High-Performance Flight Control of Variable-Pitch-Propeller Quadcopters

Zhikun Wang

Department of Automatic Control and Systems Engineering
University of Sheffield

This dissertation is submitted for the degree of
Doctor of Philosophy

March 2022

Acknowledgements

I would like to thank many individuals for their support during my PhD study.

First and foremost, I would like to express my gratitude to my supervisors Prof. Roderich Groß and Prof. Shiyu Zhao for their guidance. I would not be able to complete my PhD without their strong support. The completion of this thesis significantly benefits from their effort and encouragement. Their patience and enthusiasm have encouraged me to consider furthering my research career.

I would also like to thank all the members in Prof. Groß's lab (Natural Robotics Lab) at the University of Sheffield and the Intelligent Unmanned Systems Laboratory at Westlake University, for all of their friendship, advice, and support for this project. To name a few, special thanks are given to Yue Gu, Matthew Dolly and Matthew Hall for their valuable advise. Thank you all for creating such a friendly and academic research environment.

I would also like to express my gratitude to my colleagues Zhenglin Li, Ke Sun, Tianran He, Kai Eivind Wu and Zhen Xi for their support and friendship in helping me work through my PhD study. I feel lucky to have them as my friends during my time at the University of Sheffield. Also, I would thanks Mr. Matthew Ham and Mrs. Renata Goddardin of the ACSE department for answering my questions, queries and so on.

Finally, I would like to share my appreciation to my parents Mrs. Kaiyun Xu and Mr. Mulan Wang for their sheer support and help all through these years. I would not have had such a PhD experience without their wholehearted support. Lastly, I would like to express my thanks to my girlfriend Mi Zhou for company during my entire Ph.D study, while completing a PhD of her own.

I am so grateful to have all these kind people travelling with me during my PhD journey.

Abstract

Variable-pitch-propeller (VPP) quadcopters are a new type of micro aerial vehicle. Compared to conventional fixed-pitch-propeller quadcopters, VPP quadcopters can not only generate positive but also negative thrust and hence fly upside down. They also outperform the conventional quadcopter in terms of control bandwidth and manoeuvrability. Despite the advantages of VPP quadcopters, their potential has not been well explored yet, though there exist some relevant studies in the literature. The aim of this thesis is to explore the unique features of VPP quadcopters in two important aspects. The first aspect is about flight safety: that is, how to ensure a VPP quadcopter fly safely even when a propeller is partially damaged. Two fault-tolerant controllers for different types of VPP quadcopters are proposed, respectively. It is found that fault-tolerant control of a VPP quadcopter exhibits unique and interesting features compared to conventional quadcopters. In particular, a separated-powered VPP quadcopter is fully controllable with only three actuators. With the proposed control law, a faulty centralized-powered VPP quadcopter can track the reference path. The system and simulation experiments are established and conducted in a physical driven environment called SimScape, to verify the effectiveness of the proposed control laws. Real experiments are conducted to verify the dynamic model of VPP. The second aspect is about flight agility: that is how to fully utilize the ability of VPP quadcopters to achieve highly agile flight. In particular, novel control laws are proposed that can successfully achieve the tic-toc flight task, which is one of the most challenging manoeuvrers of aerial vehicles. Thanks to the feature that a VPP could generate either positive or negative thrusts, such a task is achievable by VPP quadcopters. The tic-toc flight is challenging to achieve partially because it is far from the hovering equilibrium state. A state-of-the-art reinforcement learning algorithm is employed to solve this problem. An extension algorithm for performance adjustment and controller redeployment is proposed as well. Simulation results verify the effectiveness of our control laws.

Contents

List of Figures

List of Tables

List of Abbreviations

UAVs Unmanned Aerial Vehicles

MEMS Micro Electro Mechanical System

IMU Inertial Measurement Unit

AI Artificial Intelligence

ETH Eidgenössische Technische Hochschule

GRASP General Robotics, Automation, Sensing and Perception

VPP Variable Pitch Propeller

RL Reinforcement Learning

SVPP Separated-powered Variable Pitch Propeller

CVPP Centralized-powered Variable Pitch Propeller

LQR Linear Quadratic Regulator

VTOL Vertical Take-off and Landing

DJI Da-Jiang Innovations

FMA Flying Machine Arena

MIT Massachusetts Institute of Technology

GPS Global Positioning System

PID Proportional Integral Derivative

LQG Linear Quadratic Gaussian

SMC Sliding Mode Control

FTC Fault Tolerant Control

AFTC Active Fault Tolerant Control

PFTC Passive Fault Tolerant Control

FDI Fault Detection and Isolation

SO(3) Special Orthogonal Group in Three Dimensions

SE(3) Special Euclidean Group in Three Dimensions

ILC Iterative Learning Control

MDP Markov Decision Process

MCTS Monte Carlo Tree Search

TD Temporal Difference

SARSA State, Action, Reward, estimate State, and estimate action

DQN Deep Q-Learning

NN Neural Network

PG Policy Gradient

A3C Asynchronous Advantage Actor Critic

CPU Central Processing Unit

DPG Deterministic Policy Gradient

DDPG Deep Deterministic Policy Gradient

TD3 Twin Delay Deep Deterministic Policy Gradient

TRPO Trust Region Policy Optimization

KL Kullback–Leibler

PPO Proximal Policy Optimization

DPPO Distributed Proximal Policy Optimization

SAC Soft Actor-Critic

DOFs Degrees of Freedom

NACA	National Advisory Committee for Aeronautics
CNC	Computer Numerical Control
ESC	Electronic Speed Control
PWM	Pulse-Width Modulation
RAM	Random Access Memory
GPU	Graphics Processing Unit

Chapter 1

Introduction

1.1 Background

Quadcopter unmanned aerial vehicles (UAVs) have been widely applied in both academic and industrial domains, due to their simple mechanical and control system structures [1, 2]. In 2010, Parrot released the world's first commercial quadcopter named AR Drone [3]. Since then, the public awareness of quadcopters has rapidly raised and become popular commercial products [4]. In the last decade, with the rapid development of micro electro mechanical system (MEMS), micro inertial measurement unit (IMU), and artificial intelligence (AI), the number of research groups on quadcopter UAVs in the globe has increased rapidly, such as the Institute for Dynamic Systems and Control in Eidgenössische Technische Hochschule (ETH) Zurich, Aerospace Engineering in Georgia Institute of Technology, University of Pennsylvania's General Robotics, Automation, Sensing and Perception (GRASP) Lab, and Stanford University's Aerospace Robotics Lab [5], to name a few.

Nowadays, quadcopters have been successfully applied in many areas [6] such as environmental monitoring, agricultural plant protection [7], earthquake relief [8], aviation safety [9], electronic countermeasures [10], and aerial photography [11].

In the future, they are expected to be commonly utilised in day-to-day tasks, such as parcel delivery (Amazon PrimeAir [12]), air rescue (Alec Momont [13]), and passenger transportation (Ehang [14, 15]). Despite the fast development of quadcopters, they still face some key challenges if they are applied into some tasks such as autonomous aerial vehicles transporting human passengers. These tasks require the flight platforms such as quadcopters to have strong safety guarantee, high agility, anti-interference, and fault-tolerance capabilities [16–19]. Therefore, the research on agile and safety-critical flight control of quadcopters have high application value and broad market prospects.

It is notable that the mechanical and dynamical structures of quadcopters have changed little since they were invented. That is because the structure is sufficiently simple and also meets the requirement of many present tasks. In particular, a quadcopter usually consists of a body frame and four motors. Each motor drives fixed-pitch propeller. By adjusting the rotational speed of the four motors, a quadcopter can adjust its attitude and fly following pre-designed trajectories. Although the existing quadcopter structure design has the advantages of low mechanical and control complexity, it also has disadvantages such as low control bandwidth (due to motor speed control) and relatively weak manoeuvrability.

As a relatively new type of quadcopters, variable pitch propeller (VPP) quadcopters exhibit superior performance than the conventional fixed-pitch ones [20]. The difference between a VPP and a fixed-pitch propeller is that a VPP could adjust the pitch angle the propeller by an independent actuator. This could bring some attractive benefits. First, compared with the rotation speed control of a fixed-pitch propellers, the pitch angle control of a VPP has higher control bandwidth, bringing higher agility to VPP quadcopters [21, 20]. Second, a VPP can generate thrusts in either upward or downward directions. This novel property brings benefits to flight control performance. For example, a VPP quadcopter will be able to fly upside down steadily, which is not possible to achieve by a fixed-pitch quadcopter. This provides VPP quadcopters with strong manoeuvrability. It also enables VPP quadcopters to recover swiftly from largely disturbed attitudes back to stable hovering conditions, which is important from safety point of view.

Due to these interesting features, VPP quadcopters have attracted increasing attention in recent years [22–24]. However, the great potential of VPP quadcopters has not been fully explored yet. Motivated by that, this thesis aims to explore the unique features of VPP quadcopters and further develop control algorithms to achieve a safe and agile flight performance.

1.2 Problem Statement

In this thesis, we focus on two aspects of VPP quadcopters.

The first aspect is about flight safety. The demand for safety-critical flight missions of quadcopter UAVs increases rapidly. For example, parcel delivery and passenger transportation by quadcopter UAVs are operated in urban areas. These tasks pose high requirement on the flight safety of quadcopters to against external disturbance and component failure. Robust control is a widely studied approach that could overcome minor failures such as actuator efficiency loss or thrust error [25]. For major failures such as actuator loss, specific fault-tolerant controllers must be designed [26]. Despite the existing studies, safe-critical flight of conventional fixed-pitch quadcopters still face big challenges. First, the current research of safety flight control for conventional quadcopters is not good enough for practical usage. Even under a well designed fault-tolerant controller, a quadcopter with one actuator loss will spin rapidly. This will be a huge shortcoming for manned transportation applications, since such an uncontrollable rapid rotation can cause serious damage to the health of the carried passengers. Second, the research into safety control of VPP quadcopters has been scant. Due to the difference in structural design, the control dynamics between conventional fixed-pitch quadcopters and VPP quadcopters are quite different. There are not many studies that detailed analyse the uniqueness of VPP quadcopters. Its safety control potential is not fully utilized. In this thesis, we study the dynamics of VPP quadcopters to fully explore its unique features. These discovered features can be used to design fault-tolerant controllers to improve safe flight performance. These controller designs contribute to the future applications of the VPP quadcopter.

The second aspect is about flight agility. Flight agility not only reflects the ability for entertainment but also serves as a proof of recovery ability [27, 28]. There are many researchers studying on manoeuvrer control for conventional quadcopters, such as passing through a vertical window [29], juggling a ball [30], and pole acrobatics [31]. However, these results could not directly reflect the agility potential of VPP quadcopters. VPP quadcopters have the ability to rapidly change their thrust between positive and negative like a helicopter. In this thesis, we want to utilize such a specific ability of VPP quadcopters by studying tic-toc manoeuvrer. Tic-toc is one of the hardest manoeuvrers that a helicopter can do, which requires the aircraft to have the ability to generate inverse thrust and the controller to have good balancing capabilities. The exist research results have not shown that any control system can achieve such a manoeuvrer without human demonstration. It is a challenging task because the whole tic-toc manoeuvre has no equilibrium point and metrics for such a manoeuvre is hard to define. Since tic-toc manoeuvrer can be performed by expert pilots through helicopter, we use the algorithm that is considered the most likely to achieve human-level performance, namely reinforcement learning (RL) [32]. Such an algorithm is proved to have the ability to achieve good control performance within complex systems by learning from the interaction between the agent and the environment [33, 34]. Moreover, RL has been successfully applied to UAVs control areas, such as helicopter manoeuvrers [35], throw-and-hover [36] and attitude control [37]. This thesis aims to develop a RL based control algorithm that can perform tic-toc manoeuvre through VPP quadcopters.

1.3 Objectives

The aim of this thesis is to study the unique characteristics of the VPP quadcopters and further develop control algorithms for VPP quadcopters to achieve stable, safe, and agile flight performance. The specific objectives of this thesis are as follows:

1. Conduct detailed literature review on the existing studies of control of conventional fixed-pitch and VPP quadcopters. Especially focus on the state-of-the-art algorithms of safety-critical and high-agility flight control.

2. Establish the dynamic model of VPP quadcopters and identify their unique dynamic features. Conduct real experiments to verify the dynamic model of a single VPP actuator.

3. For separated-powered VPP (SVPP) quadcopters, design fault-tolerance controllers to ensure safe flight when one of the actuators is broken. Validate the controller within simulation experiments.

4. For centralized-powered VPP (CVPP) quadcopters, design fault-tolerance controllers to ensure safe flight when one of the propeller is stuck. Demonstrate the controller within simulation experiments.

5. Analyse the tic-toc manoeuvrer in detail to develop several metrics for this manoeuvrer. Apply RL algorithms to develop new controllers to realize high-agile tic-toc flight control without human demonstration.

6. Design extension control system to further adjust the controller performance. Demonstrate the control system within simulation environment.

1.4 Research Contributions

The contributions of the thesis are listed as:

1. We conduct experiments on a real VPP platform to examine its mechanical and dynamical properties. With the data obtained from the experiments, we analyse the dynamical performance, control feasibility, and unique features of VPP actuators. Not only the dynamical model of VPP actuators is verified by the experiments, we also obtain a new finding that a VPP quadcopter can control the thrust and torque of each actuator independently, which is a favourable feature for high-performance flight control. This part of fundamental work lays a solid foundation for further controller design.

2. We conduct controllability analysis for an SVPP quadcopter who has one VPP actuator lost. The research shows a novel finding that a faulty SVPP quadcopter system still remains fully controllable with merely three VPPs. We further design a linear quadratic regulator (LQR) controller and validate the system within simulation environment. The demonstration shows that a faulty SVPP quadcopter can fly steadily and track desired trajectories even subject to wind disturbance or noise. The results verify our presented findings.

3. We present the dynamic analysis of a CVPP quadcopter in the presence of one propeller stuck. Based on a controllability decomposition method, an H_∞ fault-tolerance control system is designed for a faulty CVPP quadcopter with only three control inputs. Such a controller can stabilize the faulty VPP quadcopter when hovering, and can track a reference trajectory with an uncontrollable error in the yaw angle.

4. A novel self-learning aerobatic flight control system is designed. It is the first control system that can perform tic-toc manoeuvrer through a VPP quadcopter. Moreover, such a controller can perform infinite flipping manoeuvrers through a planar VPP quadcopter.

5. A reinforcement learning extension algorithm that can further adjust the flight performance according to the requirements of deployment without retraining. This extended algorithm can modify the parameters to delicate adjust the tic-toc manoeuvre. It enhances the robustness and generalization capabilities of such a control system.

1.5 Thesis Outline

The contents of each chapter is outlined as follows.

Chapter 1 This chapter introduces the background, motivation and objectives of the thesis. Other information, such as publications and the structure of this thesis, is summarized in the end.

Chapter 2 This chapter first compares three types of quadcopters and then highlights the advantages of VPP quadcopters. We further review relevant quadcopter control algorithms especially those for agile and safe flight control tasks. Finally, basics of reinforcement learning are introduced and related algorithms are reviewed.

Chapter 3 This chapter first analyses the dynamics of VPP actuators and proposes theoretical coefficients for aerodynamic properties based on airfoil and propeller simulation software (XFOIL and QPROP). Then, we develop a VPP actuator testbed and verify the theoretical values. Furthermore, the general dynamical model of the VPP quadcopter is introduced. A simplified 2D planar model, which is subsequently used to design an RL controller, is also given.

Chapter 4 This chapter considers flight control of an SVPP quadcopter in the presence of one actuator loss. We first analyse the problems faced by conventional quadcopters when dealing with such failures. In particular, a fixed-pitch quadcopter would keep rotating with high spinning rate when one propeller is lost. The controllability analysis of both conventional and VPP quadcopters are presented in the next to find that an SVPP quadcopter can keep fully controllable with only three VPPs. Then, we propose a simple LQR controller based on such findings. The simulation results show the designed controller can not only stabilize the SVPP when one VPP is lost, but also keep the dynamic system fully controllable. Such a property is favourable for many tasks such as passenger transportation.

Chapter 5 This chapter considers flight control of a CVPP quadcopter when one actuator is faulty (in particular, the propeller is stuck and could not tune its pitch angle due to actuator fault). We first review the existing fault controllers for CVPP and SVPP quadcopters and highlight the differences between these two types of VPP quadcopters. Then, Kalman decomposition is used to decompose the quadcopter system into controllable and uncontrollable components, which are subsequently used to design a linear controller. The demonstration shows, although the yaw angle of a faulty CVPP quadcopter is uncontrollable, it can exhibit a slow self-spinning speed in the presence of a propeller fault.

Chapter 6 This chapter first analyses the tic-toc manoeuvrer and shows that it is similar to a juggling called “Devil Stick”. With this idea, we simplify the 3D dynamical model to a 2D planar version. Then, we fully decompose the tic-toc manoeuvre and define several metrics to evaluate it. After comprehensive comparison of different types of reinforcement learning algorithms, we apply deep deterministic policy gradient to develop a new controller to achieve the tic-toc manoeuvrer. The performance of the proposed controller is carefully evaluated and validated by numerical simulation.

Chapter 7 This chapter summarizes the thesis with a synopsis of the research findings, methods and results. In addition, the limitations of this research are discussed and future research opportunities in this field are speculated.

1.6 Publications

The present thesis is built associated with the following publications:

Published papers

Zhikun Wang, Roderich Groß, and Shiyu Zhao. “Controllability analysis and controller design for variable-pitch propeller quadcopters with one propeller failure.” *Advanced Control for Applications: Engineering and Industrial Systems*, vol. 2, no. 2, p. e29, 2020.

Zhikun Wang, Roderich Groß, and Shiyu Zhao. “Control of Centrally-Powered Variable Pitch Propeller Quadcopters Subject to Propeller Faults.” *Aerospace Science and Technology*, vol. 120, p. 107245, 2022.

Zhikun Wang, Roderich Groß, and Shiyu Zhao. “Aerobatic Tic-Toc Control of Planar Quadcopters via Reinforcement Learning.” *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 2, pp. 2140-2147, 2022

Chapter 2

Literature Review

This chapter reviews the dynamics and control algorithms of quadcopters. In particular, this chapter analyses and compares the performance and properties of different quadcopter structures. The comparison result demonstrates that VPP quadcopters have significant performance advantages for safety-critical and agility required applications. Existing approaches for flight control of either helicopters or quadcopters aspects are reviewed. Other related methods of quadcopter flight control such as trajectory design, manoeuvrer flight control, and fault tolerance control are also reviewed.

2.1 Quadcopter Development History

2.1.1 Overview

Recent years have witnessed the rapid development of commercial UAVs across the world. Generally, UAVs can be divided into three types:

1) Fixed-wing aircraft. This type of UAV relies on the propulsion system to generate forward power and then generate the lift force through aerodynamics. Therefore, it has good flight efficiency, but cannot achieve vertical take-off and holding performance, which makes it usually requires sufficient take-off

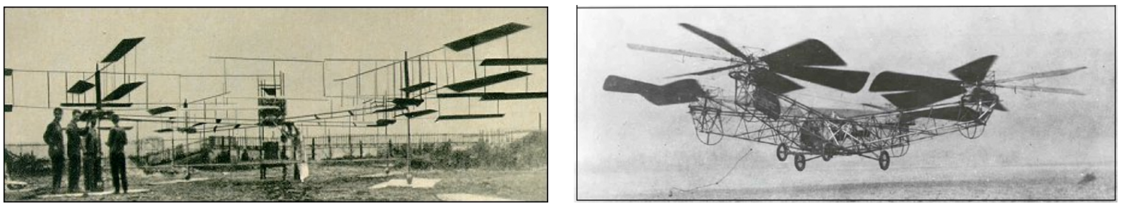
and landing distance. Nowadays, many applications require UAVs to have good manoeuvrability for operation at low-altitude and low-speed, where the fixed-wing aircraft cannot meet these requirements.

2) Helicopters. It is an aircraft that uses the main rotor to generate lift force and adjusts the pitch angle to change its flight attitude. It has a tail rotor to balance the torque generated by the rotation of the main rotor. Since the helicopter uses the lift generated by the main rotor to offset the gravity, it can achieve vertical take-off and flying. But it also brings shortages such as low efficiency, complex mechanical structure, and difficult control. Moreover, due to the large and complicated structure of the main rotor, the helicopter is dangerous when flying and has poor fault tolerance ability.

3) Multi-copter UAVs. A multi-copter UAV is composed of multiple rigid actuators, where each actuator contains a motor and a propeller. This class of UAV usually has a symmetrical layout which makes its control strategy simple and direct. Since the rotors are located at the edge of the aircraft, this type of UAV can directly control the thrust by changing the motor speed, and thereby generate the torque by the differences of thrusts. Therefore, in addition to VTOL, the multi-copters has the characteristics of simple structure and well-understood operation properties. It also suffers from low endurance limited by the battery. Among all the multi-copter aircraft types, the most typical and widely used type is the quadcopter [38].

Compared with other UAVs, quadcopters are now widely used due to the simple structural design and also the high-speed attitude adjustment [38]. The simplified structure design gives highly reliable performance and the low maintenance difficulty to itself. In addition, the advanced control theory provides the possibility for quadcopters to achieve robust or manoeuvrable flight performance. Because of these advantages, quadcopters have received extensive attention in the field of UAVs in recent years [6].

The history of the quadcopters can be traced back to the first quadcopter shown in Fig.2.1 (a), which was designed in 1907 by Louis Breguet called “Breguet-Richet Gyroplane” [39]. It is the first vertical take-off and landing (VTOL) aircraft that can carry people. However, this huge quadcopter can only maintain a few feet high and cannot fly long distances. The first quadcopter that can successfully stay in the air for minutes and move for a short distance was invented in 1920 named “Oehmichen NO.2”. US Air army developed a quadcopter with six propellers on each actuator called “de Bothezat helicopter” shown in Fig.2.1 (b) in the early 1920s [40]. At that time, the large quadcopters were controlled by mechanical methods and powered by fuel engines. Due to the lack of modern electronic technology, the control of this multi-copter is inaccurate and slow. All theses shortages make these prototypes hard to resist air friction and interference to maintain stability.



(a) (b)

Figure 2.1. Examples of giant multi-rotors helicopter built in early 1900s years, where (a) is the Breguet-Richet Gyroplane, reprinted from [39] and (b) is the De Bothezat helicopter, reprinted from [40].

Different from the early years, with the development of manufacturing, batteries, chips and control technology, the small size UAVs are becoming more popular in recent years. A typical modern quadcopter includes several components such as environmental awareness sensors (including radars, camera, inertial measurement unit, etc.), an onboard control system (including a high-level computer responsible for navigation and decision-making and a low-level microcontroller for high-frequency electronic system control to achieve real-time performance), a power system (including batteries, electronic speed controllers, motors and propellers) and a remote control system (includes communication link, 3D tracking system and ground station or remote controller), which are shown in Fig. 2.2. All of these systems allow the quadcopter to know its status and how to fly [41].

In recent years, with the rapid development of the global commercial quadcopters industry, many well-known products have been published, such as Parrot AR.Drone [3], Da-Jiang Innovations (DJI) Phantom series [42] (see Fig. 2.3(a)) and swarm quadcopters [43]. In addition to the small quadcopters, the requirement for urban transit and delivery has promoted the development of large manned quadcopters, such as Bell Boeing Quad TiltRotor (C-130 sized military transport) [44], Airbus [45] and EHANG autonomous aerial vehicle [14] (see Fig. 2.3(b)).

