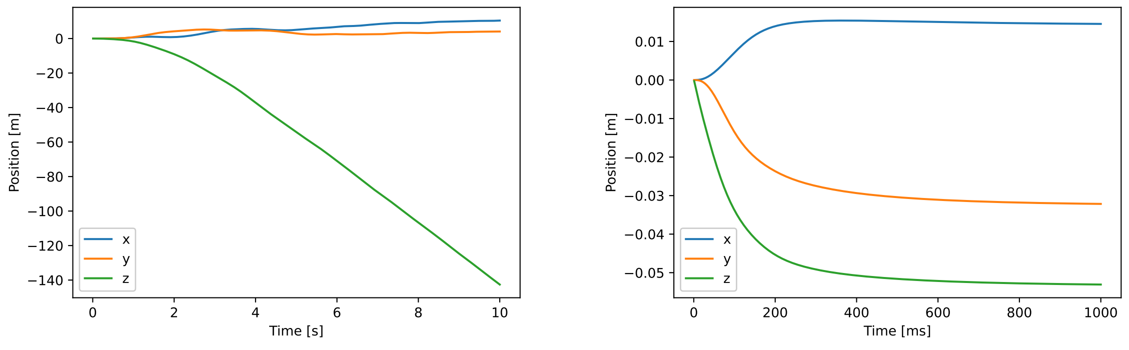
7.2 Faults Scenarios 1: rotor 0 have 0 input

7.2.1 Model Built

In this model, it is assumed that the output of one propeller (rotor 0) of the drone is constant at 0. The output of the other three propellers is first imported into the action space in the table 7.1. The design of the reward is based on the stability of the position of the drone. The Gaussian function is applied to the reward function, which is the density function of the normal distribution. If the drone is closer to our initial stable position, the higher the reward function feeds back through the Gaussian function, and the higher the return accumulated through time. With continuous training through PPO, the model created by reinforcement learning will continue to improve the return generated by its model. The model created by reinforcement learning will then continue to be optimized within a large number of training sessions.

Percentage of input of thrust of the rotor	rotor 0	rotor 1	rotor 2	rotor 3
Percentage of input of max thrust	0	action[0]	action[2]	action[3]

Table 7.1: Faults Scenarios 1: Inputs of the Rotors .



(a) Faults Scenarios 1 without Applying the Model. (b) Scenarios 1 after Applying the Model.

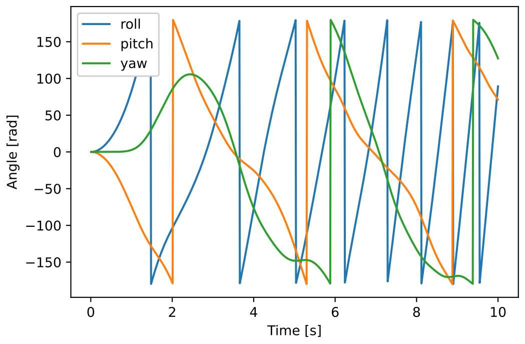
Figure 7.1: Position:X, Y, Z of Faults Scenarios 1.

7.2.2 Model Test

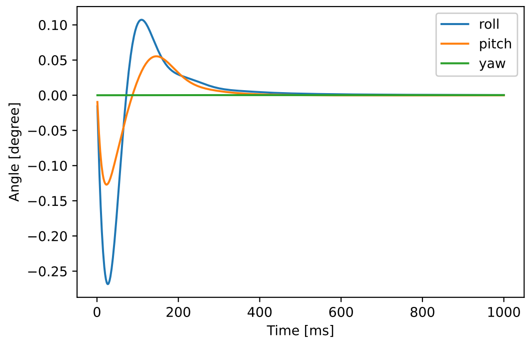
Firstly, in our previous tests (in section 6.1), the output power of the drone is gained for drone stabilization. It is assumed that all four propellers of the drone were outputting at this power, at which point one propeller blade of the drone suddenly became zero, and the change in position and speed of the drone is observed. Then, once the model generated by reinforcement learning was applied to the drone, then the research continued to observe the change in position and speed of the drone and compared it to the data that was not previously used with the model to see if the model could keep the drone in a relatively stable state. To make the performance of the model more apparent, the run time has been increased to 100s (10000ms).

7.2.3 Analysis

As you can see from the figures 8.4(b), 7.2(b), 7.3(b), and 7.4(b) on the left. However, when reinforcement learning is applied to generate a model for the drone, it is found that the drone view had a small perturbation at the beginning but remained in a relatively stable state for the rest of the

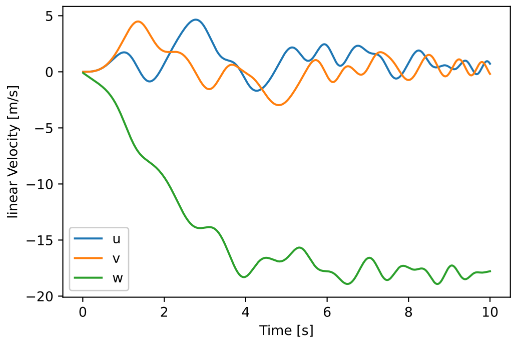


(a) Faults Scenarios 1 without Applying the Model.

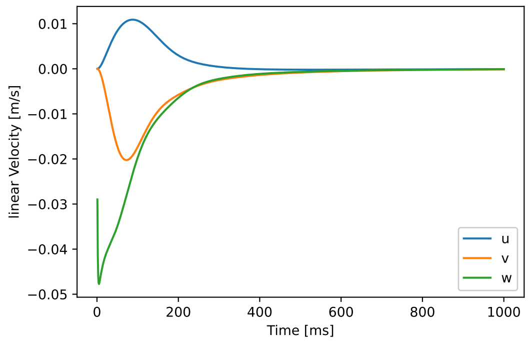


(b) Scenarios 1 after Applying the Model.

Figure 7.2: Angle Position: Roll, Pitch, Yaw angle of Faults Scenarios 1.

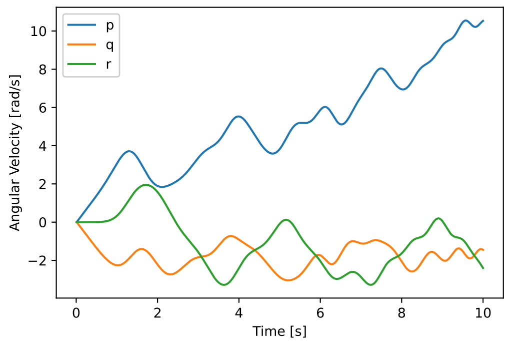


(a) Faults Scenarios 1 without Applying the Model.

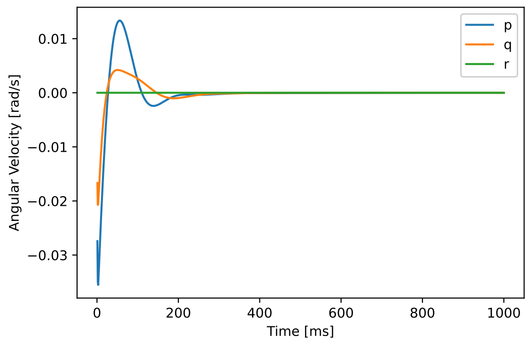


(b) Scenarios 1 after Applying the Model.

Figure 7.3: Linear Velocity: U, V, W of Faults Scenarios 1.



(a) Faults Scenarios 1 without Applying the Model.



(b) Scenarios 1 after Applying the Model.

Figure 7.4: Rotation Rate: P, Q, R of Faults Scenarios 1.

time. This perturbation is extremely small. And within acceptable limits. By comparing the model with and without the model applied, it is found that the model obtained by reinforcement learning kept the UAV stable in situ within this Faults Scenarios 1 and that the UAV became more and more stable as the run time increased.

7.3 Faults Scenarios 2: rotor 0 have the input changed with time

7.3.1 Model Built

In the previous model, it is assumed that one of the propellers of the UAV was damaged as a whole, and its output was constant at 0. However, in reality, the output of the UAV's propellers may be uncontrollable due to unknowable problems such as heat or chip aging, and it is assumed that due to some unknowable failure, one of the UAV's propellers will have a regular output that varies over time. The outputs of the propellers are designed in the table 7.3. In the same way, a Gaussian function is used to design the reward, i.e. the closer the drone is to our initial stable position, the higher the reward function fed through the Gaussian function. In the end, the optimized model created with reinforcement learning is achieved.

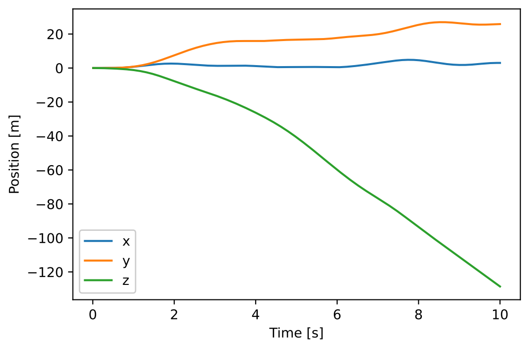
Percentage of input of thrust of the rotor	rotor 0	rotor 1	rotor 2	rotor 3
Percentage of input of max thrust	a* sin(current time)	action[0]	action[2]	action[3]

Table 7.2: Faults Scenarios 2: Inputs of the Rotors .

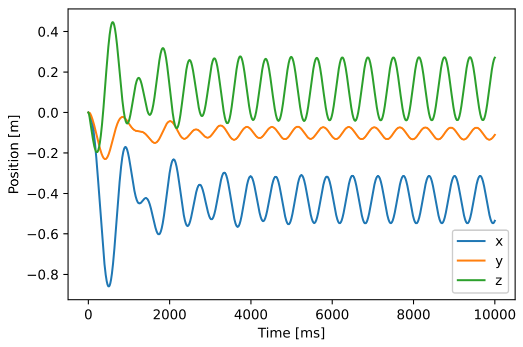
7.3.2 Model Test

Similarly, when the drone is in hover, at which point one propeller blade of the drone suddenly changes regularly over time while the other propeller blades maintain their previous output, the change in position and velocity of the drone can be observed as a comparison.