In the future, quadcopter UAVs are expected to be routinely utilised in common, day-to-day tasks, such as parcel delivery or passenger transportation. Many of these tasks are safety-critical and require UAVs to achieve high flight performance in terms of both agility and safety [16–19]. Thus, with the increasing and extensive application of quadcopters, the requirements of good manoeuvrability for normal use and high

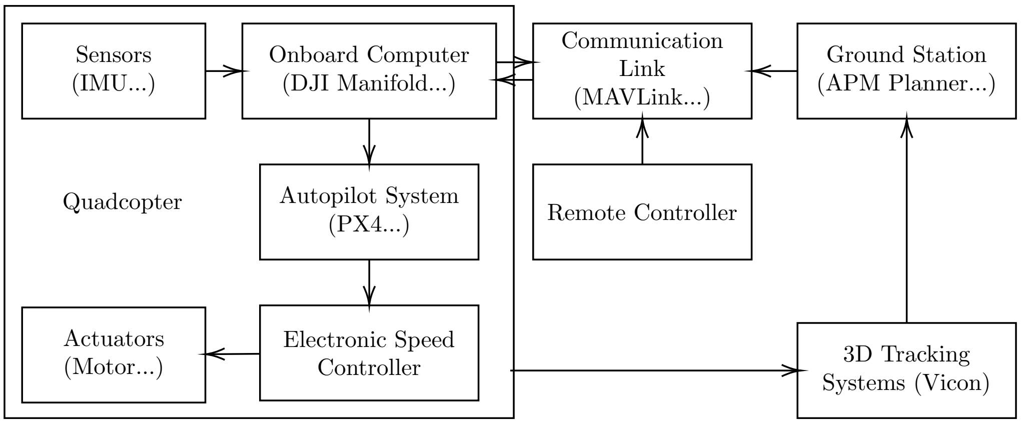


Figure 2.2. The basic components of a typical quadcopter system.

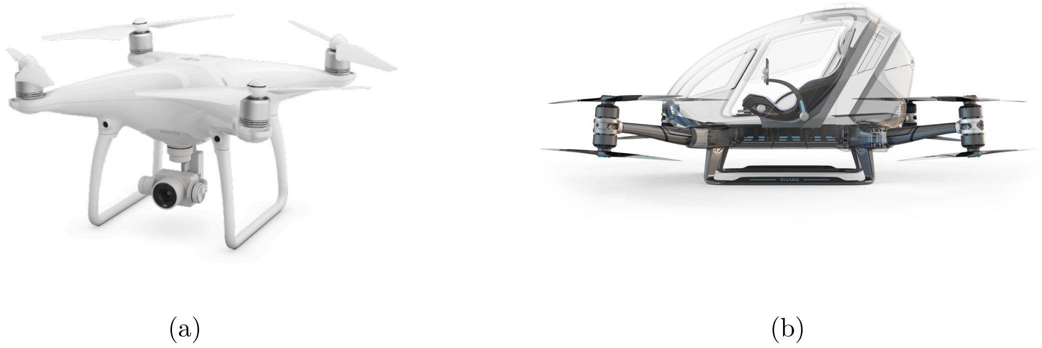


Figure 2.3. Examples of quadcopter applications. (a) is the DJI Phantom 4, reprinted from [42], which is widely used for photography and filming. (b) is the EHANG 184 autonomous aerial vehicle, reprinted from [14], which is a well-known electric passenger transportation autonomous aerial vehicle.

fault tolerance in emergency situations are getting higher and higher. The following sections will investigate the improvements of the safety and agility of quadcopters, from the structure to the controller, and the normal flight control to the emergency control.

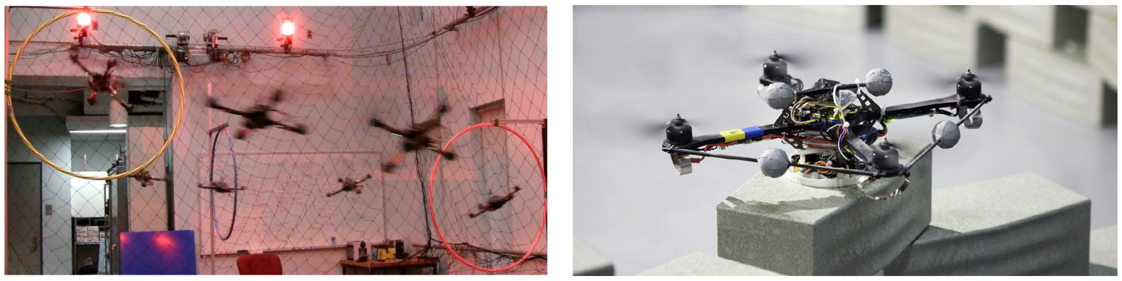
2.1.2 Quadcopter Platform Development

Since the quadcopters have experienced decades of development, many universities and research institutes have established their own quadcopter experimental research platforms. Most of the early quadcopter platforms were designed and manufactured by the research teams. Although the construction of these platforms requires a lot of effort, the performance of the platforms cannot be guaranteed.

The Stanford Testbed of Autonomous Rotorcraft for Multi-Agent Control (STAR-MAC) is one of the quadcopter research platforms for outdoor flight. It uses the onboard IMU to detect and calculate its own states and attitude. This platform has been used to validate different control algorithms including trajectory planning and aggressive manoeuvring control [46–48].

In addition to the quadcopter platform developed by the research group, many commercial companies have published their commodity quadcopters. Therefore, most of the research groups now turn to buying these products and customizing them to build the platforms to meet their requirements. For example, researchers from ETH use a customized Ascending Technologies X-3D quadcopters by replacing all original onboard electronics by themselves to achieve a quick dynamic sensor and response (with 800Hz sampling frequency of the gyroscope). Such a customization results in the two control modes of quadcopters. The first mode is used for ordinary flying control, where the quadcopter follows the usually commends via the 50 Hz communication system. In the case of manoeuvring, which is the second control mode, the quadcopter follows the 1 kHz open-loop command given from an onboard computer system instead. The quadcopter shown in Fig. 2.4 (b) can perform multi-flips [49] and quadcopter inverted pendulum [50]. Moreover, to achieve high-precision detection, ETH Flying Machine Arena (FMA) is used. FMA is an indoor aircraft flying test environment with a Vicon motion capture system that can provide 200 Hz detection frequency with millimetre accuracy [51]. With such a platform, the quadcopter can complete many demonstrations that requires quick response to different commands, such as ball juggling [30] and cooperative ball throwing or catching [52]. Their research fully demonstrate the agility potential and the benefits of this well-designed quadcopter platform. Due to the superior platform performance, a similar quadcopter testing platform (shown in Fig. 2.4 (a)) is built by the Vijay Kumar Lab at the University of Pennsylvania to validate the minimum snap trajectory generation and geometric control [53].

In general, a quadcopter platform with a strong power system, a fast dynamic system, and an accurate position and attitude sensing system is very important for research. In addition to the above quadcopter platforms, many other quadcopter platforms have been widely used, such as the DJI Matrice series, parrot drones series, etc.



(a) (b)

Figure 2.4. Examples of the quadcopter platforms where (a) is the Vicon motion capture system, reprinted from [53] and (b) is the ETH customized Ascending Technologies X-3D quadcopter, reprinted from [51].

2.1.3 Quadcopter Structure Development

The mechanical structure of the quadcopter has not changed much since its invention. An ordinary quadcopter structure usually has a rigid body and four actuators at the end of its body arm. Each actuator is constructed by a straight connection of motor and propeller, see Fig. 2.5 (a), which makes it easy to be assembled and repaired. However, this actuator mechanism can only provide positive thrust perpendicular to the plane of the body, resulting in a lack of agility and stability when flying [54].

To enhance the stability of the quadcopter subject to strong winds, a research group from Tohoku University and Kanazawa Institute of Technology changes the quadcopter tilt angle to 20 degrees, which is shown in Fig. 2.5 (d) [55]. However, although this design can improve the stability of the quadcopter to a certain extent, compared with ordinary structure, this structure will reduce the maximum thrust to 94%. Moreover, the whole quadcopter system efficiency is reduced because the horizontal components of the force cancel each other out. Therefore, such an improvement can not meet the requirements of improving flight safety and agility.

Another type of quadcopter tends to improve the thrust efficiency of a single actuator by applying two motors in a coaxial pair (e.g. X-8 quadcopter shown in Fig. 2.5 (b)). These two motors rotate in the opposite directions to provide the same direction thrust. Such a mechanism design is applied in passenger transportation applications, shown in Fig. 2.3 (b). As can be found from Fig. 2.5 (c), two separate coaxial propellers can counteract the torques generated by the rotation of blades [56]. In addition to this, the coaxial mechanism design can increase the lift thrust of the quadcopter and maintain a reasonable size and enhance the control sensitivity for the yaw angle. However, this design will increase the weight due to the structure of coaxial propellers. The actuator efficiency will be reduced because the inflow of the lower motor is influenced by the upper motor [57]. Although the coaxial motor design can increase the maximum thrust and maintains the overall diameter, it also increases the self-weight, cost and structural complexity, the usage of the coaxial motor pair is likely to be coupled with disadvantages.

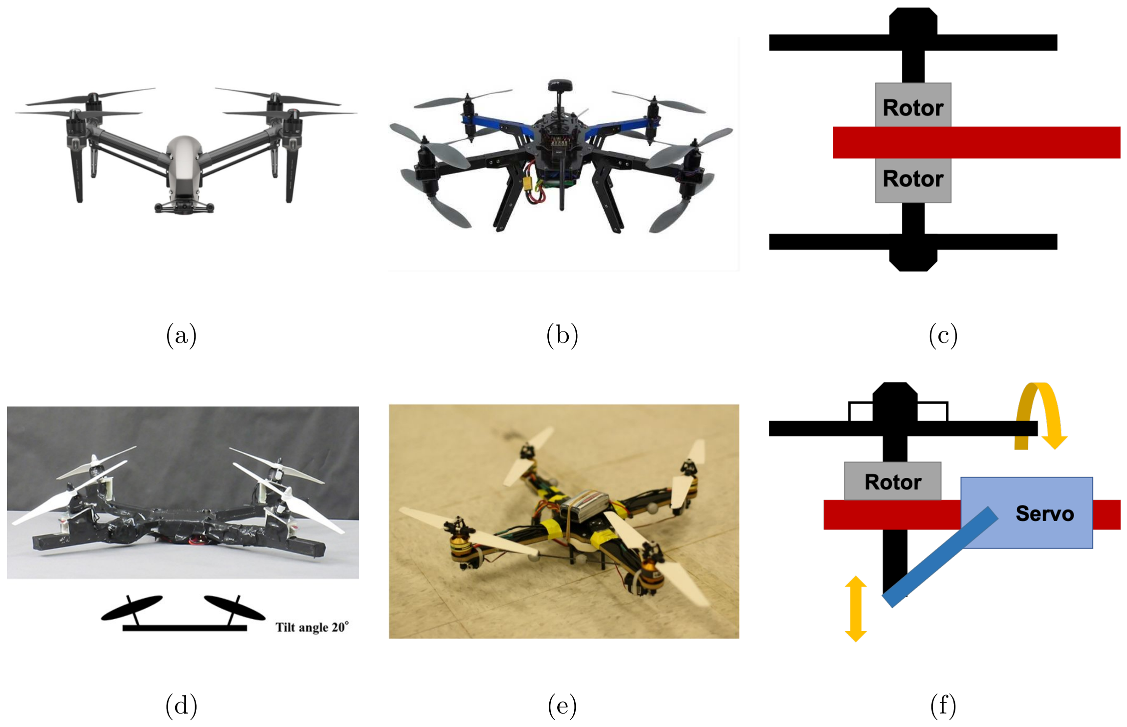


Figure 2.5. Examples of quadcopter actuation structure comparison. (a) is an ordinary quadcopter actuator structure of the DJI INSPIRE 2, image reprinted from [58]. (b) is a coaxial propeller actuator structure of the 3DR X8 quadcopter, image reprinted from [59] and (c) is the sketch of it. (d) is the Tohoku's quadcopter with 20°tilt angle, image reprinted from [55]. (e) is the MIT's VPP quadcopter, image reprinted from [21] and (f) is its corresponding sketch.

In addition, the researchers from Massachusetts Institute of Technology (MIT) use variable pitch quadcopter to bridge the gap between agility and efficiency [22]. The variable pitch actuator is derived from a fixed-wing aircraft where the motor shaft is connected with the propeller at one end and a servo at the other. The servo can provide additional translational force to the shaft to change the angle of the propeller and thereby change the thrust, shown in Fig. 2.5 (e) and (f).

According to the result of the simulated and practical experiments, the response time of changing the propeller pitch angle to reach the target thrust is much faster than that of changing motor spinning speed. Moreover, compared with the normal actuators, VPP actuators can provide double the thrust change rate (change motor speed and pitch angle together) [21]. As what the MIT research group demonstrated in experiments, VPP quadcopters can perform inverted flight and 360 degrees translating back-flip [20]. The results show that the VPP actuator structure makes the quadcopter more flexible and well balances the control complexity and manoeuvrability.

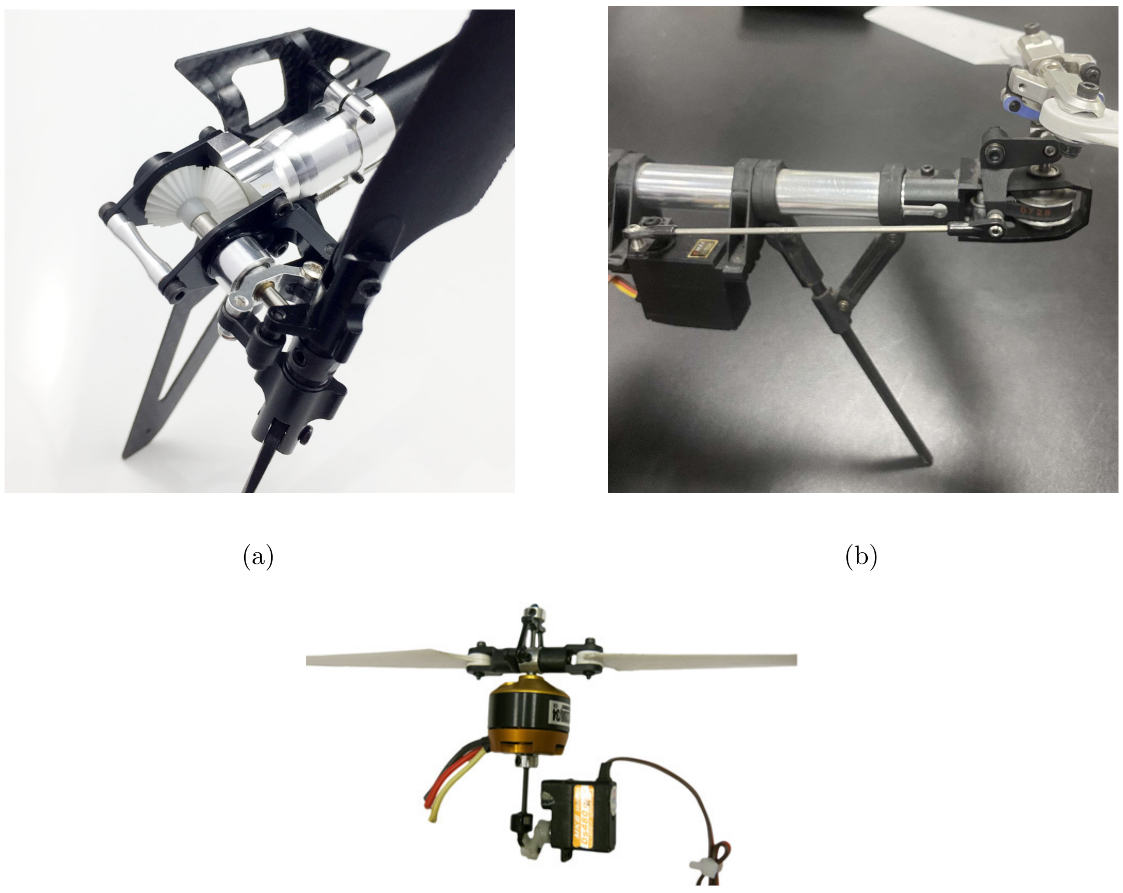
Compared to fixed-pitch propellers, VPPs have higher thrust change rate, higher actuator saturation limitation and the ability to provide reversed thrust. These features enhance the manoeuvrability of VPP quadcopters [21]. The ability of autonomous flip and inverted flight indicates that the VPP quadcopter has high manoeuvrability when flying in complex and harsh environments. Compared with ordinary quadcopters, its manoeuvrability is more practical in enhancing flight safety.

Although the high complexity of the VPP quadcopter structure increases the cost and difficulty of manufacturing and maintenance of the quadcopter system, the benefits of rapid response control of torque and thrust are more valuable for recovering from disturbances. The good recovery ability of the VPP quadcopter indicates that the system has strong stability in the face of safety-critical tasks.

These advantages propose that the VPP quadcopter meets the targets of this project, which is to improve the safety, mobility and stability of exciting quadcopters. The hardness of designing a suitable controller for this complex actuator can be solved by using modern control algorithms. As a result, the VPP quadcopter is very suitable for this project.

However, this VPP actuator structure shown in Fig. 2.5(e) will cause problems such as the deformation of the motor shaft, large vertical space occupation and the slow responding speed. Therefore, a helicopter tail rotor structure is much better where the servo is connected to a mechanical device that slides on the motor shaft and the connection structure controls the propeller pitch angle, where the figures are shown in Fig. 2.6.

In addition, based on the structure of helicopter tail rotor, the VPP quadcopter can use a central motor to power all the propellers, which is called centralized-powered variable pitch propeller (CVPP) quadcopter [61]. Correspondingly, the VPP quadcopter with each propeller powered by an independent motor is called a separated-powered variable pitch propeller (SVPP) quadcopter. The comparison between SVPP and CVPP quadcopters is shown in Fig. 2.7. Because the CVPP quadcopter can use a gasoline engine to power the whole system, the short flight time caused by battery life is not a problem any more (the gasoline-engine CVPP quadcopter can fly for almost four hours according to [62]). Meanwhile, both CVPP ans SVPP quadcopters can achieve this flipping flight because their VPP structures are the same. However, compared to SVPP actuators, CVPP actuators can only change the pitch angle to control the thrust and torque which impair its



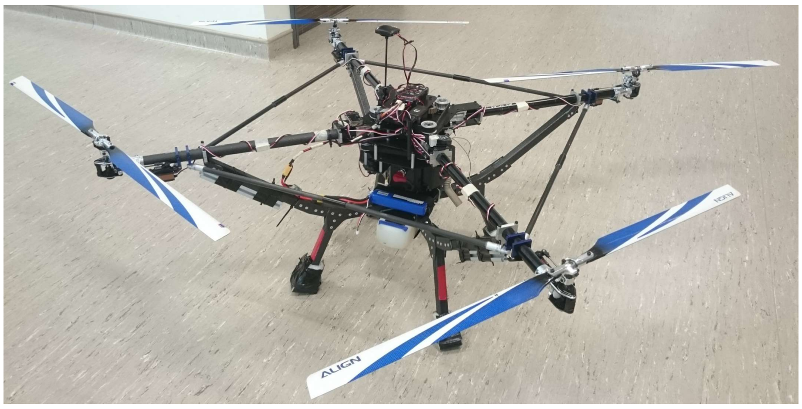
(a) (b) (c)

Figure 2.6. The VPP structure comparison between helicopter tail rotor and VPP quadcopter actuator where (a) is the VPP mechanism on VCTRC 450 helicopter tail rotor, reprinted from [60], (b) is the actuator mechanism of a CVPP quadcopter taken by the author, and (c) is the VPP mechanism on an SVPP quadcopter that applied from a fixed-wing aircraft, reprinted form [22].

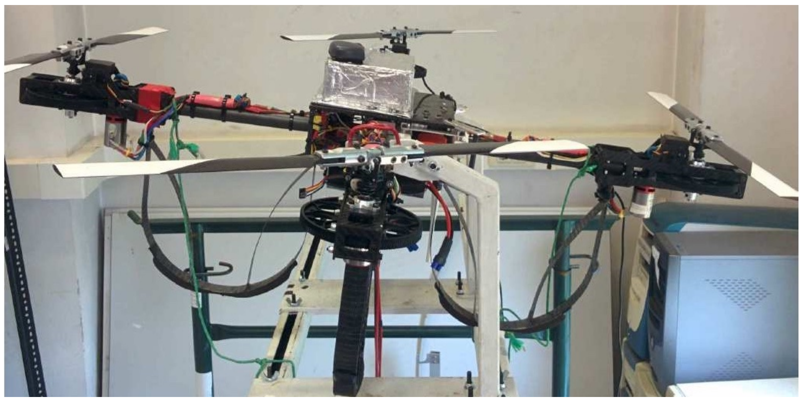
manoeuvrability [63]. In a nutshell, both two types of VPP quadcopters have their own merits, the specific choice of the VPP quadcopter depends on the needs of the application.

2.1.4 Actuator Control Comparison

There are three actuator configurations widely used in VTOLs including the main actuator for helicopter main rotor system, actuators for ordinary quadcopters, and actuators for VPP quadcopters.



(a)



(b)

Figure 2.7. The VPP quadcopter comparison based on the propeller-motor types where (a) shows a CVPP gasoline-engine quadcopter prototype, reprinted from [62] and (b) shows an SVPP quadcopter prototype, reprinted from [64].

The main rotor system of the helicopter is constructed by an active shaft and several propellers. The shaft is connected to the engine and it rotates at a fixed speed. The propellers are rigidly fixed to the main rotor hub (connected with the shaft) and the pitch of the propellers can be adjusted by a swash-plate. The swash-plate controls the pitch and roll of the helicopter main rotor by basically mechanically connection, while the pitch angle of the propellers of the main rotor will be able to achieve the varying angle criterion of the propellers. The swash-plate will tilt forward when the control command calls for the helicopter to fly forward, which means the propellers can tilt at a positive pitch angle at the rear and a negative angle at the front portion of the rotor disk to provide forward power. Therefore, the dynamical control of the helicopter highly depends on the control of propellers pitch angles. The control of the swash plate requires at least three servos and the autopilot design has to be established with the complex dynamical model. Although

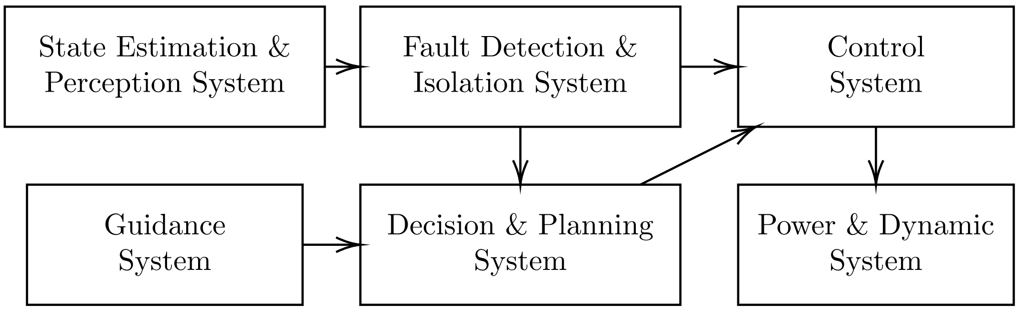

State Estimation & Perception System
Fault Detection & Isolation System
Control System
Guidance System
Decision & Planning System
Power & Dynamic System

Figure 2.8. The basic components of a typical quadcopter system.

the automatic control design of the helicopter actuator is very complex, this control design also provides the helicopter main actuator with great manoeuvrability.

The actuator structure of a common quadcopter is usually a direct connection between the propeller and the motor, and thus has the advantage of mechanical simplicity. However, since quadcopter dynamics control usually relies on the change of rotational speed to achieve, the manoeuvrability is poor compared to the other two types of actuators, especially for large scale aircraft.

VPP actuator is a hybrid structure that compromises the merits of helicopter and quadcopter actuator structures. It can be controlled by the rotation speed of the propeller and by its propeller pitch angle. Unlike the helicopter main actuator that can control the pitch angle of each propeller individually, the VPP actuators are simplified to control only the pitch angles of all propellers simultaneously. This not only greatly reduces the mechanical complexity of the actuator, but also preserves its manoeuvrability, such as the ability to rapidly achieve thrust changes and the ability to provide thrust in either positive or negative direction. Moreover, similar to ordinary quadcopters, VPP actuators can change their rotation speed to optimal control themselves to reach the desired dynamics.

In summary, the variable pitch control is a hybrid control includes blade pitch angle and rotating speed. It can achieve a good balance among the control complexity of mechanical, electronic and programming.

2.1.5 Summary

Overall, a quadcopter control system usually contains several subsystems includes a guidance system, state estimation and perception system, decision and planning system, control system, power and dynamic system and fault detection and isolation system. The relationship of the subsystems is shown in Fig. 2.8.

These subsystems can be divided into 3 parts, namely, perception, decision-making and dynamics. For the perception part, a global visual motion capture system (like Vicon) can provide accurate states feedback indoors. For outdoor environments, the multi-sensor fusion system can be used to combine IMU, Global Positioning System (GPS) and vision systems to obtain its attitude and position instead.

The decision-making subsystem is the brain of the quadcopter which designs the control signals for the dynamic system according to the tasks and the states feedback. The computation speed depends on the processor performance and the algorithm complexity. Since the computation delay will directly affect the control bandwidth of the system, the faster the decision-making subsystem calculates, the higher the control bandwidth is.

A good quadcopter dynamics should have a high control bandwidth and can thereby quickly follow the fast-changing command inputs. The higher the control bandwidth, the stronger the ability of the control system is to reproduce rapidly changing signals, and the shorter the response time of the dynamic system has. In addition, the stability of hovering and the range of the maximum thrusts are also very important.

2.2 Quadcopter Control Theories

2.2.1 Ordinary Flying Control Algorithms

The most basic algorithm is proportional integral derivative (PID) control, which is now widely used in practice [65]. The PID controller uses feedback error as the input to minimize the error value by using three parameters: proportional parameter (P), integral parameter (I), and derivative parameter (D). The equation can be explained as: proportional value multiplies the state difference now, the integral value used for the state error from the past and the future overshoot depending on derivative value. PID controller is the most basic way to control the quadcopter such as [66–68].

There are several ways to improve this basic controller, such as the fuzzy PID presents in [69] and fault-tolerant fuzzy gain-scheduled PID in [70]. In recent years, machine learning becomes popular and powerful such as masting the game of Go with only reinforcement learning (RL) [71]. After that, the idea of combining neural networks and PID comes out. In [72], the neural network assisted computationally simple PID is given.

Optimal control is a classic control law developed based on the works of Pontryagin's maximum (or minimum) principle and dynamic programming [73, 74]. The control strategy relies on the knowledge of the system dynamics or models [75]. The control target of the optimal control theory is to minimize the desired objective function over a certain period of time. One of the most commonly used optimal targets is to minimize the energy cost [76].

Moreover, when a quadcopter is hovering or near hovering (flying at a low speed with small pitch and roll angles), the dynamic model can be approximately linearised into a linear model which can be represented by a set of linear differential equations via the first-order Taylor expansion. By designing a quadratic objective function, the control system can be regarded as a linear quadratic Gaussian (LQG) problem. Therefore, an LQG controller that combines a Kalman filter and a linear quadratic regulator (LQR) can be designed. According to the separation principle, the Kalman filter and LQR controller can be designed separately. LQR is a typical optimal control method that was designed to solve to problem of finding an optimal action from multi possible options in a linear model [77]. The basic idea of this algorithm is to reduce the cost function of the state space equation by using the weight parameters provided by the user. There are many objective functions for different goals. Some of the researchers want the quadcopter to be highly energy-efficient, while other researchers want to achieve a balance between accuracy and efficiency [78].

However, it shows that there is not a huge difference in the effectiveness of the control system for a quadcopter platform when the same feedback environment is given to PID and LQR controllers [79]. All of them can have a good performance when applied to an indoor micro quadcopter [80], while the LQR controller has a better robustness performance in the comparison [81]. We use this algorithm in Chapter 4 of this thesis.

Normally, the linear control law is only valid near the equilibrium point [82]. To extend the applicability of the controller, the quadcopter system can be modelled as a linear parameters varying system. Then a linear parameter varying controller is designed to control the quadcopter attitude and validate in the practical environment [83].

Since the quadcopter system is nonlinear, the controllers designed based on linearised models usually only have good control performance near the equilibrium point [84]. Therefore, more algorithms studied based on nonlinear UAV dynamics have been proposed. Sliding mode control (SMC) is one of the most widely used nonlinear methods in quadcopter controller design. The basic idea of SMC is to generate several discontinuous control structures and switch among them based on the states. This algorithm has been validated on a Qball-X4 quadcopter [85].

Besides, many nonlinear methods can transform a nonlinear system into an equivalent linear system and then design a linear controller, such as input-output linearisation, full state linearisation [86], feedback linearisation [87] etc. For example, the full-state linearisation can be used to simplify the quadcopter model to control the system with the linear control method [88]. Feedback linearisation can be used for control strategy designing to handle the non-linear dynamic behaviour [89].

In addition to the control algorithm improvement, quadcopter dynamic modelling has been improved as well. Since Euler angles suffer from singularities, the quaternion is, therefore, a good substitution to represent the rotation of a rigid body. Quaternion based controllers are globally nonsingular and have been validated in simulated quadcopter dynamics [90].

Usually, a quadcopter task requires the aircraft to move from one position to another with constrained conditions. As a result, an entire quadcopter controller requires a path plan generator in addition to an onboard controller. Path plan is a method that can guide-way to the quadcopter [91] which is widely used in quadcopter control systems. The dynamic control methods investigated above can be used together with path planning algorithm [92]. It is a high-level controller of a quadcopter that generates a geometric path passing through some pre-defined key points and connected the initial to final position [93]. A good path generation algorithm can reduce the movement error and increase the efficiency [53].

2.2.2 Fault Tolerant Control

In addition to the normal flight control methods mentioned before, safety-critical quadcopter control is also important in the future applications. According to different damaged parts, there are many different types of system failures, including actuator fault (effectiveness), signal fault (uncertainty), sensors fault (no feedback) and plant fault (dynamics changed), etc. The actuator fault also has many different situations depending on the sources of fault (signals or motors), number of damaged motors and the fault level of them (loss of effectiveness). Depends on the varying degrees of destruction, the dynamic of the quadcopter will be affected, which will cause the controller designed for the original quadcopter dynamic to become invalidated. Such a situation will thereby endanger the quadcopter and the passenger. Since the quadcopter failure during operation is unavoidable, a controller design for such a consideration is required.

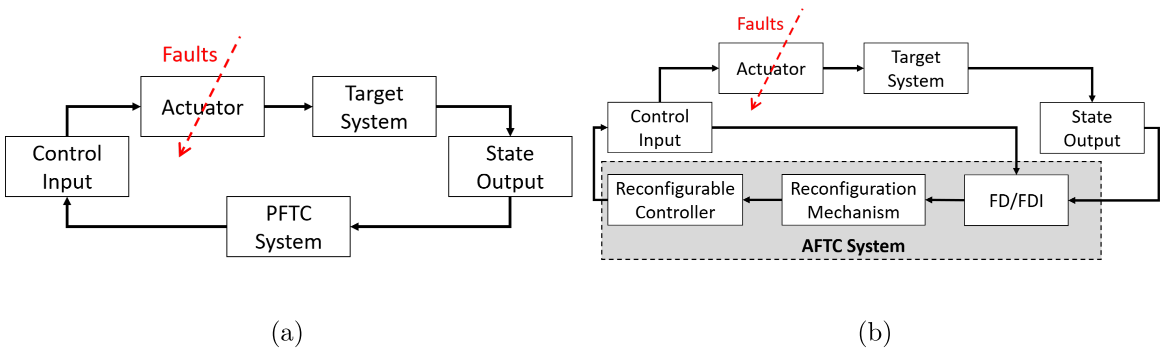


Figure 2.9. System structure comparison of FTC algorithms, where (a) is PFTC and (b) is AFTC.

Fault-tolerant control (FTC) is a method to solve the problem of the system fault. FTC is a set of methods that can increase the availability and safety of the quadcopter when faults happen and are especially necessary for future passage transportation and emergency rescue fields. There are two main types of algorithms in FTC systems, which are active FTC (AFTC) and passive FTC (PFTC) [94].

PFTC is normally considered as a robust controller that can accommodate actuator failures without receiving feedback from the sensor, shown in Fig. 2.9 (a).

PFTC is more direct to accommodate the fault and keep the system stable by treating faults as disturbances, which has been applied to quadcopters such as feedback linearisation approaches [95, 96]. However, the fault-tolerant capability of PFTC is limited, because the system fault is unknown, and the PFTC controller is the same. Different from that, AFTC can set up alternative controllers for different situations [97].

AFTC is a method of switching a normal controller to a prepared emergency controller when faults are detected, shown in Fig. 2.9 (b). Normally, an AFTC system includes not only the reconfigurable controller part but also the fault detection and isolation (FD / FDI) mechanism. Since the system knows what faults happen, corresponding control methods can be applied to the system [98, 99]. With the development of machine learning theory, a AFTC is designed through neuron network-based control method [100]. Due to the reason that an AFTC contains two parts (FDI and fault controller), it may delay the time of fault compensation and reduce the stability and safety of the system [101].

In addition to the overall AFTC design, some researchers focus on one part of AFTC, that is, the reconfigurable fault controller. Sun et al. implement a triple loop controller to control a quadcopter fly at high speed (9 m/s maximum) with loss of a single motor in reality [102]. Furthermore, some researchers considered more complex situations, such as two and three actuators failure. The algorithm is designed to control the malfunctioned quadcopter quickly spin around a designed axis and thereby holding its position [26]. It is noticeable that when the adjacent actuators are at fault, the quadcopter is considered as one actuator available situation. However, due to the quadcopter platform limitation, the faulty quadcopter gives up control for the attitude. This will cause the quadcopter to spin during flight. With the help of the VPP quadcopter platform, such a problem will be alleviated to some extent in this thesis.

Overall, FTC can reduce the risk of safety hazards of the malfunctioned aircraft and plays an important role in the safety-critical research. Since there are many types of VPP quadcopter failures, this project will focus on the most common circumstances, one actuator failure including the stuck of propeller and completely broken. Furthermore, it is also analysed that the loss of two adjacent actuators situation are based on the simplified model which makes the future development of the safety-critical quadcopter possible. As a result, we use FTC in Chapter 5 of this thesis.

2.2.3 Manoeuvrer Control Algorithms

In addition to controlling quadcopters near the hovering equilibrium, manoeuvring flight of quadcopters is also a hot research field.

Controller Design

For the first time, the research of quadcopter stall turn aerobatic manoeuvre flight was realized based on the STARMAC platform [47]. They find that the quadcopter has different aerodynamic effects when running at high speed and therefore a thrust compensation control is presented. The results show that the compensate algorithm can reduce the tracking error and is more conducive to performing manoeuvre flight. Later, the same research group presents an autonomous backflip manoeuvre [48]. They divide the whole action into three modes (impulse, drift and recovery) and analyse the mode switching algorithm based on different angle speed to ensure a safe transition between different modes. The quadcopter first reaches the desired states (enough backflip rotation speed) in the first mode. Then, it will turn off the rotors in the drift part and rotating by inertia to achieve a backflip manoeuvre. After the backflip, the recovery controller will bring the quadcopter back to the stable. The algorithm is demonstrated on STARMAC platform in a practical outdoor environment shown in Fig. 2.10 (a). However, this algorithm is workable but not robust nor intelligent. The performing result highly depends on the parameter settings and the quadcopter motors are turned off when the quadcopter enter the drift mode. All these shortages lead to successful but unsafe performance.

In addition, a research based on a planar quadcopter model to achieve a multi-flip manoeuvre within a 2D plane is proposed [49]. The basic idea of the algorithm is to design a five-step action with five parameters and then update the parameters via policy gradient adjustment. This multi flip motion can be implemented on a 3D practical quadcopter by applying iterative learning control, shown in Fig. 2.10 (b). Similar results show that the controller designed based on a 2D planar quadcopter can be applied to a 3D practical environment [103]. This algorithm can improve the control performance by learning from working in a repetitive mode. Moreover, a time-optimal control method is applied to study the shortest time trajectory between two states on a 2D plane [104]. All these studies imply that when studying a 3D motion but with 2D features, we can first simplify the 3D model into 2D model and then design a controller within a 2D plane. Finally, we can migrate the whole controller to a practical environment to achieve target performance.

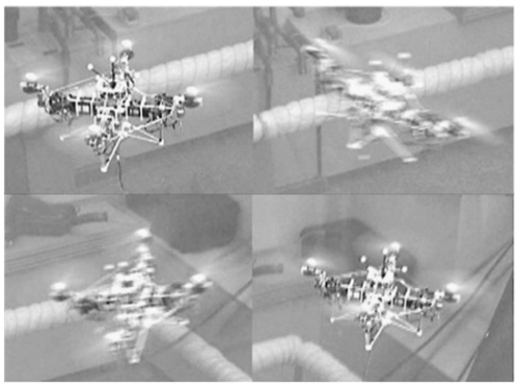
Like ordinary flight controllers, because the quaternion model solves the singularity of the Euler angle model, it is widely used in manoeuvrer flight control. However, quaternion model has its own problem, that is, one rotation can be represented by two different quaternions, which is called three-sphere double-covers 3D rotation. This may cause the unnecessarily large rotation of the quadcopter when quaternion based controllers are used [105]. Therefore, a control algorithm based on $SO(3)$ to control the attitude of the quadcopter is proposed [106].

To achieve a better control performance of the quadcopter, the geometric control is given out [107]. Since the quadcopter has 4 input DOFs, this algorithm designs a geometric controller that can control 3 position variables of the airframe and a body-fixed axis direction. This method changes the model of the quadcopter from a special orthogonal group in three dimensions ($SO(3)$) to a special Euclidean group in three dimensions ($SE(3)$) to achieve almost global controllable (initial attitude error is less than $\frac{\pi}{2}$). Geometric control uses a rotation matrix to define attitude tracking error and thereby design a controller. The results show that the quadcopter can perform complicated manoeuvres [108].

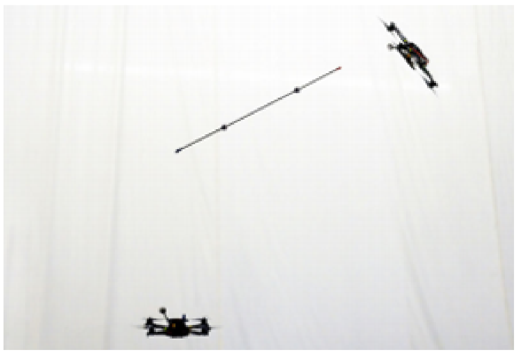
Trajectory Planning and Generation

In order to achieve high manoeuvring flight, a normal path planning algorithm is not good enough. Therefore, trajectory planning and generation algorithm is

(a)

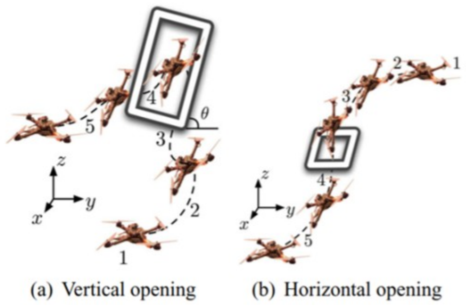


(b)



(c)

(d)



(e)

(f)

Figure 2.10. Examples of quadcopter projects that have good manoeuvrability. (a) shows the backflip manoeuvrer performed in an outdoor environment by a Stanford quadcopter, reprinted from [48], (b) shows the multi-flip manoeuvrer, reprinted from [103], (c) shows a quadcopter performing inverted pendulum and pole balance, reprinted from [31], (d) shows the inverted flip performed by an SVPP quadcopter, reprinted from [21], (e) shows the manoeuvrers that quadcopters passing through windows, reprinted from [29], and (f) shows the ball juggling by a customized quadcopter equipped with a racket, reprinted from [30].

considered, which contains path planning and time constraints [109]. Since the trajectory planning algorithms not only generate the kinematic plan but also the time plan at each keyframe, they can be used to optimise the trajectory by designing an objective function. The objective is to design according to different targets, such as minimum time, energy, jerk etc. There is also a method that can find a time-optimal trajectory to increase the path generating speed [110, 54]. Therefore, a reliable and reasonable planned trajectory can largely enhance the performance.

To have an accurate trajectory tracking performance, a precise sensing environment is needed. The ETH D'Andrea group uses a rapid and accurate quadcopter platform. The high-order system dynamics of the motor are considered to be ignored. Then, the rotational dynamics of the quadcopter can be regarded as a rapid response system. The angular acceleration and corresponding thrust can always follow the given command. Therefore, the researchers can design a controller that aims to control the angular velocity of the quadcopter, and directly plan the angular velocity trajectory of the aircraft. Relying on a precise positioning system, the control system could estimate the motion, the target impact position and the corresponding time. Then, the trajectory optimization system can plan a reasonable trajectory for the quadcopter to move from an initial state to the target state (contains pitch and roll angles, velocity towards the ball and position coordinate) at an estimated time. As a result, the presented control systems can allow the quadcopter to juggle a ball [30] (as shown in Fig. 2.10 (f)).

Besides, researchers form ETH also study how to balance an inverted pendulum on a quadcopter [31] (shown in Fig. 2.10 (c)). Optimal control and trajectory plan are used together to catch the throwing pole and then keep balance with it. In general, it is found that the trajectory plan is a very important research area that can lead to outstanding manoeuvre performance. The key of their research is to plan a reasonable real-time trajectory for the quadcopter to follow by using an accurate indoor vision capture system.

Based on the geometric control theory, researchers from Pennsylvania University design a controller and trajectory generation algorithm that can control a quadcopter to implement aggressive manoeuvre motions such as flying through windows and vertical perching [29], show in Fig. 2.10 (e). It is shown in [29] that, the controller and trajectory are designed separately, and a low latency experimental environment similar to Flying Machine Arena (FMA) is provided to reduce the error between the target trajectory and the actual trajectory. They design a full state tracking controller via differential flatness. A differential flatness system is a dynamic system that can use a group of outputs of the system and their derivatives to represent all the states and inputs. By choosing a group of reasonable output variables, the system dimensions can be effectively reduced. Furthermore, the motion plan of the states and the trajectory constraints can be mapped to the flat output space to plan the optimal trajectory and thereby mapped back to the original dynamics. This method reduces the calculation time of the trajectory optimization so that it can be implemented in real-time in practical applications.

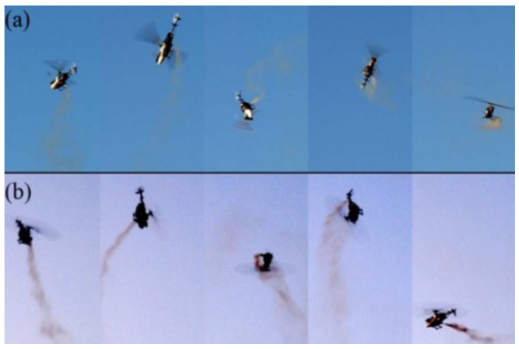
Based on this foundation, a trajectory optimization algorithm is designed which is called minimum snap trajectory generation. The basic idea is to find a trajectory that can minimize the objective function of snap (the fourth derivative of position and yaw states of each keyframe) and enable the quadcopter to fly through each keyframe smoothly. Moreover, by setting the target states of keyframes as constraints, the trajectory can guide the quadcopter to perform desired motion at certain keyframe positions.

As can be seen from the previous investigations that manoeuvring control is an important research field in quadcopter researching. This is mainly because that, with the need for aerial tasks increased, the requirements of quadcopters becomes higher. It requires the quadcopter to improve its performance in different aspects to increase employability. However, in most of these investigations, the quadcopter is uncontrollable during the manoeuvring. Other studies that use trajectory planning also have shortcomings. For example, the control system need to know the flying path in advance for planning instead of generating manoeuvrers in time. But in most practical situations where manoeuvring is actually required, such a path is not always available. Therefore, in order to overcome the shortcomings of the existing manoeuvring controllers, a robust controller that can perform controllable manoeuvre in time is needed. To the author's knowledge, a learning-based controller can use helicopters to achieve such a goal, thereby proposing a new direction for the current study.

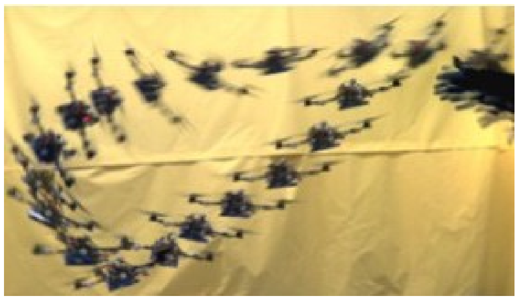
Learning Based Algorithms

Normally, a controller design needs to analyse the dynamic model of the quadcopter. An accurate quadcopter model is usually very complicated and thereby hard for controller design when considering gyroscopic effects and aerodynamic effects. Therefore, learning-based algorithms aim to enhance the capabilities of controllers designed based on simple models or generate controllers by themselves.

Iterative learning control (ILC) is a typical controller improvement method used in a repeatable task, such as a quadcopter follows a certain trajectory. For such a task, a controller always yields the same error each time. The objective of ILC is to help the original controller have a better performance by adding a correction



(a)



(b)

Figure 2.11. Examples of UAV projects that have good manoeuvrability or recoverability based on RL algorithms where (a) shows the manoeuvrers performed on a Stanford helicopter through inverse RL, reprinted from [114] and (b) shows the hand throw recovery implemented on a ETH quadcopter through RL, reprinted from [36].

signal to the controller output and update the signal repeatedly according to the error [111].

ILC has been widely used in nonlinear system controller designs such as enabling a quadcopter to perform the flip manoeuvre with a simple model [103]. Moreover, a quadcopter can learn to fly aggressively through the tunnel while avoiding obstacles via using ILC for navigation [112]. Unlike other machine learning controllers, the ILC plays the role of augmenting and tuning the control signal [113]. Since ILC does not have a lot of parameters to train, it can quickly converge.

Despite the control theory, machine learning can also implement UAV manoeuvres. A machine learning-based control system that enables the helicopter to teach itself to fly several manoeuvres (as shown in Fig. 2.11 (b)) is developed. It learns to perform the same actions from other pilot controlled helicopters [114]. Since helicopters and quadcopters have similar dynamic models, the control algorithm applied to helicopters can be regarded as the proof that machine learning has the ability to control the manoeuvring flight of quadcopters.

Furthermore, with the development of machine learning, a quadcopter has been trained to learn how to recover from random initial states (with random initial translational and rotational speed) [36], shown in Fig. 2.11 (b).