At the same time, the model generated by reinforcement learning is applied to the drone and then the research continue to observe the change in position and speed of the drone, by comparing this

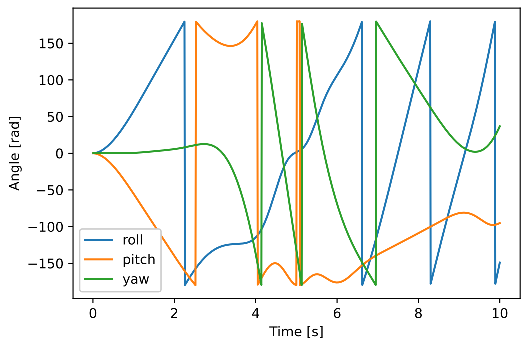


(a) Faults Scenarios 1 without Applying the Model.

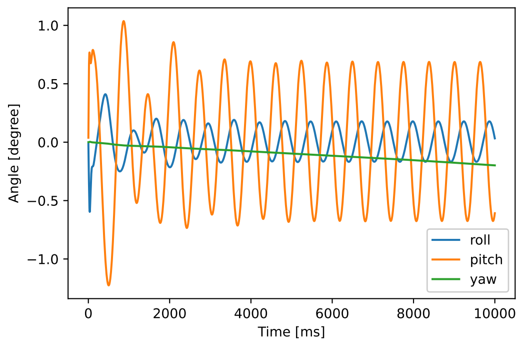


(b) Scenarios 1 after Applying the Model.

Figure 7.5: Position:X, Y, Z of Faults Scenarios 2.



(a) Faults Scenarios 1 without Applying the Model.



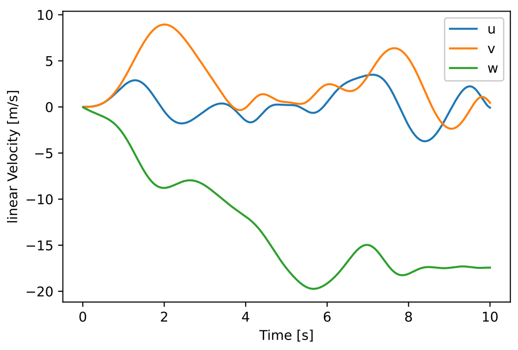
(b) Scenarios 1 after Applying the Model.

Figure 7.6: Angle Position: Roll, Pitch, Yaw angle of Faults Scenarios 2.

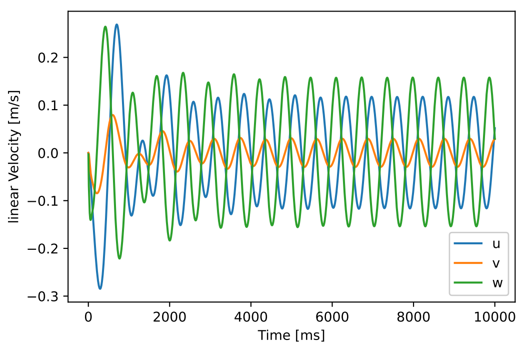
it can test whether the drone is in a relatively stable state.

7.3.3 Analysis

As can be seen, by the figures 7.5(a), 7.6(a) ,7.7(a), and 7.8(a) on the right, when the drone is in hover, one propeller blade of the drone suddenly changes regularly over time while the other propeller blades maintain their previous output. At this point, the position and speed of the drone will change very badly and the drone's position and speed will deviate from stability. As a result, the movement of the drone becomes irregular.

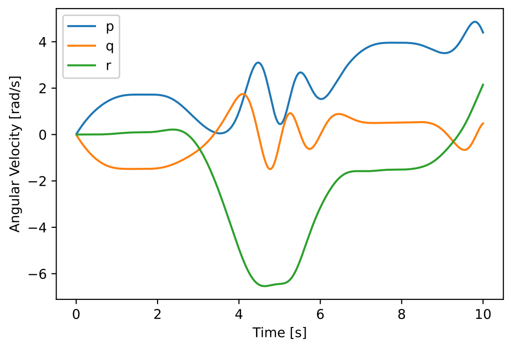


(a) Faults Scenarios 1 without Applying the Model.

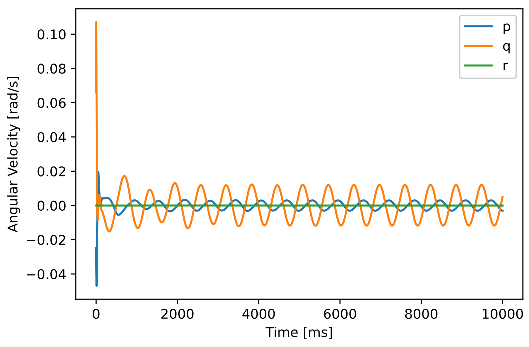


(b) Scenarios 1 after Applying the Model.

Figure 7.7: Linear Velocity: U, V, W of Faults Scenarios 2.



(a) Faults Scenarios 1 without Applying the Model.