These machine learning-based controllers are designed by model-free algorithms which do not require an accurate dynamic model. Due to the self-leaning ability, the machine learning controller will have human-level control performance in the future. This algorithm is presented in Chapter 6 of this thesis.

2.2.4 Summary

In recent years, the control algorithm has developed rapidly, and the current quadcopter flying performance has made great progress compared with the one in the past. Relying on geometric control and trajectory planning, the quadcopter can now almost achieve smooth control in the full $SE(3)$ space, although this performance requires an indoor system that provides accurate perception and rapid control response. However, when using such a quadcopter in actual outdoor applications, we cannot provide the same experimental environment, a robust and rapid response control algorithm is needed to develop in the future. To achieve human-level performance, we regard machine learning as its future development direction because it can iteratively improve control performance during multiple flights. Moreover, with the development of machine learning, more attention has been paid to applying such an end to end method (directly connected the image input and the control output through a network) to quadcopter control problems.

2.3 Machine Learning Algorithms

Machine learning is a general method that tries to let the algorithm learn by itself from dataset. Because the algorithm has to learn from big data, it needs huge computing power to support it. In recent years, due to technological developments (improved data availability and increased computing power) in this field, significant progress has been made in machine learning capabilities like good robustness [115]. In a general way, machine learning can be divided into several parts by different situations as:

1) Supervised learning: This algorithm tries to extract and fit the relationship between different data with the same label. A well-labelled training set is very important for supervised learning. The solution is then validated in a testing data set and can be further improved after being deployed.

2) Unsupervised learning: This method aims to cluster and distinguished the data without being labelled. No labels are given to the training set for unsupervised learning. The algorithm perceives the relationship within the data set in an abstract way. Because there are no pre-defined labels, the solution can adapt to the data autonomously.

3) Reinforcement learning (RL): Compared with the above two data analysis algorithms, reinforcement learning is more like a controller. The training

data are created by the states feedback of the interaction between the agent controlled by the reinforcement learning algorithm and the environment. This algorithm encourages the agent to achieve higher values according to the pre-defined reward function.

It should be noticed that RL is more suitable to train a model to learn how to behave when the environment can give feedbacks accurately [32]. Consequently, this algorithm has been widely used in many different applications, such as, game theory [116], autopilot [117], robot control [118] and also quadcopter fly control [119].

In particular, the agent needs to observe the environment and its states, and then decides the action to achieve the desired goal. The agent is the active entity of the RL algorithm. After that, based on the environment system feedback, the agent will be rewarded or punished and therefore learn from the performance evaluation. In a nutshell, RL is a type of algorithm that can learn the mapping function where the input of the mapping function is environmental information, and the output is the action. The evaluation of mapping the function performance is a designed function based on the degree to which the goal is achieved.

Therefore, in this thesis, RL is more suitable than the others because controlling a quadcopter is a typical environmental interaction problem. Here we use RL algorithms within Chapter 6.

2.3.1 Markov Decision Process

The basic idea of the RL algorithm is to maximise the reward R with the agent acting in the environment, knowing the current state (observation) and the action [120].

This process can be abstracted into a Markov Reward Process (MDP) [121, 122] 4-tuple $<\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}>$ by three steps together with Bellman equations [123]:

1) The subsequent state value $s' \in \mathcal{S}$ is proposed by previous state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ while there is no influence with previous states, where $\mathcal{S}$ is the state set and $\mathcal{A}$ is the action set. There is a special state called the termination state, after reaching this state, there will no more subsequent states. Then collects all the states fo n steps $\{S_0, A_0, S_1, A_1, \ldots, S_n, A_n\}_i$ and reward values $\{R_1, R_2, \ldots R_n\}_i$ by using the same policy π in a serial episodes $i(i = 1 \ldots N)$. Here, lowercase letters indicate a specific state, action or reward, and capital letters with subscript t indicate any of which at specific time t.

2) The agent chooses the action a based on state s under the policy $\pi(a|s)$. The probability between states change can be described with value $p(s'|s,a)$, where $p \in \mathcal{P}$.

3) According to the action taken, the agent will get a new state s' and a reward value $r(s'|s,a)$. The discounted return after the time step t is given as

$$G_t = \sum_{t=0}^{n} \gamma^t R_{t+1},$$

where $\gamma \in [0,1]$ is the discount factor. In addition, according to Bellman equations, state-action value function under policy π is given by

$$Q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a], \tag{2.1}$$

and the state value function of s under policy π is given by

$$V_\pi(s) = E_\pi[G_t|S_t = s] = \sum_a \pi(a|s)Q_\pi(s,a).$$

Here expectation E_π indicates N episodes average value of $G(s)$ under a certain policy π. At the end, the agent aims to find a optimal policy π to maximize the discounted return.

If the states transition probability model in MDP tuple is known, the process is called model-based RL [124]. Otherwise, if the agent has to learn the states transition probability, the process is called model-free RL. The agent can predict changes in the next few steps before exploring the environment, and then plan based on the environment model to reduce the number of interactions with the environment. This method is suitable for the tasks where the transition of the state P is discrete such as AlphaZero[125].

However, in most situations, the agent acting in the environment can not know the probability and reward value of all the actions. If a RL algorithm is constructed without this information, it is called model-free RL. The agent can no longer knows the values of $p(s'|s,a)$ and $Q_\pi(s,a)$ but have to interact with environment and then get the feedback to predict these values.

2.3.2 Discrete Action Space

Monte Carlo Method is a typical algorithm in order to solve this kind of situation. This algorithm uses the average discounted return value G_t from multiple episodes of experience to estimate the state action value function $Q(s, a)$ instead of $V(s)$, which is equivalent to the true value when the episode is large.

Although Monte Carlo Method can reach a good performance for unknown environment exploration, it is still a low-efficiency algorithm because it has to update the policy π after each full episode. The methods like Monte Carlo tree search (MCTS) require high learning computation requirement [126] when facing continuous controlling problems.

In order to improve the efficiency, a temporal-difference (TD) algorithm can be applied. Instead of requesting a full episode reward, the TD algorithm uses one step reward r to update the policy [127].

A typical TD RL algorithm named Q-learning [128] is a method that use greedy action $\gamma \max_{a'} Q(s', a')$ to update the policy by

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)], \tag{2.2}$$

with α represents the learning rate. Because it use maximum action instead of old policy, this makes it become an off-policy method [129].

Different from that, another TD RL algorithm named SARSA [130] which uses 5 sequenced values (state s, action a, reward r, estimate state s' and, estimate action a') to update the policy by

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)].$$

SARSA is an on-policy algorithm because it only uses the original policy to calculate $Q(s', a')$ to update $Q(s, a)$. In [131], a helicopter trained based on SARSA is able to give an inverted flying performance.

When using relevant data to train the reinforcement learning neuron network, the result performance of the training is unstable. Inspired by the biological research, an experience replay program can enhance the learning performance [132]. The deep Q-learning network (DQN) algorithm uses a reply buffer (replay memory) to store the state and action information to break the time sequence. Every training time, several random experience vectors (s, a, r, s') are picked from memory (named batch) as the training set. Moreover, DQN creates a mirror NN Q' to temporarily hold the NN updating. It separates the updating speed of running action-value function Q and target action-value function Q' to prevent overfitting. The target action-value function Q' will be updated after certain time rather than every trail. To enhance the training stability, double DQN train double Q NNs and select the less one to reduce the overestimation to update the TD-error [133].

The DQN can achieve a good learning result in the Atari games [134]. In the years 2016 and 2017, Google AlphaGo learned the Go game by reinforcement learning methods and won the world's rank number one player Lee Sedol in the Go match [135]. Research shows that the AlphaGo uses MCTS and DQN to achieve human-level gaming performance [136]. However, these algorithms can only choose a discrete output from the existing action space. Since the real control problems are often high-dimensional, it is difficult to implement these algorithms in practical control applications.

2.3.3 Continuous Action Space

Policy Gradient

To solve the problems shown above, that control output usually has to be a continuous, another type of RL called policy based RL method is investigated. Policy gradient (PG) methods select output action according to a policy function rather than a value. Therefore, instead of using the action value function $Q(s, a)$ to determine the action, PG uses a probability policy function $\pi_{\boldsymbol{\theta}}(s, a) = P_r\{a|s, \boldsymbol{\theta}\}$ to indicates the probability of selected action under states s and policy $\boldsymbol{\theta}$.

Similar to (2.1), a long term expectation value is proposed by

$$J(\boldsymbol{\theta}) = E_{\pi_{\boldsymbol{\theta}}}\left[r(s, a)\right],$$

which is expected to be high at the end [137]. In order to get the maximum value of $J(\theta)$, the gradient decent of $J(\theta)$ is provided by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_{i,t}|s_{i,t})\sum_{t=0}^{T}r(s_{i,t}, a_{i,t})\right],$$

where N is the number of trajectories and T is the length of each trajectory. And therefore, update the policy $\boldsymbol{\theta}$ by

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\nabla J(\boldsymbol{\theta}). \tag{2.3}$$

In a nutshell, PG aims to increase the probability of positive rewards and reduce the probability of negative rewards.

However, the agent can not choose all the available actions during the training. Some of the state-action trajectories can not give accurate exception value feedback. It is hard for the algorithm to search all the actions within the continuous action space.

To solve these problems, therefore, actor-critic algorithm is released [138]. It is an algorithm that uses a policy gradient NN as an actor to act in the environment and a value-based policy NN to evaluate the actor performance which is called critic, where the sketch is shown in Fig. 2.12(a).

Because the critic is an off-policy training NN, the actor-critic algorithm is able to have many agents work at the same time and update the policy reward at the end of the episode. The asynchronous advantage actor-critic (A3C) comes out in [139] to implement the idea. This is a great improvement because it can greatly shorten the learning time and improve the stability of reinforcement learning. This algorithm enables several actors to work independently and update the results asynchronously to increase the batch size and also cut the relevance of the data within the batch A3C can use multi-core working together which means the samples are low correlated, shown in Fig. 2.12(b). In addition to this, the asynchronous theory uses one multi-core CPU on the same machine to reduce the transfer cost from multi-PCs cooperation in [140].

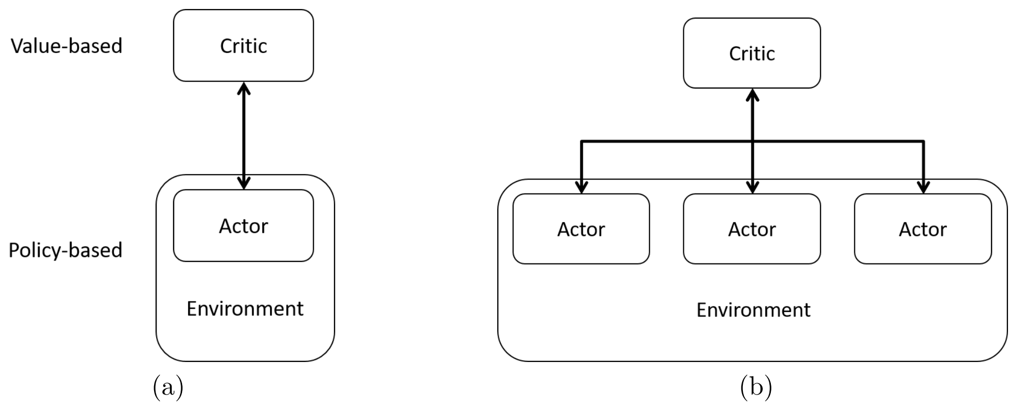


Figure 2.12. Examples of Actor-Critic serial methods, where (a) shows the single Actor-Critic structure (AC) and (b) shows the distributed Actor-Critic structure (A3C) used for parallel computation.

Deterministic Policy Gradient

Inspired by the DQN and PG methods, the deterministic policy gradient (DPG) uses deterministic value rather than the stochastic value used by PG to choose the action, which has the same function also proved by [141]. This algorithm increases the speed of action sampling if the action has a high dimension. More than this, researchers prove that the model-free deterministic strategy through mathematical derivation is exist [141]. After that, the deep neural network is applied to train both the actor and the critic network by Deepmind called deep deterministic policy gradient (DDPG) [33]. This algorithm is a widely used baseline for reinforcement learning. It uses a separate target network to create two NNs (a target network and an estimated network) in both actor and critic structures. It also uses soft target update and batch normalization to further improve the performance. In Chapter 6, DDPG was chosen as the target algorithm because it has a good balance between program complexity and effectiveness.

Because the critic NN in DDPG uses the same value-based NN compared with DQN, inspired by double DQN, a similar method can be applied to DDPG as well. Therefore, a method named Twin Delay DDPG (TD3) uses double critic NNs to decide the update value [142]. However, these deterministic policies has to design a noise function with hyper-parameters to explore the environment which makes this algorithm training slowly.

Stochastic Policy Gradient

Therefore, a new stochastic policy gradient algorithm called trust region policy optimization (TRPO) comes out [143]. It uses the trust region to restrict the policy update by adding KL divergence, so as to limit the differences between the new and old policies and prevent the updates from being too divergent. However, the calculation of trust region is complicated. Some researchers try to achieve the same target of limiting the update step size through some simple tricks, such as using a clip function, which is called proximal policy optimization (PPO-clip) [144].

This algorithm improves the data efficiency by ensuring that control training is maintained within a certain KL divergence change. Almost at the same time, the distributed proximal policy optimization (DDPO) has been put forward in [145] based on the PPO and A3C. DPPO uses distributed computing learned from A3C to speed up the learning process of PPO which makes it become possible to use parallel computation for accelerated calculation.

Researchers implement a complete reinforcement learning-based quadcopter controller that can stabilize the quadcopter from harsh initial state [36], which indicates that the RL controller has strong robustness. They use the method based on DPPO and enhanced the ability of exploration. The controller here is self-learning and model-free. The controller learns how to keep stabilized within the simulation training environment. After that, it is used to control a quadcopter in reality. Therefore the successful quadcopter control performance can give a good reference of VPP quadcopter controlling.

Other Related RL Algorithms

Due to the existing shortcomings of reinforcement learning methods, some researchers have adopted the principles of reinforcement learning algorithms and combined with other methods to develop learning based algorithms to achieve their control targets, for example, inverse RL.

Typically, reinforcement learning agent explores the training environment under the guidance of the reward function which is designed to reinforce the good action of the agent (normally the larger the better) and punish its bad action. However, there are some situations in reality where the reward function is unknown or difficult to design, but the desired action can be shown. Inverse reinforcement learning is developed to solve this problem. It has been shown that this algorithm enables the helicopter to perform complex manoeuvres learnt from pilot [114]. This algorithm first learns the reward function from the recorded manoeuvre flight experience and then use the learned reward function to teach the helicopter to perform manoeuvres. However, this method requires an expert helicopter pilot to perform this manoeuvre action which also limits its use.

Unlike the RL that learns to be a controller, learning based algorithms can be used to enhance the performance of the original controller as well. Due to the gap between the reality and the control design, there will always be some errors and differences in the practical use of the controller. It is useful to add a learning based algorithm to the controller to correct these errors if such a control process is required to be repeatable. Iterative learning control (ILC) is a system feedback controller used in repetitive operations. In the loop, the system will perform the same action again in order to improve tracking accuracy for the reason that the learning process uses the previous tracking error to improve the control signal by subsequent iterations [111]. The theory is feasible in a quadcopter 2D model such as [49]. The example gives a triple-flip action to prove the improvement of agility. Since the RL controller itself is also designed based on simulation, there is such a simulation-to-reality gap for actual implementation. This method can help reduce the gap and help RL controller to be implemented in the real world.

2.3.4 Summary

Compared to the classic control laws, the reinforcement learning-based controllers have several advantages: (i) less dependence on the controlled object model; (ii) less controller parameter tuning difficulty; and (iii) self-improvement capability. Therefore, it is undoubtedly a cutting-edge technology that has the potential to be implemented in the real world. However, this method still suffers from several shortages [146, 147]: (i) reward functions are often difficult to design, for example, a network learns cheating rather than expected behaviour; (ii) the network architecture and other hyper-parameters affect the performance; (iii) low migration ability, a trained network can only be applied on one type of mission; and (iv) low sample efficiency, long training time, high hardware performance requirements, for example, training a RL network for Atari games requires hundreds of hours of play experience while AlphaGo Zero requires 40 days of training to learn to play the game of Go from scratch. Although RL has these shortcomings, its novel learning characteristics are still worth looking forward to. From the relationship and development of RL methods, shown in Fig. 2.13, it can be found that RL is moving in the direction of continuous action space which is more close to reality. Moreover, RL is developing towards the easy building, quick and stable convergence and better network migration capability. Based on self-learning ability, reinforcement learning has great potential to improve its own control performance after being deployed. Other algorithms, such as ILC, can be adapted to help the RL controller overcome these shortcomings and become robust. Overall, the use of reinforcement learning algorithms in control problems is both a challenge and an opportunity.

2.4 Summary

In this chapter, we reviewed the development history and existing control methods for quadcopters. Although researchers have tried different methods to improve the control performance of conventional fixed-pitch quadcopters, the limit of the flight performance could not breakthrough due to the intrinsic limitation of fixed-pitch quadcopters. Therefore, we will study VPP quadcopters because it has high efficiency under various flight conditions.

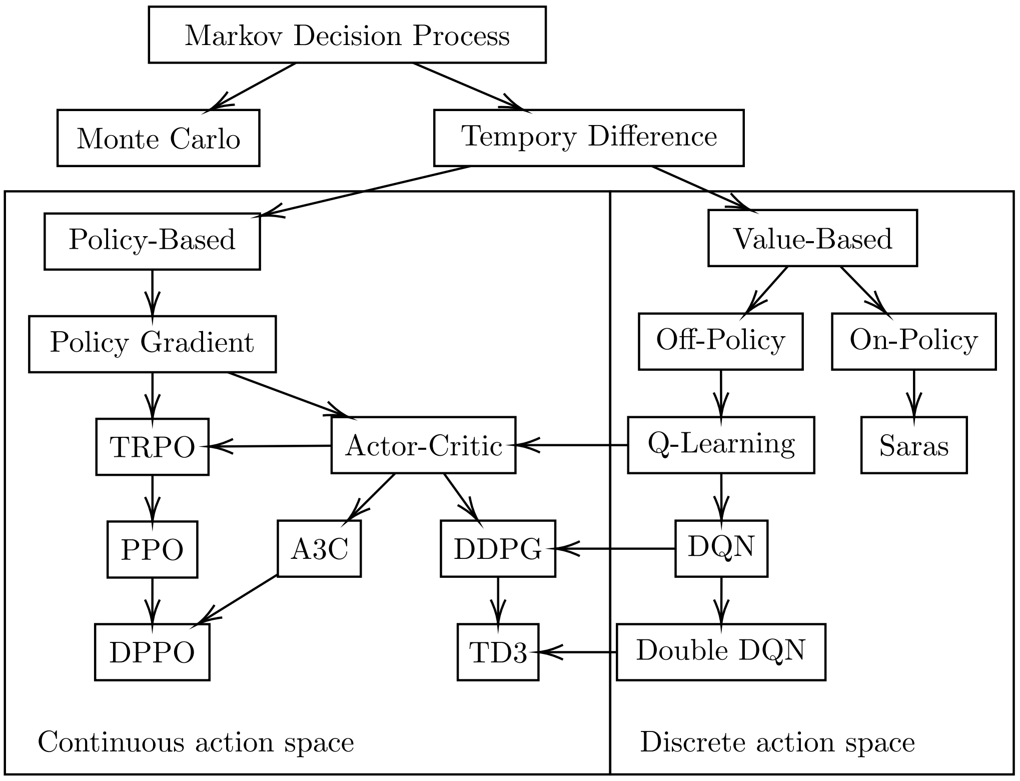


Figure 2.13. The figure of RL algorithms development.

Chapter 3

Model Analysis and Validation

3.1 Introduction

In this chapter, we will focus on the dynamics of VPP quadcopters. This chapter first introduces the model of VPP actuators and validates its dynamical equations by experiments. We construct a VPP actuator testbed that could be used to verify the dynamic equations and parameters. The difference between two different types of VPP quadcopters is discussed as well. Finally, the dynamical models of VPP quadcopters are presented for both 3D and 2D cases.

3.2 Actuator Model

This section compares the dynamics of a conventional fixed pitch propeller and a VPP.

3.2.1 Fixed Pitch Propeller

An ordinary quadcopter actuator contains a fixed-pitch propeller and a brushless motor. The propeller usually has two or three blades, depending on the thrust demand. The propeller can provide more thrust under the same rotation speed with more blades but will cause extra power consumption and noise. The propeller rotates in the air mainly to produce thrust, which is parallel to the rotation axis. The torque is the aerodynamic drag caused by the propeller rotation, which is in the opposite direction to the rotation.

Assume the propeller is rotationally symmetric, the surrounding air around the quadcopter is stationary relative, the motor runs near the rated speed, and the thrust generated by the actuator has a linear relationship with motor rotational rate. Normally a quadcopter actuator model can be describe as following dynamic model

$$T = C_L \omega^2, \tag{3.1}$$

$$M = C_M \omega^2 + I_M \dot{\omega}, \tag{3.2}$$

where T represents the thrust provide by an actuator, M is the torque, ω is the motor speed, C_M is the drag coefficient, thrust coefficient C_L and I_M are inertia moments of the rotor [148]. Most of the researchers ignore the effect of $\dot{\omega}$ as the value is small [149]. Using a simple model of the motor [150, 151], the actuator thrust and torque Eq. (3.1) can be simplified as

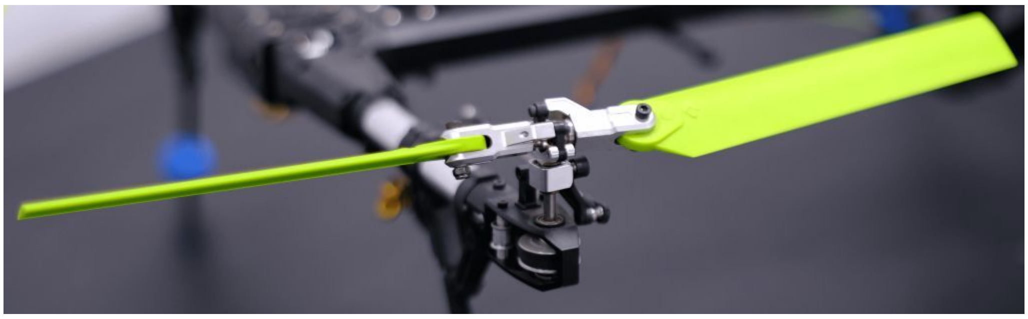
$$\begin{aligned} T &= C_L \omega^2, \\ M &= C_M \omega^2, \end{aligned}$$

which is only an approximation for the fixed pitch propeller actuator.

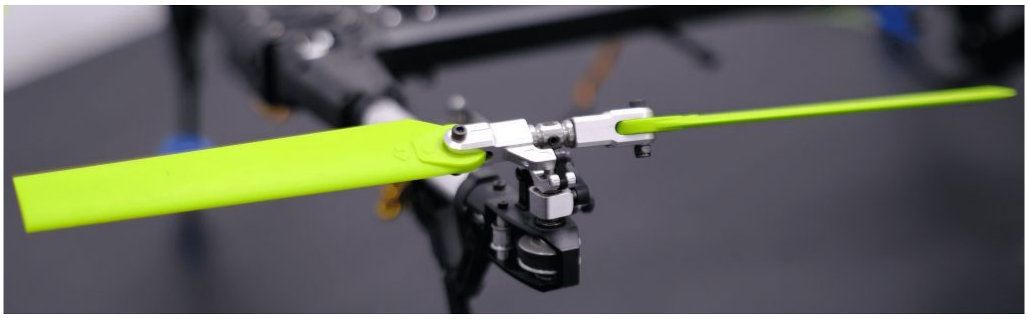
Overall, as a kind of aerodynamic force, torque and thrust can be considered to be proportional to the quadratic in the propeller angular velocity, where the coefficients are determined by the propeller and environment properties.