(b) Scenarios 1 after Applying the Model.

Figure 7.8: Rotation Rate: P, Q, R of Faults Scenarios 2.

On the contrary, the figures 7.5(b), 7.6(b), 7.7(b), and 7.8(b) on the left is a stark contrast. When the trained model is applied to the drone, it makes a regular oscillating motion in its initial position and remains stable to a certain extent. In the beginning, the drone is subjected to a slight shock, but the position of the shock is within 1m and the angle of the shock is within 2 degrees. The UAV is then under the control of the model. Although one of the helices is still changing over time at this point, the other 3 rotors keep the UAV in a good steady state with the model controlling it within an extremely small range of oscillations. Compared to the case without the model applied, the drone performed very well with the reinforcement learning model loaded.

7.4 Faults Scenarios 3: rotor 0 have the random input

7.4.1 Model Built

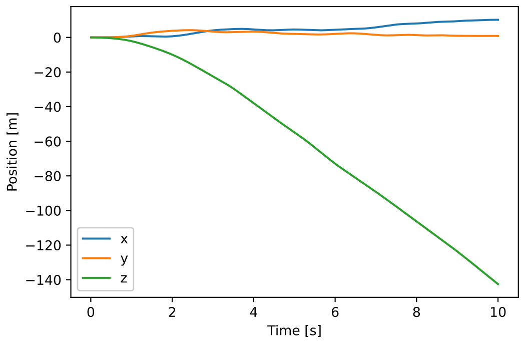
The failure scenarios for drones are varied, if the drone's propeller blades are physically damaged or if the chip does not control the motor, i.e. the motor's output is not determinable. The output of the propeller blade of the drone will not be determinable. It is assumed that the output of the UAV's propeller blade is a random quantity at this point, and reinforcement learning aims to be used to generate a model that allows the UAV to remain relatively stable in this situation.

Percentage of input of thrust of the rotor	rotor 0	rotor 1	rotor 2	rotor 3
Percentage of input of max thrust	Random number	action[0]	action[2]	action[3]

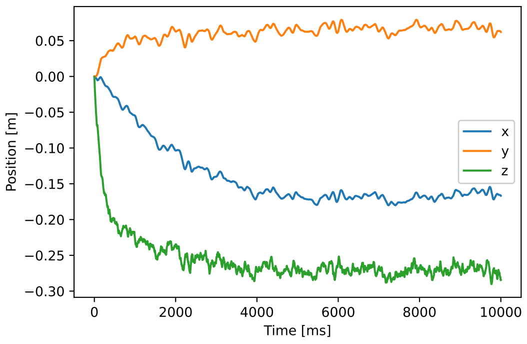
Table 7.3: Faults Scenarios 2: Inputs of the Rotors .

7.4.2 Model Test

Since the output of one propeller in the UAV is a random quantity, i.e. the output of the propeller will vary randomly over a small amount of time dt. At this point, the position and speed of the UAV change as shown. Once the model of the UAV has been applied, the changes can be observed in the corresponding parameters of the UAV. Similarly, for a clearer view of the changes in drone position and speed, the test period is increased to 100s (10000ms).

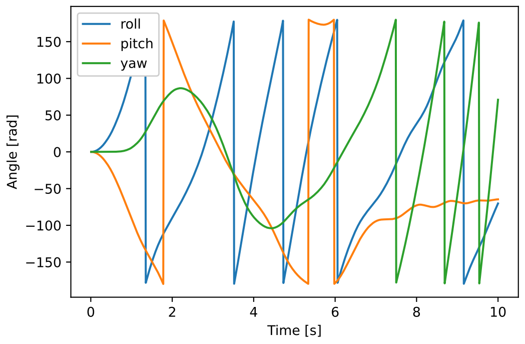


(a) Faults Scenarios 1 without Applying the Model.

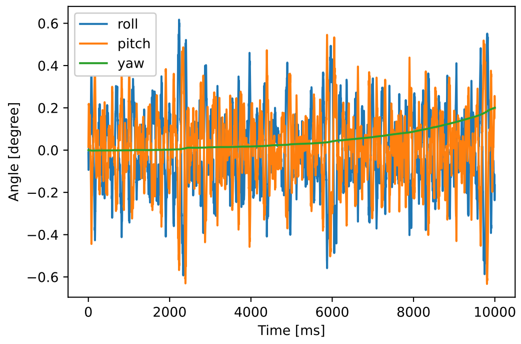


(b) Scenarios 1 after Applying the Model.

Figure 7.9: Position:X, Y, Z of Faults Scenarios 3.

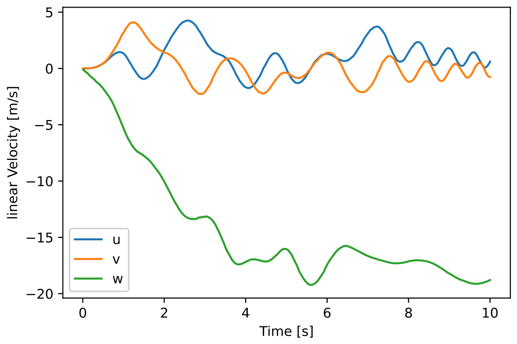


(a) Faults Scenarios 1 without Applying the Model.

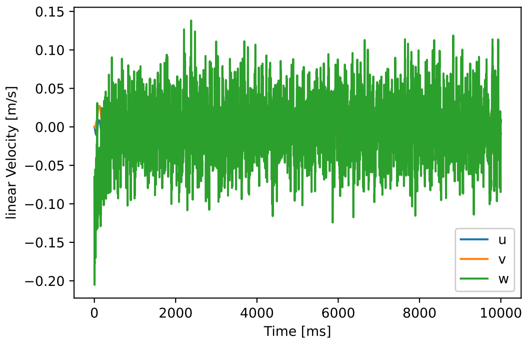


(b) Scenarios 1 after Applying the Model.

Figure 7.10: Angle Position: Roll, Pitch, Yaw angle of Faults Scenarios 3.

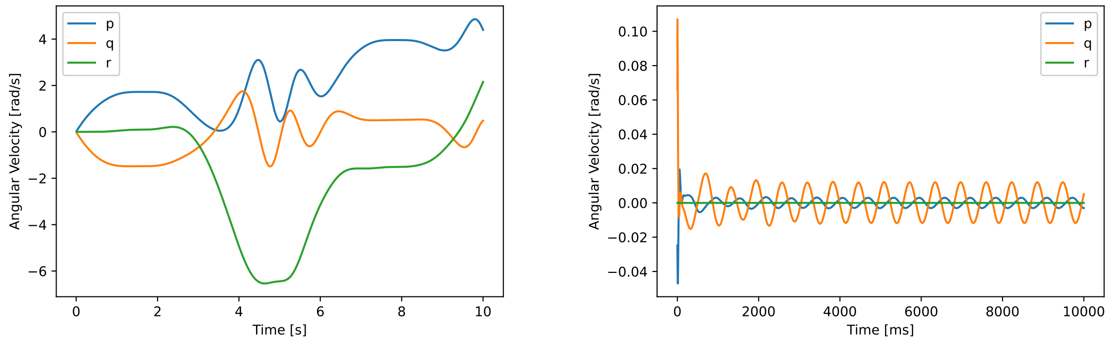


(a) Faults Scenarios 1 without Applying the Model.



(b) Scenarios 1 after Applying the Model.

Figure 7.11: Linear Velocity: U, V, W of Faults Scenarios 3.



(a) Faults Scenarios 1 without Applying the Model.   (b) Scenarios 1 after Applying the Model.

Figure 7.12: Rotation Rate: P, Q, R of Faults Scenarios 3.

7.4.3 Analysis

As can be seen from the figures 7.9(a), 7.10(a) ,7.11(a), and 7.12(a) on the right, the vertical position of the drone has changed considerably and the drone has then reversed itself several times. The drone is in a very unstable state. The output of one of the drone's propeller blades will not be determined, the drone will not be able to maintain balance, and the drone will continue to flip and move away from the target point.

After we apply the model generated by reinforcement learning to control the output of the other three propeller blades, the drone has a small perturbation at the beginning. However, it stays relatively stable for the rest of the time. As can be seen from the graph 7.9(b), 7.10(b), 7.11(b), and 7.12(b) on the right, the drone position remains stable within 0.1m and the drone angle within 0.6 degrees even though the drone's propeller is a random number. After the model generated by reinforcement learning is applied to control the output of the other three propeller blades, the drone has a small perturbation at the beginning but stays in a relatively stable state for the rest of the time.

7.4.4 Results

As seen from the tests and analysis above, the model created by reinforcement learning can handle the faults that arise very well. Reinforcement learning maintains good stability for stabilising the UAV by comparing data from the UAV without the applied model and with the applied model.

When the drone was not applied to the model, i.e. when the drone encountered the designed faults scenarios, the position of the drone deviated greatly and the drone deflected violently. The drone is then in a very unstable state. When the UAV applied the model obtained by reinforcement learning to the same faults scenarios, the results were in stark contrast to the previous data The drone remains stable in the initial position of the UAV after a very small amount of jolting. Almost no external disturbances were applied. The drone shows good handling of external changes.