3.2.2 Variable Pitch Propeller

The VPP mechanism is adapted from the tail rotor of the helicopter. A servo can change the VPP pitch angle by pulling or pushing the connection, thereby generating forces in either positive or negative directions, where Fig. 3.1 shows the two force generation status of the VPP. Compared to ordinary quadcopter propellers,



(a)



(b)

Figure 3.1. A VPP actuator can use spinning speed and propeller pitch angle to change the required force and torque. A counter-clockwise rotating VPP generates a positive force with a positive propeller angle (a) and a negative force with a negative propeller angle (b). Both pictures were taken by the author.

this mechanism provides negative thrust, thus enhancing the agility and safety of quadcopters.

Figure 3.2 illustrates the schematic structure of a VPP, where the rotating speed is given by ω_i and pitch angle is α_i. Here i indicates the corresponding order of the VPP. Assuming that the angle control can be reached instantly, the rotation speed is relatively stable, and the pitch angle and rated speed can be controlled separately. According to [152, 153], when considering the propeller pitch angle α, the overall thrust and torque equations can be rewritten as

$$\begin{aligned} T &= K_0\omega^2\alpha, \\ M &= K_1\omega^2 + K_2\alpha^2\omega^2 + K_3\omega\alpha, \end{aligned} \tag{3.3}$$

where K_0, K_1, K_2 and K_3 are lift and drag coefficient parameters related to the practical aerodynamic properties of the VPP and the resistance of the motor.

The thrust generated by a propeller is generally determined in a number of ways, including airfoil profile, blade length, number of blades, rotating speed and pitch angle. For a ordinary quadcopter actuator, most of these parameters are already determined at design time, and only the rotation speed can be used for control.

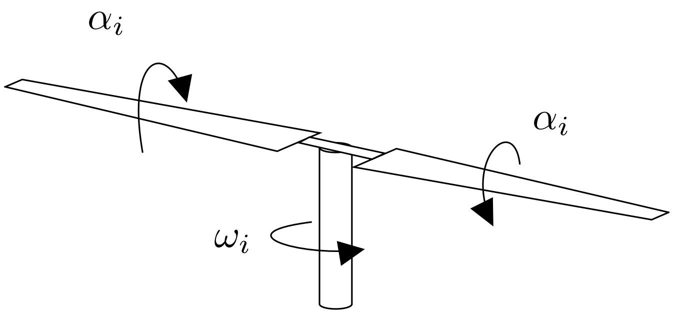


Figure 3.2. The schematic structure diagram of a counter-clockwise rotating VPP.

In addition to changing the rated speed of the motor like a fixed pitch propeller actuator, we can also modify the pitch angle of the VPP to adjust the thrust and torque of the actuator. However, any change of the propeller pitch angle impacts the motor dynamic and therefore affect thrust output. Still, this negative impact is so small that the propeller pitch angle adjustment can quickly compensate for it.

3.2.3 Experiment Validation

To further analyse the VPP actuator performance and evaluate the simulation equations from Section 3.2, we build a VPP testing hardware platform based on RCbenchmark Series 1585 Test Stand, as shown in Fig. 3.3. The main VPP structure is derived from a GARTT 450 helicopter tail rotor, which only retains the blade drive and connecting parts. A SUNNYSKY x2305 brushless motor and a pair of 65 mm NACA 0012 propellers are used to rebuild the VPP mechanism. A DEKO HV1295 digital servo is applied at the side to control the propeller pitch angle. The servo and the brushless motor of the VPP actuator can be controlled independently. Other connecting parts are designed according to the actual situation, and then 3D printing technology is used to complete the production. All 3D printed mechanical design parts are detailed in Appendix A.

The experiment platform can directly measure the torque (Nm), thrust (N), rotation speed (RPM) and also power (W) of the actuator. The specifications of the corresponding testing hardware platform are shown in Table 3.1.

Here, the accuracy of thrust and torque is given according to parameters of the thrust sensor. Although the accuracy of the thrust sensor is generally higher than this number, the accuracy will be reduced due to the effects of vibration during the operation of the test stand. Because the blade pitch angle is the same as the servo angle, we can define it based on the servo accuracy, which is 0.01 rad in our

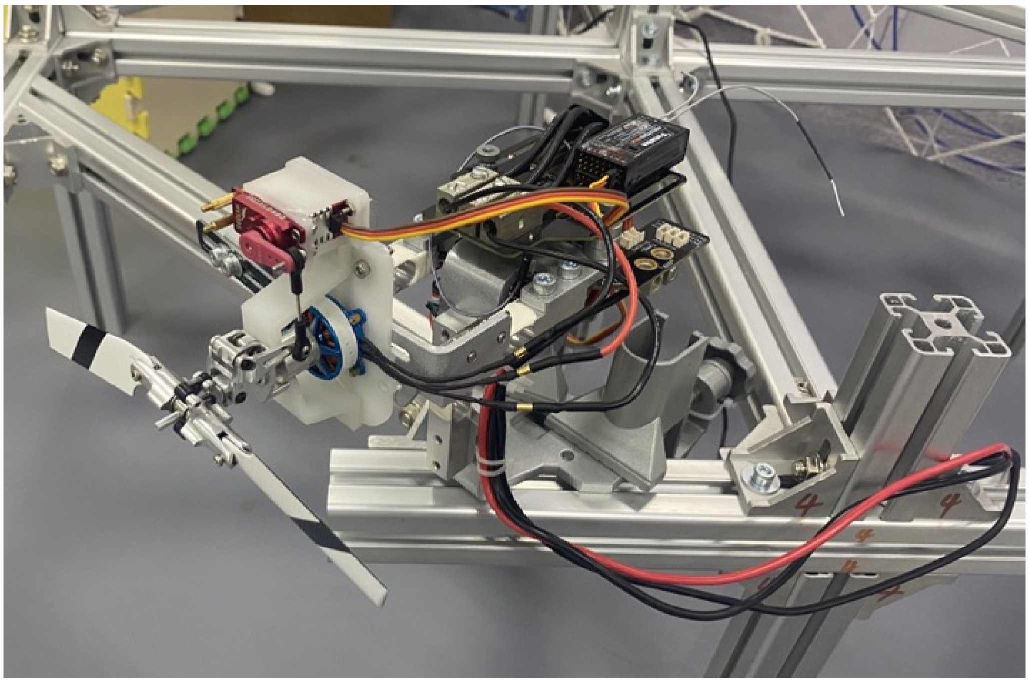


Figure 3.3. The VPP actuator testing platform developed by the author.

Table 3.1. VPP actuator test platform parameters.

Specification	Min	Max	Accuracy	Unit
Thrust	-49	49	0.01	N
Torque	-1.5	1.5	0.001	Nm
Power	0	250	2.5	W
Rotation Speed	0	30k	300	RPM
Pitch Angle	-0.3	0.3	0.01	rad

experiment. Therefore, although the range limitation of thrust and pitch angle are different, their accuracy is similar.

The power sensor accuracy is determined by the average deviation over the entire detection range of the sensor, which is 2.5 W. The rotation speed range is tested with a motor photoelectric sensor and the average accuracy is 300 RPM. However, the accuracy of the photoelectric sensor is much higher when the target is rotating at low speed than when it is rotating at high speed. It has better accuracy when the target is moving in low speed. In our case, the maximum rotational speed is 11000 RPM, which is much smaller than the upper limit of the range the sensor can measure into, so it can give a more accurate result.

According to the experiment preparation, the propellers used on the variable-pitch actuator are symmetric, tapered, 65 mm diameter blades. A NACA 0012 airfoil is chosen to model the propeller. Assume the propeller is rotating at 600 rad/s which gives an operating Reynolds number of around 50,000. Using XFOIL [154] and QPROP [155], the coefficients used in Eq. (3.3) can be estimated and are listed in Table 3.2 for reference.

Table 3.2. Estimated VPP aerodynamic coefficients

K_0	K_1	K_2	K_3
$4.65 \cdot 10^{-6}$	$5.36 \cdot 10^{-9}$	$2.58 \cdot 10^{-7}$	$2.52 \cdot 10^{-6}$

We set the maximum adjustable angle range of the propeller to 0.35 rad, which is smaller than the stall angle by a safe margin. The stall angle of the variable pitch propeller depends on the size of the propeller and also the running velocity. As the size of the propeller and the speed increases, the adjustment range of the variable pitch angle will become larger [62].

Therefore, we use the constructed experiment platform to test the thrust, torque and also overall efficiency of the VPP actuator. The thrust and torque generated at different motor speeds and propeller pitch angles according to requirements are tested, where details are shown in Appendix A.

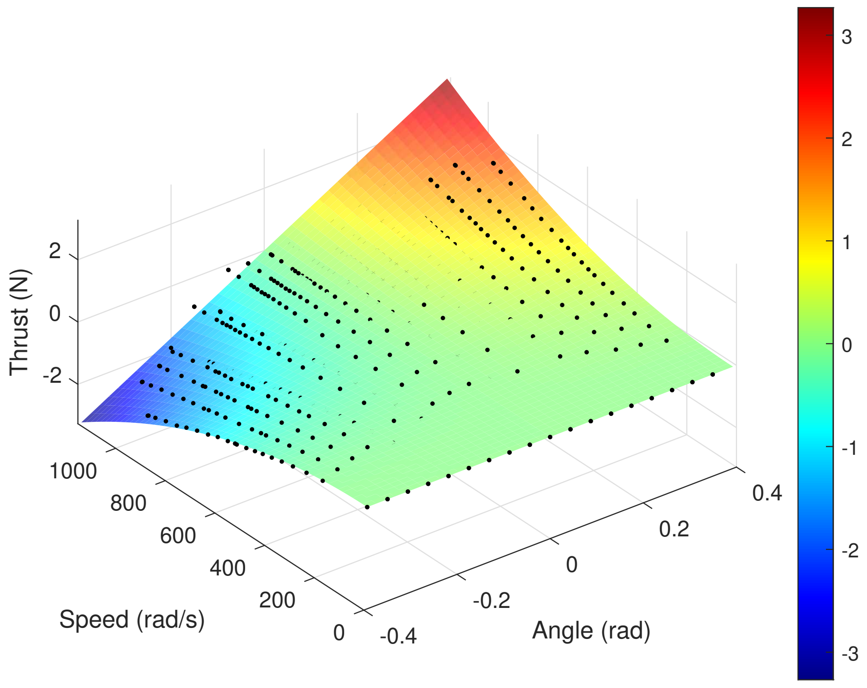
The thrust comparison and error between simulation estimation and experiment validation are shown in Figs. 3.4 (a) and 3.4 (b), correspondingly. The torque comparison and error between the simulation estimation and the experiment are shown in Figs. 3.5 (a) and 3.5 (b), respectively.

Black dots are the points collected from the experiment and the colourful surface is calculated by applying the parameters from Table. 3.2 to Eq. (3.3).

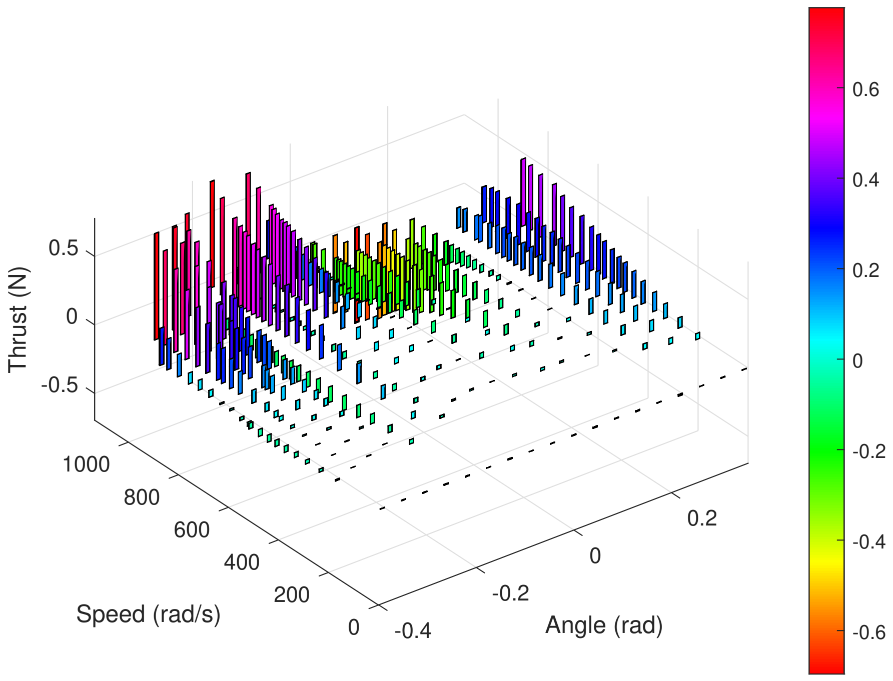
The result shows that the torque equation can fit the experiment very well, but the force equation has some errors when the detected forces are large. The difference may be caused by problems such as sensor accuracy, instrument vibration, measurement error, propeller stall, and inaccuracy of the simulation model. The symmetrical force and torque diagrams also show that the experimental results are consistent with the predictions, verifying that the symmetrical propeller has similar dynamic performance through positive and negative propeller pitch angles.

It is notable that, under our experiment environment, the torque is mainly affected by the motor rotation speed, and the thrust can be adjusted by the motor rotation speed and the propeller pitch angle. With a high rotation speed, the VPP can quickly change its thrust and torque but with a large energy cost.

Moreover, we analyse the VPP efficiency by $E = T/P$, where P is the power of the motor and T is the thrust. The efficiency graph is shown in Fig. 3.6 which shows that when the pitch angle of the VPP actuator is within the adjustable range, the



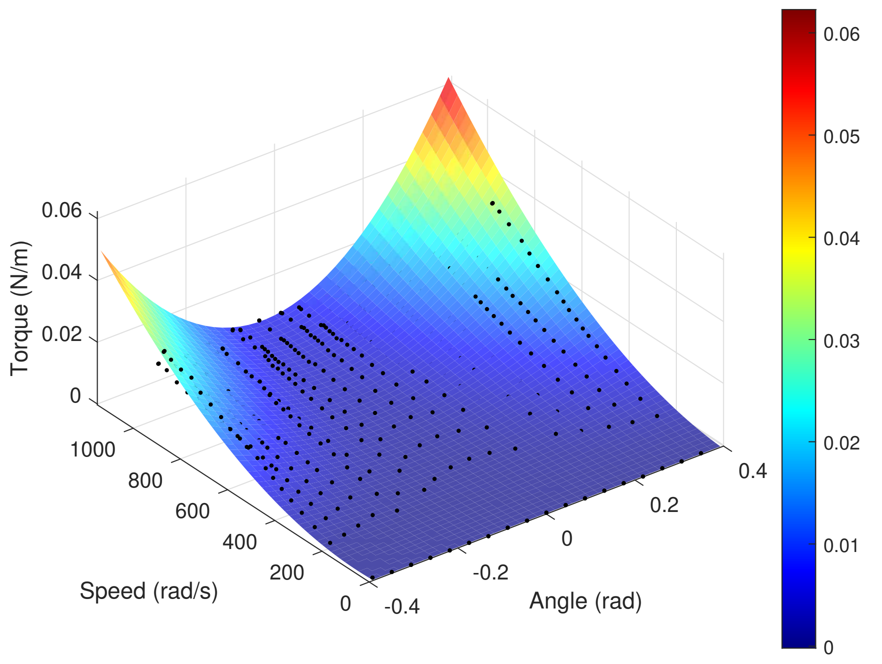
(a)



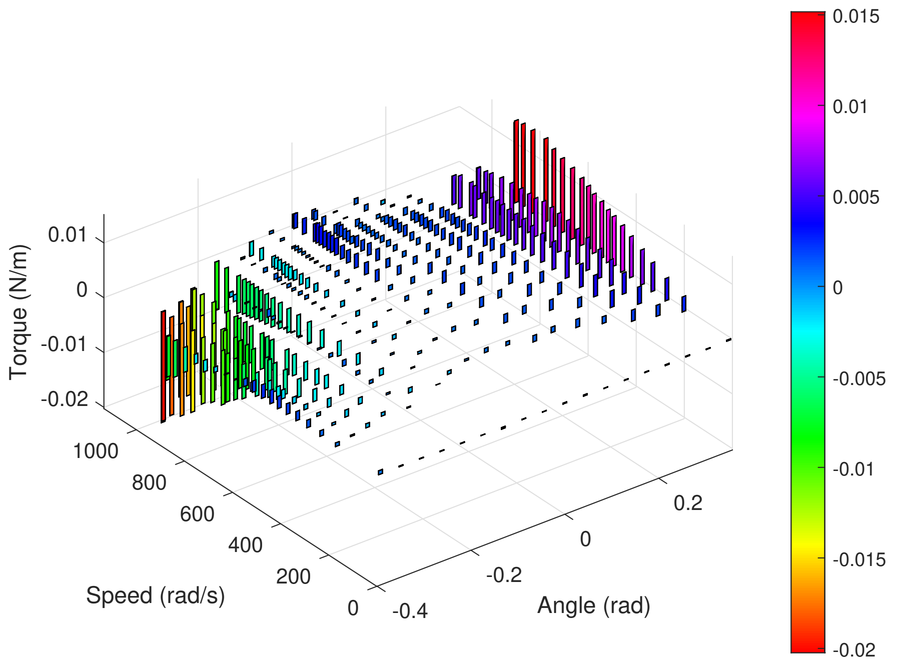
(b)

Figure 3.4. The comparison for the thrust of a VPP actuator between simulation and experimental validation, where (a) is the comparison and (b) is the error between them, with 20.4 sum squared error.

larger the angle and the smaller the motor rotation speed, the higher the efficiency.



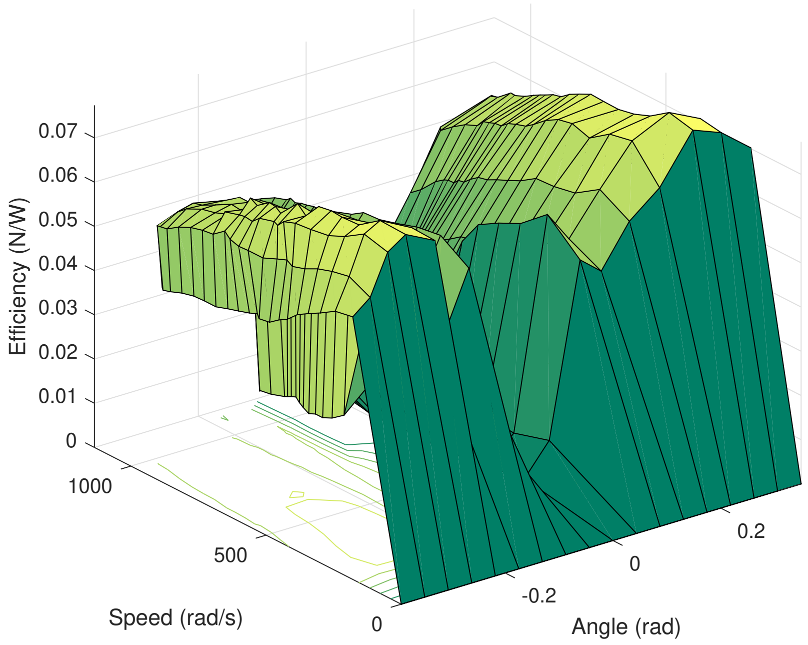
(a)



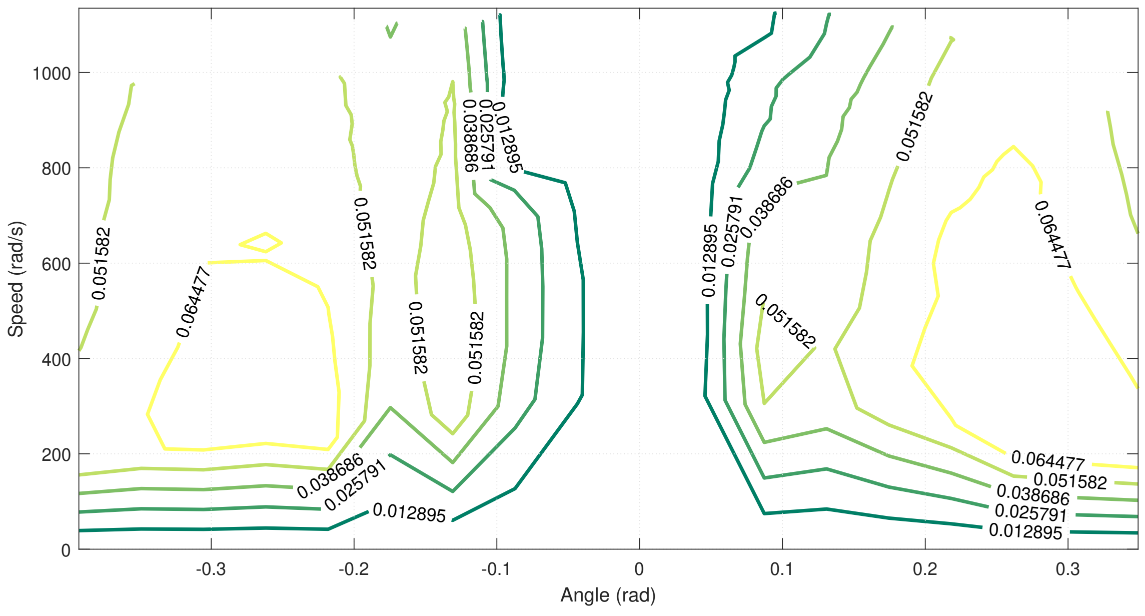
(b)

Figure 3.5. The comparison for the torque of a VPP actuator between simulation and experimental validation where (a) is the comparison and (b) is the error between them, with 0.0036 sum squared error.

The graph also shows that the stall propeller pitch angle of the actuator is a function of the motor speed.



(a)



(b)

Figure 3.6. The VPP actuator efficiency evaluation based on the experiment validation where (a) shows the overall 3D view and (b) shows the detailed work efficiency of a VPP actuator.