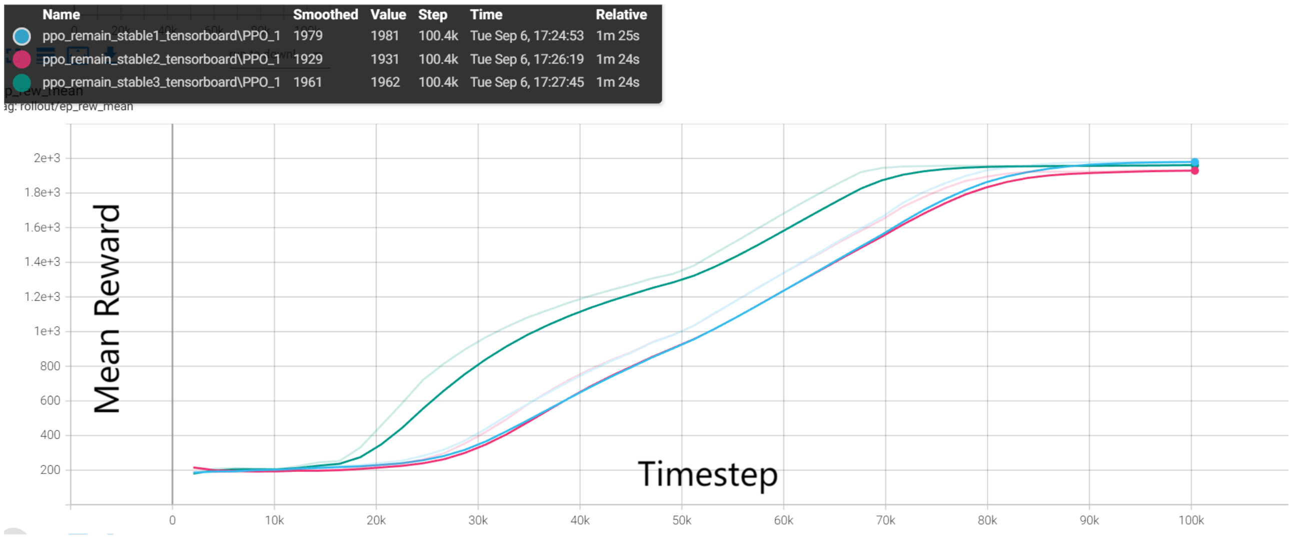
To a certain extent, the performance of the resulting model can be improved by adjusting the number of training sessions, designing an optimized reward function, and using different reinforcement learning algorithms.

8 Evaluation of the Model

8.1 Randomness of Reinforcement Learning during Training Session

Each trained model is evaluated with deterministic actions. Thus, when the model is tested in a particular state, the model performs the same action every time. But reinforcement learning generates the model randomly in each model training, given a certain state and parameters in time, PPO. Figure 8.1 shows simulation results of the 3 training processes and figure 8.2 tests of the 3PPO models. The "PPO_remain_stable1_tensrboard\ppo1" represent model 1. The next curve is similar to this.

Table 8.1 provides the mean rewards for each of the PPO runs.



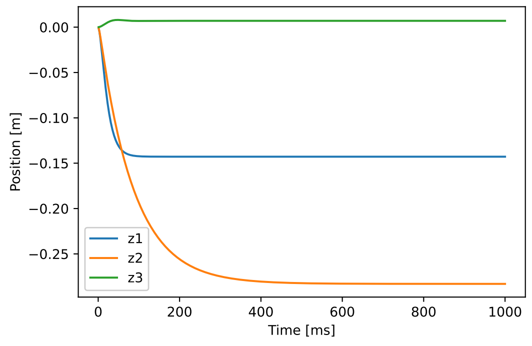
	Name	Smoothed	Value	Step	Time	Relative
●	ppo_remain_stable1_tensorboard\PPO_1	1979	1981	100.4k	Tue Sep 6, 17:24:53	1m 25s
●	ppo_remain_stable2_tensorboard\PPO_1	1929	1931	100.4k	Tue Sep 6, 17:26:19	1m 24s
●	ppo_remain_stable3_tensorboard\PPO_1	1961	1962	100.4k	Tue Sep 6, 17:27:45	1m 24s

Figure 8.1: Training History of three Stabilisation Model (training times: 100000) .

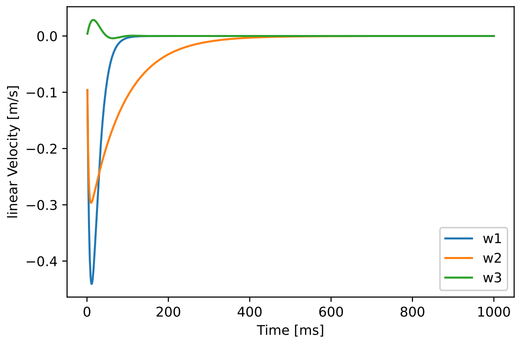
The mean reward is different for each model, and also the model performs differently in the real world. This is because the reinforcement learning process is stochastic, and so are the results each

Model of Different Training	Model 1	Model 2	Model 3
Mean Reward	1979.72582	1940.12422	1998.65327

Table 8.1: Mean Reward.



(a) Attitude (Z) Data of three Models.



(b) Vertical Speed (W) Data of three Models.

Figure 8.2: Data of three Models.