The relation among power, efficiency and thrust is shown in Fig. 3.7. It is notable that when the VPP actuator generates the same thrust, different combinations of rotation speed and propeller angle can result in different efficiency. Generally speaking, within the rated range, the larger the propeller angle, the higher the

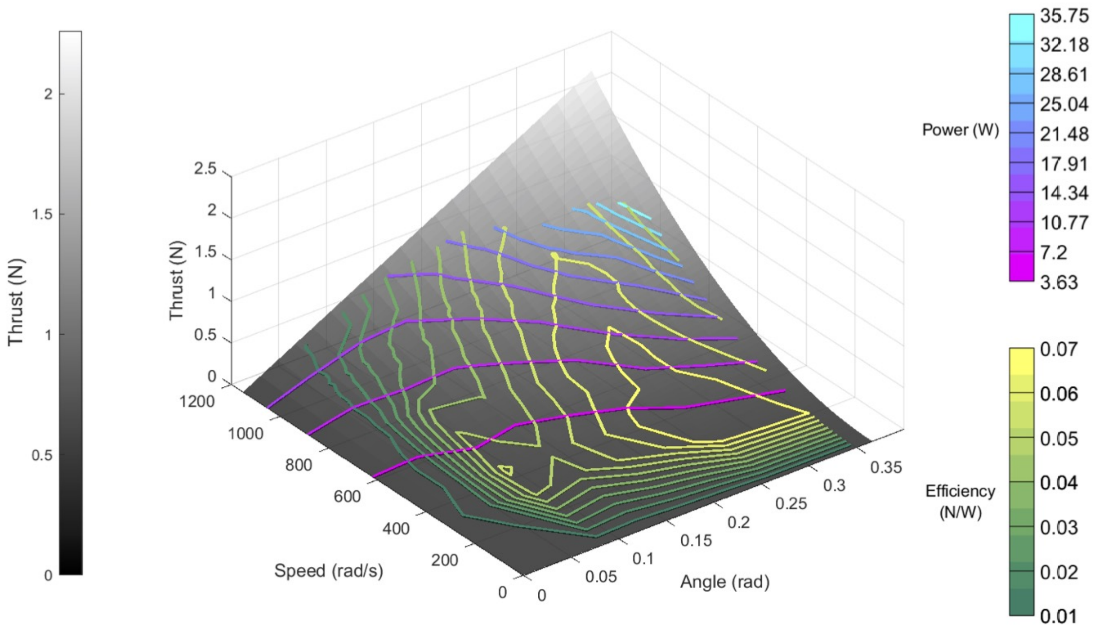


Figure 3.7. The surface (from grey to white) of thrust and lines of efficiency (curves, from green to yellow) and power (horizontal lines, from purple to blue), as a function of propeller pitch angle and rotation speed.

efficiency, the higher the motor speed, and the greater the energy consumption. Besides, it shows that the best work efficiency pitch angle of the propeller is around 0.28 rad and the best work efficiency motor speed is around 400 rad/s.

During the experiment, the vibration generated by the actuator becomes larger as the motor speed increases, shown in Fig. 3.8, especially when the propeller pitch angle is small. The excessive vibration generated by the actuator makes the actuator become unsafe to use. Due to the reason that the CVPP quadcopter usually maintains a rated propeller rotation speed within the flight, in order to ensure safety, the rated speed has to be kept always within the working area, which may lead to efficiency loss and manoeuvrability reduction. But this will not be a big problem for the SVPP quadcopter. Because the SVPP quadcopter can independently change the motor speed and propeller pitch angle of each actuator to maximize its efficiency and manoeuvrability.

Overall, the experiment validates that Eq. (3.3) can be used to describe the dynamic of the VPP actuator when both propeller pitch angle and motor rotation speed are within the working area. Moreover, the torque and thrust equations indicate that different combinations of speed and propeller angle can be used to make the VPP run at the same target thrust.

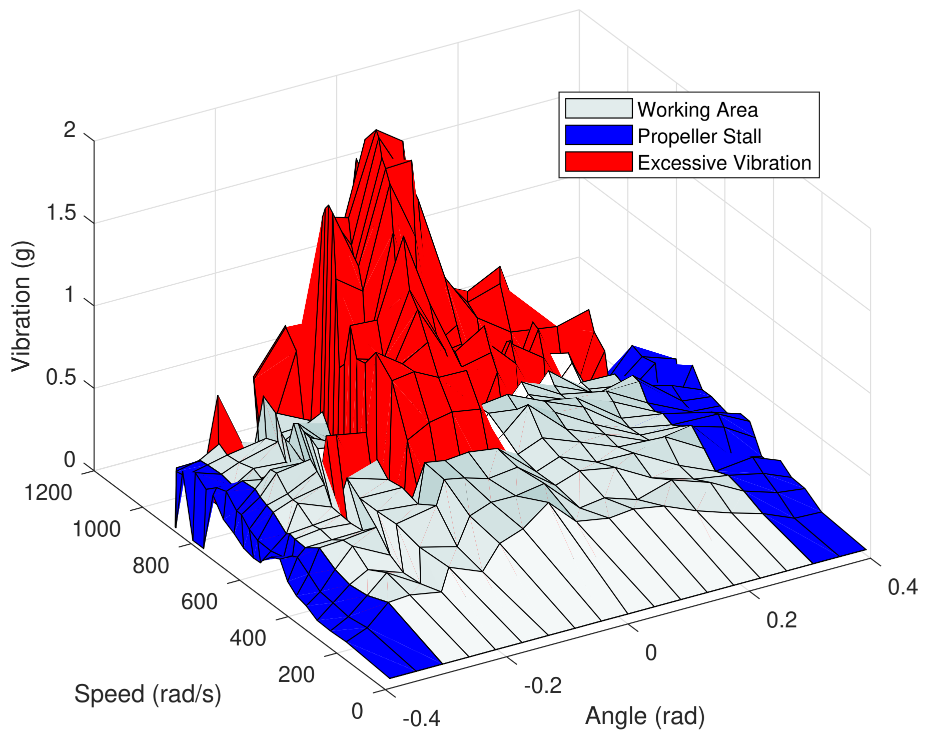


Figure 3.8. Working area of the actuator is represented by the white area, the red area indicates excessive vibration detected by the stand where vibration is larger than 0.5 g, and blue area is the simplified propeller stall range based on the efficiency diagram.

3.2.4 Actuator Comparison

In order to further compare the difference among quadcopters with different types of actuators, the same parameters generated from the experiment are applied to calculate the difference in response time of thrust requests for each type of actuator. We assume that the dynamics of the actuator response is regarded as a second-order model. Usually, the thrust control of the actuator is the main requirement. Assuming that the initial states are with the subscript 0, the target states are with the subscript t. According to Eq. (3.3), the equation for a VPP actuator to change from initial states $[\alpha_0, \omega_0]$ to the desired thrust T_t is given by

$$T_t = K_0(\alpha_0 + \Delta\alpha)(\Delta\omega + \omega_0)^2,$$

where $\Delta\omega$ and $\Delta\alpha$ are desired changing state values. The running time t of achieving the target thrust T_t from T_0 is chosen as

$$t = \max\left(\frac{\Delta\alpha}{V_\alpha} + t_\alpha, \frac{\Delta\omega}{V_\omega} + t_\omega\right),$$

where V_ω is the value of motor rotating speed acceleration (decided by motor), V_α is the propeller angle changing speed (decided by servo), t_α and t_ω are constants delay parameters according to servo and motor, respectively. Therefore, the target force T is a function of running time t, given by

$$T(t) = K_0[\alpha_0 + V_\alpha(t - t_\alpha)][\omega_0 + V_\omega(t - t_\omega)]^2. \tag{3.4}$$

In order to analyse the control difference among these three types of quadcopter actuator, we applied the parameters from our hardware platform and also and referenced the literature [21], where the values are summarized in Table 3.3.

Table 3.3. VPP actuator technical specifications

T_0	t_α	t_ω	V_α	V_ω
0.25 N	0.003 s	0.001 s	10.47 rad/s	6833 rad/s^2

Due to the reason that the VPP can individually change the pitch angle and rotating speed, there are three ways to control the VPP to reach the desired thrust and can be considered as three types of actuator:

1) Ordinary quadcopter actuator. It only controls the motor speed ($\Delta\alpha = 0$) to change the thrust which has a low delay time t_ω. However, with the size of the propeller increases, the speed change will become slower.

2) CVPP quadcopter actuator. It only controls propeller pitch angle ($\Delta\omega = 0$) to change the thrust where it can also deal with the negative force target. Due to the servo control, this method has a high delay time t_α.

3) SVPP quadcopter actuator. It is a hybrid control mode where pitch angle and motor speed can be controlled individually. This is a unique way that can be used to have both advantages of the other two control modes but increase the actuator mechanical complexity. Due to the control redundancy in this way, the desired thrust and torque can be better adjusted. Furthermore, it can obtain a specific torque (within achievable range) under a certain thrust requirement by adjusting the corresponding rotation speed and propeller angle.

Because $t_\alpha > t_\omega$, motor rotating speed will start to change before the servo begins, which gives the control thrust threshold

$$T_{threshold} = K_0\alpha_0\left([V_\omega(t_\alpha - t_\omega)]^2 + 2V_\omega(t_\alpha - t_\omega)\omega_0\right).$$

When the desired thrust variation target is small where $|T_t - T_0| = |\Delta T| <= T_{threshold}$, only motor spinning change $\Delta\omega$ is enough to achieve the target force. Therefore, when the actuator has a high control bandwidth and requires a small range of thrust demand, the motor speed control has a better effect.

However, when the thrust variation target is large where $|\Delta T| > T_{threshold}$, the propeller pitch angle controlled actuator has better force acceleration performance comparing with an ordinary actuator, shown in Fig. 3.9. The comparison between Figs. 3.9(a) and 3.9(b) show that when an agile flight is desired, a configuration with low propeller pitch angle and high motor speed is recommended. In order words, this configuration can provide fast thrust response speed but its efficiency is low.

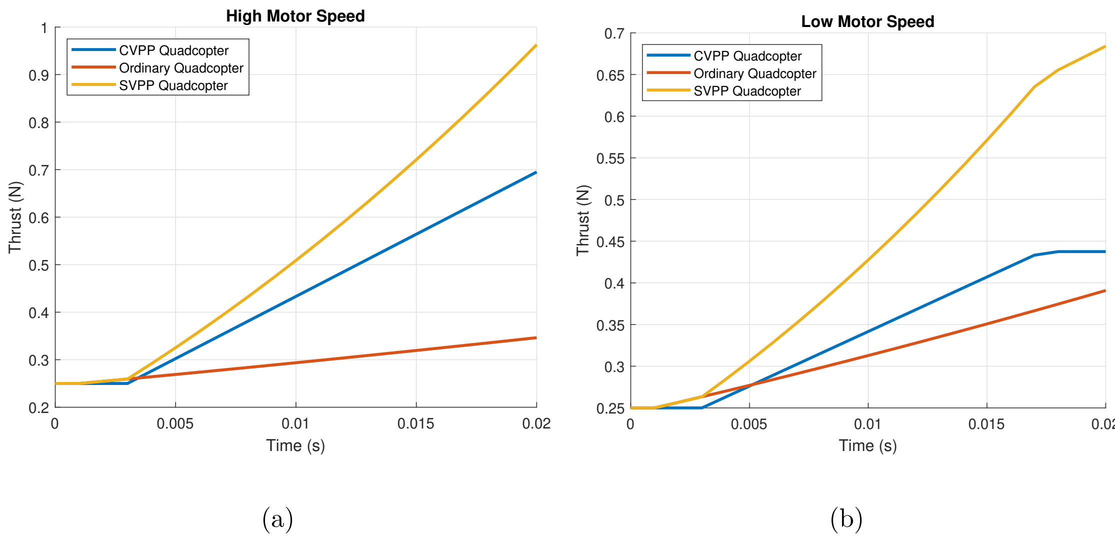


Figure 3.9. Comparison between different initial settings, where (a) starts from $\alpha_0 = 0.1, \omega_0 = 733$ is an efficient configuration and (b) starts from $\alpha_0 = 0.2, \omega_0 = 518$ is an agile configuration.

The thrust response simulation indicates that compared with ordinary quadcopters, VPP quadcopters can quickly achieve the thrust command and provide more control redundancy for the system. Furthermore, propeller pitch angle control enables the quadcopter to provides a negative force which highly expands the control range and also provides better dynamic performance. In addition, research shows that when the control bandwidth is high (over 50 Hz), compared with CVPP quadcopters, SVPP quadcopters have the best control performance among these three types of quadcopters.

Overall, while it has obtained VPP actuators with better manoeuvrability from the helicopter, the VPP quadcopter exhibits great potential in many applications that require high-performance flight. On the other hand, VPP actuators also bring other problems that are different from conventional ones to the VPP quadcopters, such as propeller angle stuck or large vibration during flight. This stems from the extra mechanical structure that can generate rapid thrust change. In general, this shows that although the VPP actuator can benefit the quadcopter for better agile flight, it requires extra system designs to guarantee its safe performance.

To further explore the performance of the VPP quadcopters, including both separately-powered and centrally-powered, general dynamic equations based on Euler angles and rotation matrix are constructed and compared in the following.

3.3 Quadcopter Euler Model

3.3.1 Thrust and Torque Model

Configuration Comparison

Normally, a VPP quadcopter has two different structure configurations, as shown in Fig. 3.10. To facilitate the follow-up description, we use a "X" configuration structure, where the quadcopter heads to the positive direction of the x-axis, as shown in Fig. 3.10 (a). The propeller number is defined from 1 to 4 through clockwise direction, where propeller P_1 is on the x-axis.

Define the propellers P_1 and P_3 rotate clockwise and propellers P_2 and P_4 rotate counter-clockwise to offset the propeller rotation torque. Define the torque and force generate by each propeller according to Eq. (3.3) as M_i and T_i where i is the propeller number. Define the force and torque act on the quadcopter are F and $\boldsymbol{\tau}$, we can construct the input of the quadcopter system as

$$\begin{bmatrix} F \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} T_1 + T_2 + T_3 + T_4 \\ l(T_3 - T_1) \\ l(T_4 - T_2) \\ -M_1 + M_2 - M_3 + M_4 \end{bmatrix}, \tag{3.5}$$

where l is the half length of the diagonal wheelbase of the VPP quadcopter.

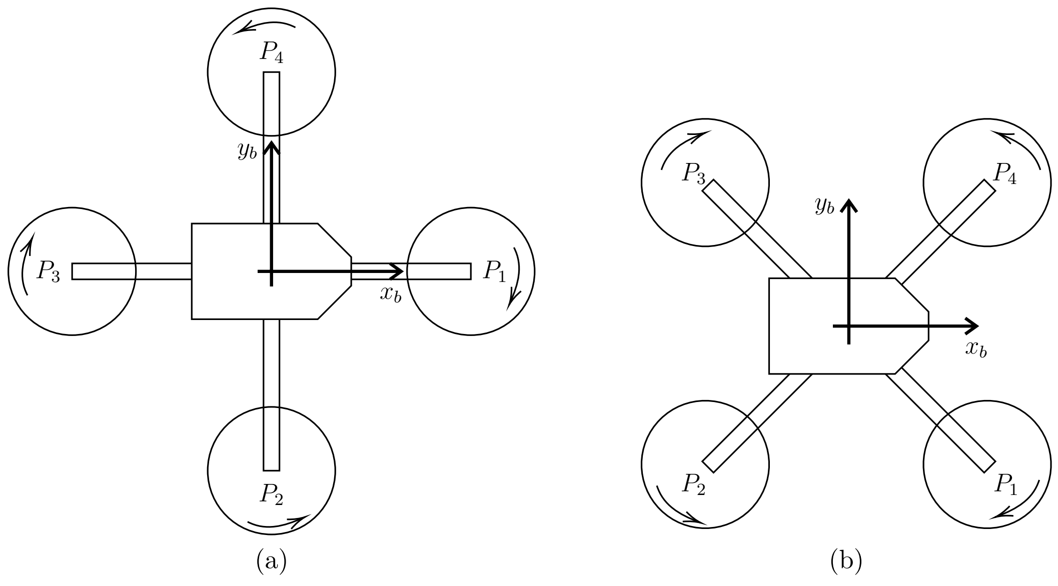


Figure 3.10. The schematic figures of two different quadcopter configurations where (a) is the plus configuration and (b) is the cross configuration.

When the quadcopter structure configuration changes to “X”, shown in Fig. 3.10 (b), Eq. (3.5) changes to

$$\begin{bmatrix} F \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} T_1 + T_2 + T_3 + T_4 \\ \frac{\sqrt{2}}{2} l(-T_1 + T_2 + T_3 - T_4) \\ \frac{\sqrt{2}}{2} l(-T_1 - T_2 + T_3 + T_4) \\ -M_1 + M_2 - M_3 + M_4 \end{bmatrix}.$$

Define a virtual input vector $\mathbf{u}$ instead of the actual control inputs F and $\boldsymbol{\tau}$ by

$$\mathbf{u} = [u_1, u_2, u_3, u_4]^T = [F, \boldsymbol{\tau}]^T. \quad (3.6)$$

In the following section, we will use the plus configuration quadcopter as the standard, and the result is no different than the cross configuration.

VPP Quadcopter Comparison

Since there are two different kinds of VPP quadcopters according to the way of propeller transmission mode, the overall force and torque dynamic equations of the actuator can be different. Suppose the target quadcopter is a CVPP quadcopter, where the spin of all propellers is powered by a central motor placed at the centre of the quadcopter [156]. In this case, all the propellers have the same and fixed spinning speed ω_0, which gives

$$\begin{aligned} T &= K_0\omega_0^2\alpha := b_0\alpha, \\ M &= K_1\omega_0^2 + K_2\alpha^2\omega_0^2 + K_3\omega\alpha := b_1 + b_2\alpha^2 + b_3\alpha, \end{aligned} \tag{3.7}$$

where $b_0 = K_0\omega_0^2$, $b_1 = K_1\omega_0^2$, $b_2 = K_2\omega_0^2$, and $b_3 = K_3\omega_0$ are constant parameters. Substituting Eq. (3.7) and (3.6) into Eq. (3.5) gives

$$\begin{aligned} u_1 =& b_0(\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4), \\ u_2 =& b_0 l(\alpha_3 - \alpha_1), \\ u_3 =& b_0 l(\alpha_4 - \alpha_2), \\ u_4 =& b_2(-\alpha_1^2 + \alpha_2^2 - \alpha_3^2 + \alpha_4^2) + b_3(-\alpha_1 + \alpha_2 - \alpha_3 + \alpha_4). \end{aligned} \tag{3.8}$$

In addition, when using a CVPP quadcopter under normal circumstances, the fixed spinning speed ω_0 will be uncontrollable to make it running efficiently. Therefore, only 4 propeller pitch angles can be controlled for ordinary use. However, when the CVPP quadcopter is in an emergency situation, the spinning speed ω_0 will become controllable.

Consider another type of VPP quadcopter, where the spin of each propeller is powered by an independent motor, named SVPP quadcopter. In this case, the spinning speeds of different propellers can be different [20], therefore, each actuator of the quadcopter can automatically adjust its own spinning speed and propeller pitch angle according to the requirements. Substituting Eq. (3.3) and (3.6) into Eq. (3.5) gives

$$\begin{aligned} u_1 =& K_0(\omega_1^2\alpha_1 + \omega_2^2\alpha_2 + \omega_3^2\alpha_3 + \omega_4^2\alpha_4), \\ u_2 =& K_0 l(\omega_3^2\alpha_3 - \omega_1^2\alpha_1), \\ u_3 =& K_0 l(\omega_4^2\alpha_4 - \omega_2^2\alpha_2), \\ u_4 =& K_1(\omega_3^2 + \omega_1^2 - \omega_2^2 - \omega_4^2) + K_2(\omega_3^2\alpha_3^2 + \omega_1^2\alpha_1^2 - \omega_2^2\alpha_2^2 - \omega_4^2\alpha_4^2) + \\ & K_3(\omega_3\alpha_3 + \omega_1\alpha_1 - \omega_2\alpha_2 - \omega_4\alpha_4). \end{aligned} \tag{3.9}$$

Overall, a comparison between an SVPP quadcopter and a CVPP quadcopter is proposed in Table 3.4 to give a clear statement.

Table 3.4. SVPP and CVPP comparison

	CVPP	SVPP
Power Mode	all of four VPPs are powered by the same motor placed in the centre of the quadcopter, connected by belts	an SVPP actuator can control its rotating speed (and propeller angle) separately
Control Inputs	propeller angle α_i	rotating speed ω_i and propeller angle α_i
Thrust	$T_i = b_0\alpha_i$	$T_i = K_1\omega_i^2\alpha_i$
Torque	$M_i = b_1 + b_2\alpha_i^2 + b_3\alpha_i$	$M_i = K_1\omega_i^2 + K_2\omega_i^2\alpha_i^2 + K_3\omega_i\alpha_i$
Parameters	$b_0 = K_0\omega_0^2$, $b_1 = K_1\omega_0^2$, $b_2 = K_2\omega_0^2$, and $b_3 = K_3\omega_0$	

3.3.2 Rigid Body Model

The overall dynamics model of the VPP quadcopter is the same as that of the fixed-pitch propeller quadcopter. We make the following assumptions when modelling the quadcopter dynamics:

1) The quadcopter is a rigid body with a symmetrical physical structure and uniform mass distribution.

2) The moment of inertia is constant.

3) The earth is regarded as a horizontal plane.

4) Gravitational acceleration does not change with height.

5) The gyro effect should be ignored.

Two frames are used to build the dynamic model: a global frame $\{O_g, x_g, y_g, z_g\}$ and a body frame $\{O_b, x_b, y_b, z_b\}$, where O_b is fixed to the centre of the quadcopter, which is shown in Fig. 3.11. O_g is related to O_b by a position vector $[x, y, z]^T$. Angle vector $[\phi, \theta, \psi]^T$ represents the orientation angle from body frame to global frame, using yaw-pitch-roll notation. The angular rate in the body frame is given by $[p, q, r]^T$. The linear speed in global frame is given by $[u, v, w]^T$. Therefore, the state vector can be constructed as $\mathbf{x} = \left[x, y, z, \phi, \theta, \psi, p, q, r, u, v, w\right]^T$.

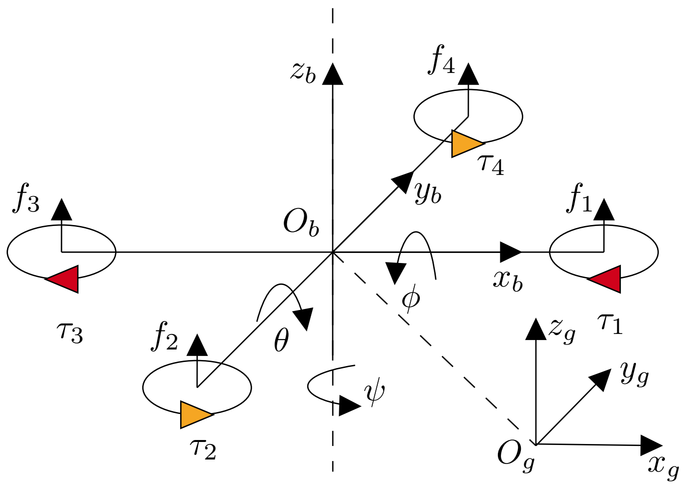


Figure 3.11. The schematic diagram of a quadcopter.

Translation Model

Since the quadcopter does not have a significant lift surface, it seems that the air resistance can be expressed by the following formula

$$\mathbf{F}_D = -k_\beta |v| v, \tag{3.10}$$

where k_β is the linear air resistance coefficient. Define the mass of the CVPP quadcopter as m, and the gravitational acceleration constant as g. According to Newton's law, the translation equation of the quadcopter is given by

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \mathbf{R}_b^g \begin{bmatrix} 0 \\ 0 \\ u_1/m \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \mathbf{F}_D/m, \tag{3.11}$$

where $\mathbf{R}_b^g$ transfer the quadcopter kinematics from body frame to global frame, which is constructed according to the order of rotation (roll-pitch-yaw) as follows

$$\mathbf{R}_b^g = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}.$$

In general, substituting Eq. (3.10) to Eq. 3.11 gives the translational motion of the quadcopter contains coordinate transformation from the body frame to the global frame as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \\ \dfrac{(\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi)u_1 - k_\beta u|u|}{m} \\ \dfrac{(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi)u_1 - k_\beta v|v|}{m} \\ \dfrac{(\cos\theta\cos\phi)u_1 - k_\beta w|w|}{m} - g \end{bmatrix}. \tag{3.12}$$

Rotation Model

According to the rigid body rotation, the quadcopter rotation is presented by angular velocities, given by

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \mathbf{I}^{-1}\left(\begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \mathbf{I}\begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}\right),$$

where $\mathbf{I}$ is the inertia matrix. From the assumptions of the symmetry structure of the quadcopter, $\mathbf{I}$ is given as

$$\mathbf{I} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}, \tag{3.13}$$

where I_x, I_y and I_z are moments of inertia in body frame.

When consider the rotations of the quadcopter are large, suppose the Euler angle notion is given by Z, Y, X, it gives

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \mathbf{R}_\phi \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \mathbf{R}_\phi \mathbf{R}_\theta \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}, \tag{3.14}$$

where

$$\mathbf{R}_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}, \quad \mathbf{R}_\theta = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}.$$

Therefore, simplify Eq. (3.14) and invert the matrix gives the relation between the deviation of Euler angles and angular velocity vector as

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan\theta \sin\phi & \tan\theta\cos\phi \\ 0 & \cos\phi & -\sin\phi \\ 0 & \dfrac{\sin\phi}{\cos\theta} & \dfrac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{3.15}$$

Finally from Eq.(3.15), the rotational motion of the quadcopter can be presented as:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\ q\cos\phi - r\sin\phi \\ q\sin\phi\sec\theta + r\cos\phi\sec\theta \\ qr\dfrac{I_z - I_y}{I_x} + u_3\dfrac{1}{I_x} - p|p|\dfrac{k_\gamma}{I_x} \\ rp\dfrac{I_x - I_z}{I_y} + u_2\dfrac{1}{I_y} - q|q|\dfrac{k_\gamma}{I_y} \\ u_4\dfrac{1}{I_z} - r|r|\dfrac{k_\gamma}{I_z} \end{bmatrix}. \tag{3.16}$$

where k_γ is the rotation air resistance coefficient.

In addition, when the quadcopter only flies at a small rotation angle, the deviation of Euler angles and angular velocity vector can be considered as the same, thus simplifying the model.

Since the dynamic model of the VPP quadcopter is similar to that of conventional ones, the VPP quadcopter can have similar characteristics, such as simple structure and fast dynamic response.

3.4 Quadcopter Rational Matrix Model

Although the Euler angle model is easy to build and understand, there are some shortcomings. First, Euler angles have different adopted notations (such as yaw-pitch-roll convention). Euler angles decompose one orientation of a rigid body with respect to a fixed reference coordination, into three rotations to along with, for example, z-y-x axis (depends on the notation). With the different notations, the same orientation can be expressed by different Euler angles but with the same rotational matrix. Second, when using Euler angles, the system will lose one degree of freedom in a three-dimensional mechanism when one of the Euler angles reaches $\pi/2$, which is also called Gimbal lock. This will not be a serious problem when only considering

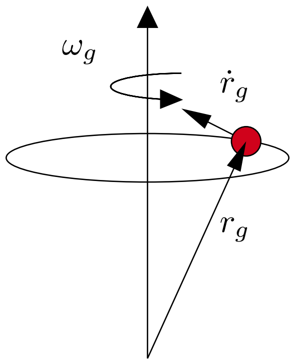


Figure 3.12. The schematic diagram of rotation velocity.

the stable flight of the quadcopter but it will cause consequences when used for agile flight, such as a crash.

One of our motivations, specifically, regarding the tic-toc manoeuvrer control, will be heavily affected by Gimbal lock. Tic-toc is one of the typical operations that need the quadcopter to fly around such a Gimbal lock angle. Therefore, the angle ambiguities caused by the Gimbal lock will lead to the loss of one degree of freedom of the quadcopter model, which is a big problem for our research. For this reason, further research on manoeuvring will use the rotation matrix to build a dynamic system.

Therefore, we can use the same frame setting (O_g and O_b) with the Euler angle model where O_g is related to O_b by a position vector $\mathbf{x}$. The linear speed vector in the global frame is given by $\mathbf{v}$. Rotation matrix $\mathbf{R} \in SO(3)$ represents the orientation from the body frame to the global frame. The angular rate vector in the body frame is given by $\boldsymbol{\omega}_b$.

As shown in Fig. 3.12, it gives the derivative equation of a vector $\boldsymbol{r}_g$ rotating around a fixed axis as

$$\dot{\boldsymbol{r}}_g = \boldsymbol{\omega}_g \times \boldsymbol{r}_g. \tag{3.17}$$

Set a rotation matrix $\mathbf{R}_b^r$ that can transfer coordinate from body frame to reference frame, as a combination of 3 body frame unit vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3$ in reference frame as

$\mathbf{R}_b^r = [\boldsymbol{b}_1\ \boldsymbol{b}_2\ \boldsymbol{b}_3]$. Substituting (3.17) to $\dot{\mathbf{R}}_b^r$ gives

$$\begin{aligned}
\dot{\mathbf{R}}_b^r &= \left[\dot{\boldsymbol{b}}_1\ \dot{\boldsymbol{b}}_2\ \dot{\boldsymbol{b}}_3\right] \\
&= [\boldsymbol{\omega}_r \times \boldsymbol{b}_1\ \boldsymbol{\omega}_r \times \boldsymbol{b}_2\ \boldsymbol{\omega}_r \times \boldsymbol{b}_3] \\
&= [\mathbf{R}_b^r\boldsymbol{\omega}_b \times \boldsymbol{b}_1\ \mathbf{R}_b^r\boldsymbol{\omega}_b \times \boldsymbol{b}_2\ \mathbf{R}_b^r\boldsymbol{\omega}_b \times \boldsymbol{b}_3] \\
&= [\mathbf{R}_b^r\boldsymbol{\omega}_b \times \mathbf{R}_b^r\boldsymbol{e}_1\ \mathbf{R}_b^r\boldsymbol{\omega}_b \times \mathbf{R}_b^r\boldsymbol{e}_2\ \mathbf{R}_b^r\boldsymbol{\omega}_b \times \mathbf{R}_b^r\boldsymbol{e}_3] \\
&= [\mathbf{R}_b^r(\boldsymbol{\omega}_b \times \boldsymbol{e}_1)\ \mathbf{R}_b^r(\boldsymbol{\omega}_b \times \boldsymbol{e}_2)\ \mathbf{R}_b^r(\boldsymbol{\omega}_b \times \boldsymbol{e}_3)] \\
&= \mathbf{R}_b^r\,[\boldsymbol{\omega}_b \times \boldsymbol{e}_1\ \boldsymbol{\omega}_b \times \boldsymbol{e}_2\ \boldsymbol{\omega}_b \times \boldsymbol{e}_3] \\
&= \mathbf{R}_b^r\hat{\boldsymbol{\omega}}_b, \qquad (3.18)
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{e}_1 &= [1\ 0\ 0]^T, \\
\boldsymbol{e}_2 &= [0\ 1\ 0]^T, \\
\boldsymbol{e}_3 &= [0\ 0\ 1]^T,
\end{aligned}$$

are defined in the body frame and $\hat{\boldsymbol{\omega}}_b$ is a skew-symmetric matrix formed from $\boldsymbol{\omega}_b$ which can be written as

$$\hat{\boldsymbol{\omega}}_b = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}.$$

The equations of motion of this quadcopter can be written according to definitions as

$$\begin{aligned}
\dot{\mathbf{x}}_g &= \mathbf{v}_g, && (3.19) \\
\dot{\mathbf{v}}_g &= \frac{F}{m}\mathbf{R}\boldsymbol{e}_3 - g\boldsymbol{e}_3, && (3.20) \\
\dot{\mathbf{R}} &= \mathbf{R}\hat{\boldsymbol{\omega}}_b, && (3.21) \\
\boldsymbol{I}\dot{\boldsymbol{\omega}}_b &= -\boldsymbol{\omega}_b \times \boldsymbol{I}\boldsymbol{\omega}_b + \boldsymbol{\tau}, && (3.22)
\end{aligned}$$

where $\boldsymbol{\tau} = [u_2, u_3, u_4]^T$ and $\boldsymbol{I}$ is the inertia matrix expressed in the body frame.

The analysis of the dynamics makes the manoeuvrer simulation of quadcopter become possible and can be used to better analyse the VPP quadcopter system. Moreover, a 2D version of the VPP quadcopter dynamics is utilised in next section to simplify the manoeuvrers and assist manoeuvrer analysis.

3.5 Planar Quadcopter Model

Next, we present the planar quadcopter model. This model is special since it is a simplified planar quadcopter model that can represent the characteristics of the two-dimensional motion of the quadcopter. The reason why we consider using this model is firstly because the controller based on the 2D model design can be further studied and implemented in the 3D environment. It is inspired by [31] and [49], where a 2D simplified model is used to design a flipping action controller. Then, the designed 2D controller was further extended to a 3D environment and validated in the practical. Their work shows the feasibility of this simplifying process. Secondly, it is complicated and difficult to directly analyse the manoeuvrer motion in 3D environment. Because there are almost double degrees of freedom (DOFs) compared to the 2D environment (6 DOFs in 3D and 3 DOFs in 2D). Another important reason to use a 2D model is that one of the goals of this research is to analyse and study a tic-toc manoeuvrer with plane motion as the mainstay. This manoeuvrer requires the quadcopter to operate like an inverse simple gravity pendulum, which can be regarded as an idealized planar motion mathematical model [157].

Although this work can approximate the study of the tic-toc manoeuvrer by a simplified model, there is still a gap between 2D and 3D. One of the most important problems is that the tic-toc manoeuvrer can not be equivalent to a mathematical pendulum model. Because a real quadcopter is flown in the air, it is interpreted as being vulnerable to external disturbances, such as friction, interference, etc., and loses its stability.

A 2D controller can not deal with such controllability loss problems caused by the lack of DOFs. Extra controllers are required to be designed for these problems. Further research will try to bridge this gap through learning-based algorithms.

In general, using a plane model to analyse tic-toc manoeuvrers has the advantages of simplifying the dynamic model, clarifying the manipulation process, and facilitating the algorithm design. Therefore, the use of a simplified model is of critical significance for this work to analyse and realize this tic-toc manoeuvrer.

As the first step of planar quadcopter analysis, a primary concern of the consistency between 2D and 3D environments is needed. This work considers the plus configuration quadcopter where thrust T_1 and T_3 are within the control plane (xz plane), shown in Fig. 3.13.

According to the analysis shown in Section 3.2, the VPP actuator can control its torque and thrust separately. It is noticeable that substitutes $\alpha = 0$ to Eq. (3.3)

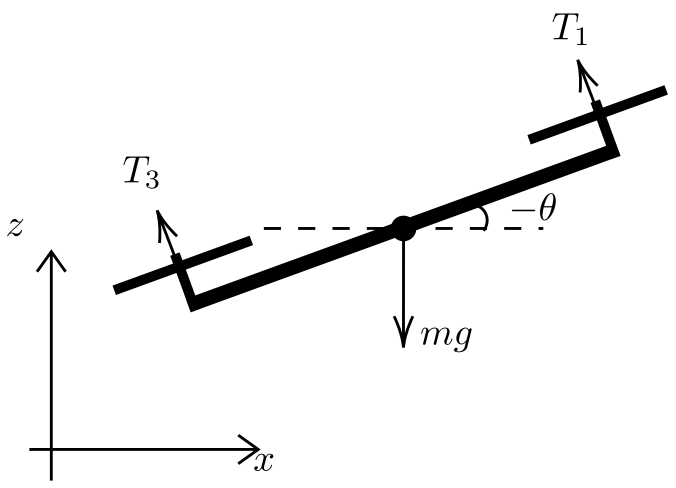


Figure 3.13. The planar quadcopter model.

gives

$$\begin{aligned} T &= 0, \\ M &= K_1\omega^2. \end{aligned}$$

Therefore, we could make an assumption that the thrust T_2 and T_4 can balance the angles, ϕ and ψ, to 0, which gives

$$\begin{aligned} \phi, \dot{\phi} &\to 0, \\ \psi, \dot{\psi} &\to 0, \\ T_2 - T_4 &\to 0, \\ u_4 &\to 0. \end{aligned} \tag{3.23}$$

Substituting Eq. (3.23) to Eqs. (3.12) and (3.16) gives the overall planar quadcopter dynamic equations as

$$\begin{aligned} ma_z &= (T_1 + T_2 + T_3 + T_4)\cos\theta - mg, \\ ma_x &= (T_1 + T_2 + T_3 + T_4)\sin\theta, \\ I_x a_\theta &= L(T_3 - T_1)/2, \end{aligned} \tag{3.24}$$

where m is the mass of the planar quadcopter, L is the length between No.1 and 3 propellers, and a_x, a_z and a_θ indicate the second derivative of x, z and θ.

To prevent interference with T_1 and T_3, the value of T_2 and T_4 should be as low as possible, which gives

$$T_2 + T_4 \to 0. \tag{3.25}$$

Therefore, rebuild the model Eq. 3.24 with the input parameters T_1, T_3 gives

$$\begin{aligned} a_z &= (T_1 + T_3)\cos\theta/m - g, \\ a_x &= (T_1 + T_3)\sin\theta/m, \\ I_x a_\theta &= L(T_3 - T_1)/2. \end{aligned} \tag{3.26}$$

To avoid confusion from the symbols, the planar model uses the parameters along the following definitions shown in Table. 3.5.

Table 3.5. Planar quadcopter notations list.

Symbol	Description	Symbol	Description
s_x, s_z	position	m	mass
v_x, v_z	velocity	L	length
T_1, T_3	thrust	I	moment of inertia
θ	angle	w	angle velocity
g	acceleration of gravity		

Defined a new input vector

$$\boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} T_1 + T_3 \\ T_3 - T_1 \end{bmatrix} \tag{3.27}$$

and a state vector

$$\boldsymbol{x} = [s_x, s_z, v_x, v_z, \theta, w]^T.$$

Then, combine Eqs. (3.27) and (3.26) gives

$$\dot{\boldsymbol{x}} = f(x,u) = \begin{bmatrix} \dot{s}_x \\ \dot{s}_z \\ \dot{v}_x \\ \dot{v}_z \\ \dot{\theta} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} v_x \\ v_z \\ -\dfrac{\sin\theta}{m}u_1 \\ \dfrac{\cos\theta}{m}u_1 - g \\ w \\ \dfrac{L}{2I}u_2 \end{bmatrix}, \tag{3.28}$$

where $I = \dfrac{1}{12}mL^2$.

Eq. (3.28) can be rewritten as

$$\dot{\boldsymbol{x}} = f(x) + \sum_{i=1}^{2} g_i(x)u_i, \tag{3.29}$$

where

$$f(x) = \begin{bmatrix} v_x \\ v_z \\ 0 \\ -g \\ w \\ 0 \end{bmatrix},$$

and

$$g_1(x) = \left[0, 0, -\frac{\sin\theta}{m}, \frac{\cos\theta}{m}, 0, 0\right]^T,$$
$$g_2(x) = \left[0, 0, 0, 0, 0, -\frac{L}{2I}\right]^T.$$

In addition, this research also considers the situation that only one force T_1 is available at the end of the planar quadcopter. Therefore, the planar quadcopter can no longer be regarded as a stick rotating around the centre. Assume the distance between the rotation centre and the end of the stick is $d = kL$, shown in Fig. 3.14. The moment of inertia is given by

$$I = \int_{-d}^{L-d} d^2 \frac{m}{L} dd = mL^2(k^2 - k + \frac{1}{3}),$$

and the total thrust f_T and torque τ_T are given by

$$\begin{aligned} f_T =& f, \\ \tau_T =& (L-d)f. \end{aligned} \tag{3.30}$$

A related juggling named ‘Devil Stick’ shows a movement similar to our research target, the tic-toc manoeuvrer. The significance of establishing a plane quadcopter model that only has one thrust is to imitate the dynamics of juggling and analyse the relationship between the motion of the rigid body and the rotating centre, so as to finally realize the fine control of the tic-toc manoeuvrer.

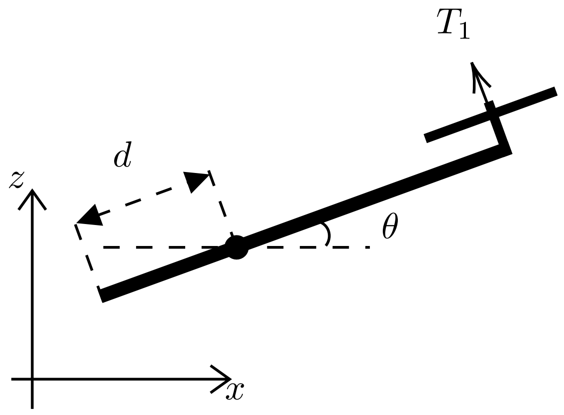


Figure 3.14. The planar quadcopter model with one force.

3.6 Summary

This chapter details the actuator and body dynamics of the VPP quadcopters. In Section 3.2, we first give the dynamic equations of VPPs from the literature. Then, a VPP actuator experimental hardware platform that can validate the dynamic equations is designed and constructed. Both positive and negative propeller pitch angles are tested and analysed. Although the data with high motor speed will vibrate and cause inaccuracy, the overall data trend verifies the dynamic equations. It indicates that the VPP quadcopter does not have a large difference compared with ordinary quadcopters except for its inversion flight ability. The actuator efficient working area is analysed according to the experiment as well. According to the features we present, in the following chapters, these capabilities are further analysed and fault-tolerant control laws are developed to explore the safety-critic flight of the VPP quadcopters.

Furthermore, we compare the actuator dynamic performance between CVPP, SVPP and ordinary quadcopters according to the simulation and experiment results. It shows that VPP actuators can have a faster thrust response speed which can further extend the manoeuvrability and safety-critic performance of the quadcopter applications.

A commonly used Euler model to describe the quadcopter dynamic is presented in Section 3.3. However, since the quadcopter has to fly with a large manoeuvrer, a dynamic model based on the rational matrix is presented in Section 3.4. Finally, a simplified 2D planar model of the VPP quadcopter is presented for designing a tic-toc manoeuvrer controller.

In summary, this chapter points out that the VPP actuator can provide different torques while maintaining the same thrust. This is the first feature of VPP found in this thesis, which is the basis for the extended research of Chapter 4 and 5. Then, this chapter shows the second feature that a VPP actuator can quickly change the thrust to positive and negative by adjusting the propeller pitch angle. Chapter 6 shows the research content inspired by this feature. In order to ensure the efficiency, safety and agility of the VPP actuator, the rotation speed and propeller angle range are analysed according to the efficiency curves and working areas. Based on the presented VPP actuator features, it shows that the VPP quadcopter is capable of overcoming the failure of one of the actuators and is able to perform many agile movements that ordinary quadcopters cannot.

Chapter 4

Control of Separated-Powered Variable Pitch Propeller Quadcopters Subject to One Actuator Failure

4.1 Introduction

Attributable to the wide range of recreational or commercial needs, urban air parcel delivery and passenger transmit are becoming an important part of many day-to-day applications in the future. Compared to many other types of usages, such as photography and detection, the tasks of transportation have higher requirements for flight safety. Since many of these tasks require VPP quadcopters to achieve high flight performance in terms of both reliability and safety, the fault-tolerant control research of them is very important [101].

Fault-tolerant control is to control the quadcopter to maintain a relatively stable state in the presence of some faults such as rotor failures [158, 98]. Fault-tolerant control of quadcopters is of great importance and have attracted extensive studies [94, 99, 159, 160].

However, the conventional quadcopter which has simple structural design, has certain limitations when facing safety-critical flight tasks. For conventional quadcopters, when one or more rotors/propellers fail to work properly, the platforms usually become extremely hard to control and sophisticated controllers must be designed [26, 161]. More importantly, even under fault-tolerant controllers, conventional quadcopters with faults are only able to fly in very special manners such as continuously rotating [95, 162, 163]. As a comparison, SVPP quadcopters shows strong fault-tolerant ability as we show in this chapter.

In addition, although fault-tolerant control of quadcopters has been studied extensively [164, 165, 158, 166], the case of VPP quadcopters remains largely unexplored.

The contributions of this chapter are stated as the following. First, we explore a new fault-tolerant feature of SVPP quadcopters based on the findings shown in Chapter 3. Then, we conduct the controllability analysis of a VPP quadcopter with one propeller failure based on the linearised dynamical model. It is shown that, when one propeller fails, a faulty SVPP quadcopter still remains fully controllable. For comparison purposes, we also analyse the controllability of conventional quadcopters with one rotor failure and show that they are uncontrollable with rotor failures. Secondly, we identify the equilibrium point and derive the linearised dynamical model of VPP quadcopters. Based on the linearised model, we design an LQR controller to handle quadcopter control under one actuator failure. Based on such a simple controller, the VPP quadcopter could accurately track a given trajectory subject to wind disturbances or noises. This property is important for safety-critical tasks such as flying taxis, where it is necessary to transport the human passengers safely back to a safe place in the presence of propeller failures. Thanks to this property, VPP quadcopters provide an important alternative type of platform for safety-critical aerial tasks.

It is worth mentioning that we do not consider fault detection, isolation, or switching controllers in this chapter. These important topics will be addressed in the future work. The focus of this chapter is to explore the fault-tolerant ability of VPP quadcopters which has not been reported in the literature.

4.2 Problem Setup

Consider the scenario that one of the actuators (contain propeller and motor) is broken. Without loss of generality, suppose the FDI system can immediately detect the failure of propeller 1. Then, actuator 1 provides zero force and zero torque. That is f_1 and τ_1 are always zero. Here, we consider a '+' configuration SVPP quadcopter where Eq. (3.9) becomes

$$\begin{aligned}
u_1 =& K_0(\omega_2^2\alpha_2 + \omega_3^2\alpha_3 + \omega_4^2\alpha_4), \\
u_2 =& K_0\omega_3^2\alpha_3, \\
u_3 =& K_0(\omega_4^2\alpha_4 - \omega_2^2\alpha_2), \\
u_4 =& K_1(\omega_3^2 - \omega_2^2 - \omega_4^2) + K_2(\omega_3^2\alpha_3^2 - \omega_2^2\alpha_2^2 - \omega_4^2\alpha_4^2)+ \\
& K_3(\omega_3\alpha_3 - \omega_2\alpha_2 - \omega_4\alpha_4),
\end{aligned} \tag{4.1}$$

where the parameter settings are shown in Table. 4.1.

Although propeller 1 fails, there remain six independent control quantities, $\omega_2, \omega_3, \omega_4$ and $\alpha_2, \alpha_3, \alpha_4$. To generate desired $\mathbf{u}$, the six control quantities are still redundant, which is the fundamental reason why the system remains controllable in the presence of propeller failures.

According to the features we found in Chapter 3, the thrust and torque of the VPP actuator can be independently controlled by the rotation speed and propeller pitch angle. And because we consider the separated-powered VPP quadcopter as the target platform, where both ω_i and α_i can be adjusted independently by each propeller. As a result, the torque and thrust of each actuator of the SVPP quadcopter can be regarded as independent control inputs.

In addition, torque is generated by a variety of factors, including frictional drag and the drag component from propeller motion. Therefore, even if the pitch angle and thrust are zero, the torque generated by the actuator of the SVPP quadcopter can be non-zero.

4.3 Linearisation and Controllability Analysis

In this section, we identify the equilibrium point of the system in the presence of a propeller failure, linearise the system at the equilibrium point, and conduct controllability analysis. Due to the rotational symmetry structure of the quadcopter,

Table 4.1. SVPP quadcopter parameters.