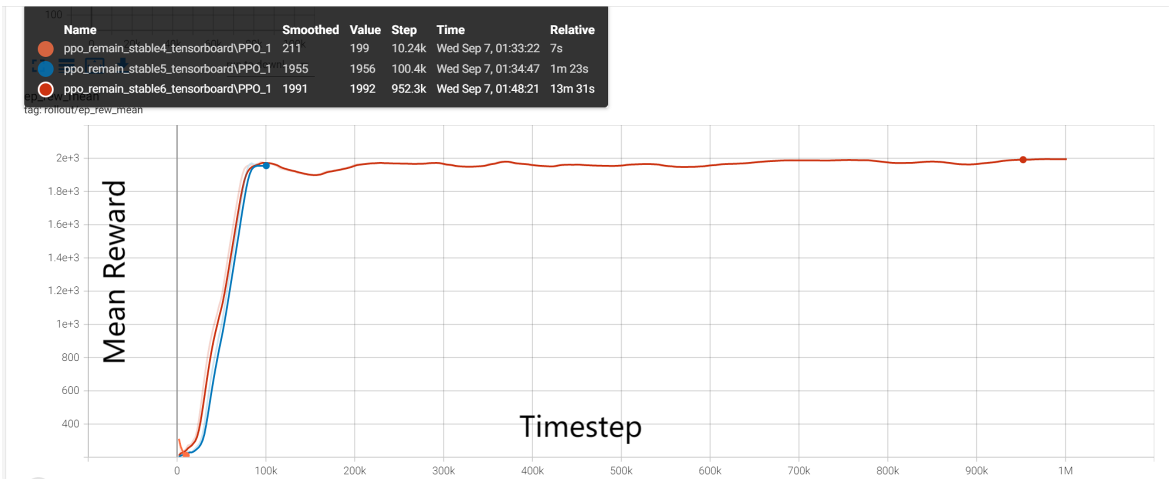
time. However, each model also performs differently in the actual simulation, and the graph 8.2 shows that model3 performs best. mode3 causes the UAV to oscillate less at first and for a shorter period , so that the UAV remains stable for the smallest distance from its initial position. model1 is the next best performer. model2 is the worst performer among the three models.Also from table 8.1, model 3 has the highest mean reward. model 2 comes second and model 1 has a relatively low mean reward. Their mean reward is directly related to their performance.

It can be seen that even if the models are obtained by reinforcement learning, the models that perform well are selected by repeated training and comparison with each other.

8.2 Effect of Number of Training Steps

When the different number of training sessions for reinforcement learning is set to 10,000, 100,000, and 100,000, 3 different models will be created. The process of training the models, the mean reward of the models, and the results of testing the models are recorded.

Figure 8.1 shows simulation results of the 3 training processes. The "PPO_remain_stable4_tensrboard\ppo1"



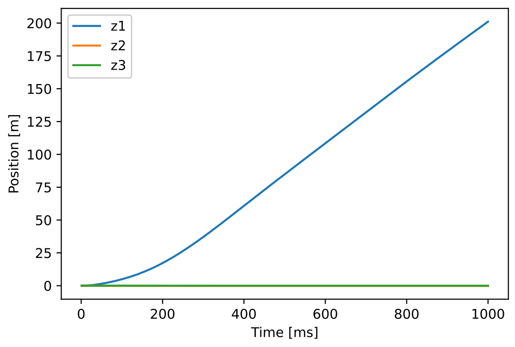
	Name	Smoothed	Value	Step	Time	Relative
●	ppo_remain_stable4_tensorboard\PPO_1	211	199	10.24k	Wed Sep 7, 01:33:22	7s
●	ppo_remain_stable5_tensorboard\PPO_1	1955	1956	100.4k	Wed Sep 7, 01:34:47	1m 23s
●	ppo_remain_stable6_tensorboard\PPO_1	1991	1992	952.3k	Wed Sep 7, 01:48:21	13m 31s

Figure 8.3: Training History of three Stabilisation Model (training times: 1e4, 1e5, 1e6) .

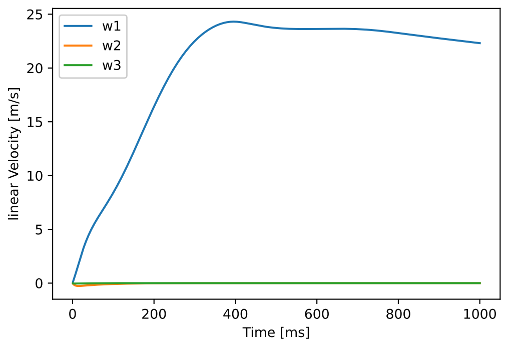
Model of Different Training	Model 4	Model 5	Model 6
Training Sessions	10000	100000	1000000
Mean Reward	175.12582	1960.32283	1998.57956

Table 8.2: Mean Reward.

represent model 4. The next curve is similar to this. It is found that model 4 yielded a significantly lower mean reward. And the comparison based on other images shows that the learning process of our model stopped, resulting in an incomplete model obtained from training. The results of model 1 in the figure 8.4 test showed that the drone did not meet the requirements of the mission, the drone kept going up and up and the drone was not able to maintain balance. At the same time, model 4 has the smallest mean reward (175.12582) in the table 8.2.



(a) Attitude (Z) Data of three Models.



(b) Vertical Speed (W) Data of three Models.

Figure 8.4: Data of three Models.

9 Discussion and Conclusion

By modelling the quadrotor UAV, the flight of the UAV can basically be simulated. Using the reinforcement learning PPO algorithm, after a limited number of learning sessions, the resulting model is able to control some basic operations of the UAV, such as stable hovering, ascent and descent, and horizontal movement of the UAV. However, the horizontal movement still suffers from the problem of the UAV flipping once, and although in the simulation, the UAV remains stable and reaches the target position after flipping, in reality, the UAV does not allow this to happen in many cases.

When UAVs encounter faults scenarios, for which a propeller is born faulty Using the reinforcement learning PPO algorithm, the model derived after a finite number of learning sessions achieves good robustness while controlling the UAV to maintain stability.

The objectives set out at the beginning of the text were largely achieved. Drones have the ability to deal with faults to a certain extent. The drone maintains good robustness in handling failures. Compared to the case where no model is applied, the UAV is able to remain largely stable in its initial position in the event of failure.

However, when the research derives a model through reinforcement learning, the resulting model may vary slightly due to the uncertainty in the training process, and to eliminate chance, we need to train the model several times and find the most suitable model. In the future, in order to improve the performance of the trained models, we can optimize the reward function by assigning different weights to the influence factors in the reward function. Finally, the best-performing model is selected and compared under the same conditions.

By increasing the number of training sessions, the mean reward of the reinforcement learning algorithm will tend to be more stable and higher. This does not mean, however, that the model needs to be built with a particularly high number of training sessions. The actual number of training sessions is also limited by the hardware, and it is important to find an appropriate number of sessions based on the hardware and the specific requirements of the task.

Selecting different reinforcement learning algorithms. For models where the external environment is known, a model-based reinforcement learning algorithm such as AlphaZero can be chosen, and when this works, AlphaZero can significantly improve the sample efficiency compared to methods without a model. At the same time, a complex task can be designed into different stages of the task so that the appropriate model is chosen separately to meet the needs of the task in a comprehensive way.

References

[1] I. C. A. Organization, *International civil aviation organization (icao). accident statistics, 2019*, `https://www.icao.int/safety/iStars/Pages/Accident-Statistics.aspx/` Accessed march 15, 2022.

[2] J. S. Achiam, *Exploration and Safety in Deep Reinforcement Learning*. University of California, Berkeley, 2021.

[3] T. Luukkonen, Modelling and control of quadcopter, *Independent research project in applied mathematics, Espoo*, vol. 22 2011, p. 22, 2011.

[4] S. Musa, Techniques for quadcopter modeling and design: A review, *Journal of unmanned system Technology*, vol. 5, no. 3 2018, pp. 66–75, 2018.

[5] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[6] J. Achiam., *Spinning up in deep reinforcement learning, benchmarks, 2018.* `https://spinningup.openai.com/en/latest//` Accessed march 15, 2022.

[7] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, *nature*, vol. 521, no. 7553 2015, pp. 436–444, 2015.

[8] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, Deep reinforcement learning: A brief survey, *IEEE Signal Processing Magazine*, vol. 34, no. 6 2017, pp. 26–38, 2017.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, Continuous control with deep reinforcement learning, *arXiv preprint arXiv:1509.02971* 2015, 2015.

[10] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in *International conference on machine learning*, PMLR, 2014, pp. 387–395.

[11] A. Paszke, Reinforcement learning (dqn) tutorial, pytorch tutorials, *Dostupno na: https://pytorch.org/tutorials/intermediate/reinforcementqlearning. html* 2018, 2018.

[12] V. Konda and J. Tsitsiklis, Actor-critic algorithms, *Advances in neural information processing systems*, vol. 12 1999, 1999.

[13] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *International conference on machine learning*, PMLR, 2018, pp. 1861–1870.

[14] K. Dally, Deep reinforcement learning for flight control: Fault-tolerant control for the ph-lab 2021, 2021.

[15] W. Koch, R. Mancuso, R. West, and A. Bestavros, Reinforcement learning for uav attitude control, *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2 2019, pp. 1–21, 2019.

[16] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, Control of a quadrotor with reinforcement learning, *IEEE Robotics and Automation Letters*, vol. 2, no. 4 2017, pp. 2096–2103, 2017.

[17] A. Milhim, Y. Zhang, and C.-A. Rabbath, Gain scheduling based pid controller for fault tolerant control of quad-rotor uav, in *AIAA infotech@ aerospace 2010*, 2010, p. 3530.

[18] P. McKerrow, “Modelling the draganflyer four-rotor helicopter,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA’04. 2004*, IEEE, vol. 4, 2004, pp. 3596–3601.

[19] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, “Quadrotor helicopter flight dynamics and control: Theory and experiment,” in *AIAA guidance, navigation and control conference and exhibit*, 2007, p. 6461.

[20] A. Quessy and T. Richardson, Quad2plane: An intermediate training procedure for online exploration in aerial robotics via receding horizon control, *arXiv preprint arXiv:2203.10910* 2022, 2022.

[21] W. shusen, *Wangshusen/deeplearning. [online] github.* `https://github.com/wangshusen/DeepLearning>[Accessed8September2022/` Accessed march 15, 2022.

[22] Spinningup.openai.com, *Proximal policy optimization*, `https://spinningup.openai.com/en/latest/algorithms/ppo.html#id3/` Accessed march 15, 2022.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347* 2017, 2017.