Parameters	Name	Value	Unit
Mass	m	1	kg
Arm length	l	0.35	m
Gravitational acceleration	g	9.8	m/s^2
Pitch angle limit	$\alpha_{\max}$	0.26	rad
	$\alpha_{\min}$	-0.26	rad
Motor speed limit	$\omega_{\max}$	1047	rad/s
	$\omega_{\min}$	0	rad
Force coefficient	K_0	$4.5 \cdot 10^{-5}$	-
Torque coefficient	K_1	$9.1 \cdot 10^{-7}$	-
	K_2	$1.8 \cdot 10^{-8}$	-
	K_3	$3.8 \cdot 10^{-6}$	-

the control algorithm developed with the same principle can be implemented on the SVPP quadcopter with failure on another actuator.

4.3.1 Equilibrium Point

Consider the state

$$\mathbf{x}^* = \left[x^*, y^*, z^*, 0, 0, \psi^*, 0, 0, 0, 0, 0, 0\right]^T .$$

In order to keep the system at this state, the input must be

$$\mathbf{u}^* = [mg, 0, 0, 0]^T. \tag{4.2}$$

Substituting Eq. (4.2) into Eq. (4.1) yields

$$\begin{aligned} mg =& K_0\omega_2^2\alpha_2 + k_1\omega_4^2\alpha_4, && (4.3)\\ 0 =& K_0\omega_4^2\alpha_4 - k_1\omega_2^2\alpha_2, && (4.4)\\ 0 =& K_0\omega_3^2\alpha_3, && (4.5)\\ 0 =& K_1(\omega_3^2 - \omega_2^2 - \omega_4^2) + K_2(\omega_3^2\alpha_3^2 - \omega_2^2\alpha_2^2 - \omega_4^2\alpha_4^2) + \\ & K_3(\omega_3\alpha_3 - \omega_2\alpha_2 - \omega_4\alpha_4). && (4.6) \end{aligned}$$

The next step is to solve the above equations. Equation (4.5) implies that $\omega_3^2\alpha_3 = 0$. It is implied from Eq. (4.6) that $\omega_3 \neq 0$; otherwise, the right-hand side of Eq. (4.6) is less than zero. As a result, we know $\alpha_3 = 0$ and $\omega_3 \neq 0$, which means propeller 3 still spins but with zero pitch angle. On the other hand, equations Eq. (4.3) and

Eq. (4.4) imply

$$K_0\omega_2^2\alpha_2 = K_0\omega_4^2\alpha_4 = \frac{mg}{2}, \tag{4.7}$$

which means propellers 2 and 4 provide forces to counter gravity. Without loss of generality, suppose

$$\omega_2 = \omega_4. \tag{4.8}$$

As a result,

$$\alpha_2 = \alpha_4, \tag{4.9}$$

Substituting Eq. (4.5), (4.8), (4.9) into Eq. (4.6) gives

$$K_1\omega_3^2 = 2(K_1\omega_2^2 + K_2\omega_2^2\alpha_2^2 + K_3\omega_2\alpha_2). \tag{4.10}$$

Substituting Eq. (4.7) to Eq. (4.10) yields

$$K_1\omega_3^2 = 2\left[K_1\omega_2^2 + \frac{K_2}{K_0^2}\left(\frac{mg}{2\omega_2}\right)^2 + \frac{K_3mg}{2K_0\omega_2}\right] := r(\omega_2), \tag{4.11}$$

where $r(\omega_2)$ represents the right-hand side of Eq. (4.11).

Next we identify the range of the value of $r(\omega_2)$. The derivative of $r(\omega_2)$ with respect to ω_2 is

$$\frac{\mathrm{d}r(\omega_2)}{\mathrm{d}\omega_2} = 2\left(2K_1\omega_2 - \frac{K_2m^2g^2}{2K_0^2\omega_2^3} - \frac{K_3mg}{2K_0\omega_2^2}\right). \tag{4.12}$$

By analyzing Eq. (4.12), we notice that $r(\omega_2)$ is a monotonically increasing function when $\omega_3 \in [0, 1047]$. In addition, Eq. (4.7) implies that, when the pitch angle takes the maximum value $\alpha_2 = 0.26$, the rotating speed would take the minimum value $\omega_2 = 649$. As a result, $\omega_2 \in [649, 1047]$. Substituting the interval into Eq. (4.11) gives $r(\omega_2) \in [0.7714, 2]$. Meanwhile, the left-hand side of Eq. (4.11) satisfies $K_1\omega_3^2 \in [0, 1]$ when $\omega_3 \in [0, 1047]$. By combining the bounds of the left- and right-hand sides of Eq. (4.11), we know

$$K_1\omega_3^2 = r(\omega_2) \in [0.7714, 1]. \tag{4.13}$$

Substituting Eq. (4.13) to Eq. (4.11) yields $\omega_2 \in [649, 740]$ and $\omega_3 \in [919, 1047]$. We simply choose $\omega_3 = 942$, a intermediate value in $[919, 1047]$. Note that ω_3 is fixed all the time.

By substituting $\omega_3 = 942$ into Eq. (4.11), we can obtain four solutions: $\omega_2^* = -23$, 23, 665, and -665. Since $\omega_2 \in [649, 740]$ as aforementioned, $\omega_2^* = 665$ in the only feasible solution. Then, substituting ω_2^* into Eq. (4.7) gives $\alpha_2^* = 0.2476$.

Let $\mathbf{z} = [\omega_2, \alpha_2, \alpha_3, \alpha_4]^T$. It follows from the above analysis that, in order to keep the system at the state $\mathbf{x}^*$, $\mathbf{z}^*$ should be

$$\begin{aligned}\mathbf{z}^* &= \left[\omega_2^*, \alpha_2^*, \alpha_3^*, \alpha_4^*\right]^T \\ &= \left[665, 0.2476, 0, 0.2476\right]^T.\end{aligned}$$

4.3.2 Linearised Model

To linearise the nonlinear dynamical system, consider $\bar{\mathbf{z}} = \mathbf{z} - \mathbf{z}^*$ and $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^*$. Let $\mathbf{F}(\bar{\mathbf{x}}, \bar{\mathbf{z}}) \in \mathbb{R}^{12}$ be the right-hand side of Eq. (3.12) and (3.16). Then, the linearised model is

$$\dot{\bar{\mathbf{x}}} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\bar{\mathbf{z}},$$

where

$$\begin{aligned}\mathbf{A} &= \left.\frac{\partial \mathbf{F}(\bar{\mathbf{x}}, \bar{\mathbf{z}})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*, \mathbf{z}=\mathbf{z}^*} \\ &= \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{E}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \in \mathbb{R}^{12\times12}, \\ \mathbf{E}_{3\times3} &= \begin{bmatrix} g\sin\psi^* & g\cos\psi^* & 0 \\ -g\cos\psi^* & g\sin\psi^* & 0 \\ 0 & 0 & 0 \end{bmatrix}, \\ \mathbf{I}_{3\times3} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\ \mathbf{0}_{3\times3} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\end{aligned}$$

and

$$
\begin{aligned}
\mathbf{B} &= \left.\frac{\partial \mathbf{F}(\bar{\mathbf{x}}, \bar{\mathbf{z}})}{\partial \mathbf{z}}\right|_{\mathbf{x}=\mathbf{x}^*, \mathbf{z}=\mathbf{z}^*} \\
&= \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & h_3\frac{l}{I_x} & 0 & 0 \\
-h_2\frac{l}{I_y} & 0 & -h_4\frac{l}{I_y} & n_4 - n_2 \\
-m_2 & m_3 & -m_4 & -o_2 - o_4 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\frac{k_1}{m}\omega_2^{*2} & \frac{k_1}{m}\omega_3^{*2} & \frac{k_1}{m}\omega_4^{*2} & p_4 + p_2
\end{bmatrix} \in \mathbb{R}^{12\times 4},
\end{aligned}
$$

with

$$
\begin{aligned}
h_i &= K_0\omega_i^{*2}, \\
m_i &= 2K_2\alpha_i^*\omega_i^{*2} + K_3\omega_i^*, \\
n_i &= 2K_0\omega_i^*\alpha_i^*\frac{l}{I_y}, \\
o_i &= (2K_1\omega_i^* + 2K_2\omega_i^*\alpha_i^{*2} + K_3\alpha_i^*)\frac{1}{I_z}, \\
p_i &= 2\frac{K_0}{m}\omega_i^*\alpha_i^*.
\end{aligned}
$$

4.3.3 Controllability Analysis of VPP Quadcopters

It can be calculated that the rank of the controllability matrix is $\text{rank}(\mathbf{Q}_C) = \text{rank}[\mathbf{B}\ \mathbf{AB}\ \mathbf{A}^2\mathbf{B}\ ...\ \mathbf{A}^{11}\mathbf{B}] = 12$. As a result, the controllability matrix is of full row rank. Hence, all the states remain controllable when one propeller fails. Although this controllability is analysed based on the linearised system, we show later by simulation that the nonlinear model of a VPP quadcopter would be able to be fully controlled around the equilibrium point. This is a significant merit of VPP quadcopters compared to the conventional ones. The fundamental reason for this merit is that the VPP control system has eight independent control inputs whereas a conventional quadcopter only has four. In the future, we will study the control of VPP quadcopters subject to two or more propeller failures.

4.3.4 Controllability Analysis of Conventional Quadcopters

For comparison purposes, we analyse the controllability of conventional fixed-pitch quadcopters with one rotor failure in this section. We consider a specific mechanical configuration where rotors located on the same arm spin in opposite directions. For such a configuration, a fixed-pitch quadcopter could have an equilibrium when one rotor fails.

Suppose propeller 1 of a conventional quadcopter fails to provide any thrust or torque. The dynamical model Eqs. (3.12) and (3.16) and input Eq. (4.1) also apply to the conventional quadcopter. The only difference is that the pitch angles α_i in Eq. (4.1) are fixed and hence could be merged with the coefficients k_i in Eq. (4.1) to form a new coefficient k_{ci}, where the subscript c denotes "conventional". Note that rotor 2 and rotor 4 spin in opposite directions such that there exists an equilibrium state when rotor 1 fails. As a result, Eq. (4.1) becomes

$$\begin{aligned}
u_{\mathrm{c}1} =& k_{\mathrm{c}0}(\omega_{\mathrm{c}2}^2+\omega_{\mathrm{c}3}^2+\omega_{\mathrm{c}4}^2),\\
u_{\mathrm{c}2} =& k_{\mathrm{c}0}\omega_{\mathrm{c}3}^2,\\
u_{\mathrm{c}3} =& k_{\mathrm{c}0}(\omega_{\mathrm{c}4}^2-\omega_{\mathrm{c}2}^2),\\
u_{\mathrm{c}4} =& k_{\mathrm{c}1}(\omega_{\mathrm{c}3}^2-\omega_{\mathrm{c}2}^2-\omega_{\mathrm{c}4}^2)+k_{\mathrm{c}2}(\omega_{\mathrm{c}3}-\omega_{\mathrm{c}2}-\omega_{\mathrm{c}4}),
\end{aligned} \tag{4.14}$$

with $k_{\mathrm{c}0}, k_{\mathrm{c}1}, k_{\mathrm{c}2}$ as constant parameters. As can be seen from Eq. (4.14), $\omega_{\mathrm{c}i}$ with $i = 2, 3, 4$ are the independent control quantities. Denote $\mathbf{z}_{\mathrm{c}} = [\omega_{\mathrm{c}2}, \omega_{\mathrm{c}3}, \omega_{\mathrm{c}4}]^T$.

Next we need to identify the equilibrium point and linearise the model. Consider the equilibrium state

$$\mathbf{x}_{\mathrm{c}}^* = \left[x_{\mathrm{c}}^*, y_{\mathrm{c}}^*, z_{\mathrm{c}}^*, 0, 0, \psi_{\mathrm{c}}^*, 0, 0, 0, 0, 0, 0\right]^T.$$

It can be calculated that, in order to stay at the equilibrium state, the control quantities should be

$$\mathbf{z}_{\mathrm{c}}^* = [\omega_{\mathrm{c}2}^*, \omega_{\mathrm{c}3}^*, \omega_{\mathrm{c}4}^*] = \left[\frac{mg}{2k_{\mathrm{c}0}}, 0, \frac{mg}{2k_{\mathrm{c}0}}\right].$$

Let $\bar{\mathbf{z}}_{\mathrm{c}} = \mathbf{z}_{\mathrm{c}} - \mathbf{z}_{\mathrm{c}}^*$ and $\bar{\mathbf{x}}_{\mathrm{c}} = \mathbf{x}_{\mathrm{c}} - \mathbf{x}_{\mathrm{c}}^*$. The linearised dynamical equation is

$$\dot{\bar{\mathbf{x}}}_{\mathrm{c}} = \mathbf{A}_{\mathrm{c}}\bar{\mathbf{x}}_{\mathrm{c}} + \mathbf{B}_{\mathrm{c}}\bar{\mathbf{z}}_{\mathrm{c}}, \tag{4.15}$$

where

$$\mathbf{A}_\mathrm{c} = \left.\frac{\partial \mathbf{F}_\mathrm{c}(\mathbf{x}_\mathrm{c}, \mathbf{z}_\mathrm{c})}{\partial \mathbf{x}_\mathrm{c}}\right|_{\mathbf{x}_\mathrm{c}=\mathbf{x}_\mathrm{c}^*, \mathbf{z}_\mathrm{c}=\mathbf{z}_\mathrm{c}^*}$$

$$= \begin{bmatrix} 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & \mathbf{I}_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & \mathbf{I}_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & \mathbf{E}_{\mathrm{c}3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \in \mathbb{R}^{12\times12},$$

$$\mathbf{E}_{\mathrm{c}3\times3} = \begin{bmatrix} g\sin\psi_\mathrm{c}^* & g\cos\psi_\mathrm{c}^* & 0 \\ -g\cos\psi_\mathrm{c}^* & g\sin\psi_\mathrm{c}^* & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

and

$$\mathbf{B}_\mathrm{c} = \left.\frac{\partial \mathbf{F}_\mathrm{c}(\mathbf{x}_\mathrm{c}, \mathbf{z}_\mathrm{c})}{\partial \mathbf{z}_\mathrm{c}}\right|_{\mathbf{x}_\mathrm{c}=\mathbf{x}_\mathrm{c}^*, \mathbf{z}_\mathrm{c}=\mathbf{z}_\mathrm{c}^*}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & k_{\mathrm{c}0}\omega_{\mathrm{c}3}^*\frac{l}{I_x} & 0 \\ -k_{\mathrm{c}0}\omega_{\mathrm{c}2}^*\frac{l}{I_y} & 0 & k_{\mathrm{c}0}\omega_{\mathrm{c}4}^*\frac{l}{I_y} \\ r_2 & -r_3 & -r_4 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{2k_{\mathrm{c}0}}{m}\omega_{\mathrm{c}2}^* & \frac{2k_{\mathrm{c}0}}{m}\omega_{\mathrm{c}3}^* & \frac{2k_{\mathrm{c}0}}{m}\omega_{\mathrm{c}4}^* \end{bmatrix} \in \mathbb{R}^{12\times3},$$

$$r_i = -2k_{\mathrm{c}1}\omega_{\mathrm{c}i}^* + k_{\mathrm{c}2}\frac{1}{I_z},$$

with $\mathbf{I}_{3\times3}$ as identity matrix.

Denote $k_{\mathrm{c}0}l\omega_{\mathrm{c}3}^*/I_x = c_1$, $-k_{\mathrm{c}0}l\omega_{\mathrm{c}2}^*/I_y = c_2$, $r_2 = r_4 = c_3$ and $2k_{\mathrm{c}0}\omega_{\mathrm{c}2}^*/m = c_4$. Then, the controllability matrix $\mathbf{Q}_{\mathrm{cc}}$ is given in Eq. (4.16). It can be counted that the rank of controllability matrix is $\mathrm{rank}(\mathbf{Q}_{\mathrm{cc}}) = 8$. Since the full row rank of $\mathbf{Q}_{\mathrm{cc}}$ is 12, the linearised system is not fully controllable and there are 4 uncontrollable modes.

$$
\mathbf{Q}_{\mathrm{cc}} = [\mathbf{B}\ \mathbf{AB}\ \mathbf{A}^2\mathbf{B}\ ...\ \mathbf{A}^{11}\mathbf{B}]
$$
$$
= \begin{bmatrix} \begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_2 g & 0 & c_2 g \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & c_4 & 0 & c_4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -c_2 & 0 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & c_3 & \frac{k_{\mathrm{c}2}}{I_z} & -c_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-c_2 & 0 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
c_3 & \frac{k_{\mathrm{c}2}}{I_z} & -c_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -c_2 g & 0 & c_2 g & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
c_4 & 0 & c_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix} & \mathbf{0}_{12\times 24} \end{bmatrix}. \tag{4.16}
$$

Next, we conduct controllability decomposition to identify the uncontrollable states. Define a new state as $\tilde{\mathbf{x}}_{\mathrm{c}} = \mathbf{T}_{\mathrm{c}}^{-1}\bar{\mathbf{x}}_{\mathrm{c}}$, where $\mathbf{T}_{\mathrm{c}}$ is a transformation matrix. It follows from the linearised model in Eq. (4.15) that

$$
\dot{\tilde{\mathbf{x}}}_{\mathrm{c}} = \mathbf{T}_{\mathrm{c}}^{-1}\mathbf{A}_{\mathrm{c}}\mathbf{T}_{\mathrm{c}}\tilde{\mathbf{x}}_{\mathrm{c}} + \mathbf{T}_{\mathrm{c}}^{-1}\mathbf{B}_{\mathrm{c}}\bar{\mathbf{z}}_{\mathrm{c}}. \tag{4.17}
$$

Following the controllability decomposition procedure [167], take 10 linearly independent columns of Eq. (4.16) and add four custom columns to make $\mathbf{T}_{\mathrm{c}}$ nonsingular, as in Eq. (4.18). Then, the new state could be partitioned into controllable component

$$
\mathbf{T}_{\mathrm{c}} = \left[\begin{array}{cccccccc:cccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_2 g & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & c_4 & 0 & c_4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & -c_2 & 0 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & c_3 & \frac{k_{\mathrm{c}2}}{I_z} & -c_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
-c_2 & 0 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
c_3 & \frac{k_{\mathrm{c}2}}{I_z} & -c_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -c_2 g & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
c_4 & 0 & c_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]. \tag{4.18}
$$

$\widetilde{\mathbf{x}}_c$ and uncontrollable component $\widetilde{\mathbf{x}}_{uc}$, and Eq. (4.17) becomes

$$\begin{bmatrix} \widetilde{\mathbf{x}}_c \\ \hline \widetilde{\mathbf{x}}_{uc} \end{bmatrix}' = \left[\begin{array}{c|c} \mathbf{A}_c & \mathbf{A}_{12} \\ \hline \mathbf{0} & \mathbf{A}_{uc} \end{array}\right] \begin{bmatrix} \widetilde{\mathbf{x}}_c \\ \hline \widetilde{\mathbf{x}}_{uc} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_c \\ \hline \mathbf{0} \end{bmatrix} \bar{\mathbf{z}}_c. \tag{4.19}$$

Substituting Eq. (4.18) into $\widetilde{\mathbf{x}}_c = \mathbf{T}_c^{-1}\bar{\mathbf{x}}_c$ yields

$$\widetilde{\mathbf{x}}_c = \begin{bmatrix} \frac{x_{12}m}{2c_4} - \frac{x_8}{2c_2} \\ \frac{x_9}{c_5} + \frac{c_3x_8}{c_2c_5} \\ \frac{x_8}{2c_2} + \frac{x_{12}}{2c_4} \\ \frac{x_3}{2c_4} - \frac{x_5}{2c_2} \\ \frac{x_6}{c_5} + \frac{c_3x_5}{c_2c_5} \\ \frac{x_5}{2c_2} + \frac{x_3}{2c_4} \\ -\frac{x_{10}}{c_2g} \\ -\frac{x_1}{c_2g} \\ \hdashline x_2 \\ x_4 \\ x_7 \\ x_{11} \end{bmatrix}.$$

According to Eq. (4.19), the last four elements of $\widetilde{\mathbf{x}}_c$ correspond to uncontrollable modes. Hence x_2, x_4, x_7, and x_{11} are uncontrollable. Here, x_4 and x_7 represent ϕ and p, respectively; and x_2 and x_{11} represent y and v, respectively.

4.4 Controller Design and Simulation Validation

4.4.1 Controller Design

Unlike conventional quadcopters, a VPP quadcopter remains controllable in the presence of one propeller failure. Therefore, simple LQR controllers could be designed based on the linearised model derived in the preceding sections. This is an advantage of VPP quadcopters compared to conventional ones.

4.4.2 Simulation Examples

We next present three simulation examples to verify the effectiveness of the proposed controller design. In the simulation, suppose propeller 1 fails to work and hence it gives zero thrust force and zero torque. Since there will be no manoeuvrer actions within the whole process, the quadcopter dynamic system is built based on the Euler model. In the simulation, we model the actuator and speed control of a VPP as a first-order transfer function to approximate their dynamics. The cut-off frequency is chosen as 10 Hz.

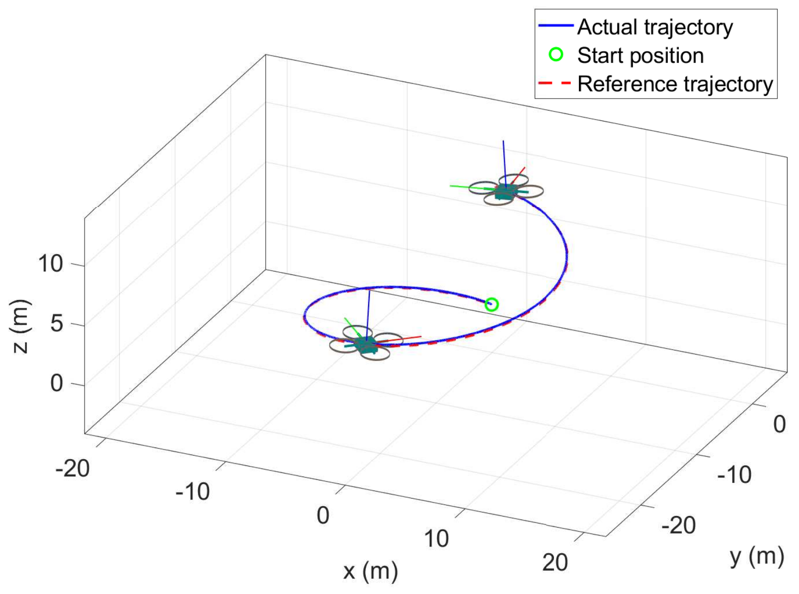


Figure 4.1. 3D trajectory in scenario 1.

Scenario 1: No Noise nor Disturbance

In this simulation scenario, suppose all the states could be measured perfectly and there are no external disturbances. The trajectory reference is a continuous circle together with altitude increasing and orientation varying.

Simulation results are shown in Figs. 4.1 and 4.2. As can be seen, the quadcopter remains fully controllable through one propeller fails. Here, the reference tracking is achieved by tracking a moving target point with desired position and altitude.

Scenario 2: Measurement with Noise

In scenario 2, the quadcopter needs to track the target points varying from $[0, 0, 0]$, $[0, 0, 5]$, $[5, 0, 5]$, and finally to $[5, 5, 5]$ at time $t = 0, 2, 5, 10$, respectively. The desired yaw angle changes from zero to 0.1 rad at $t = 1$. During the whole process, noises with the signal-to-noise ratio as 15 dB are added to all the feedback states.

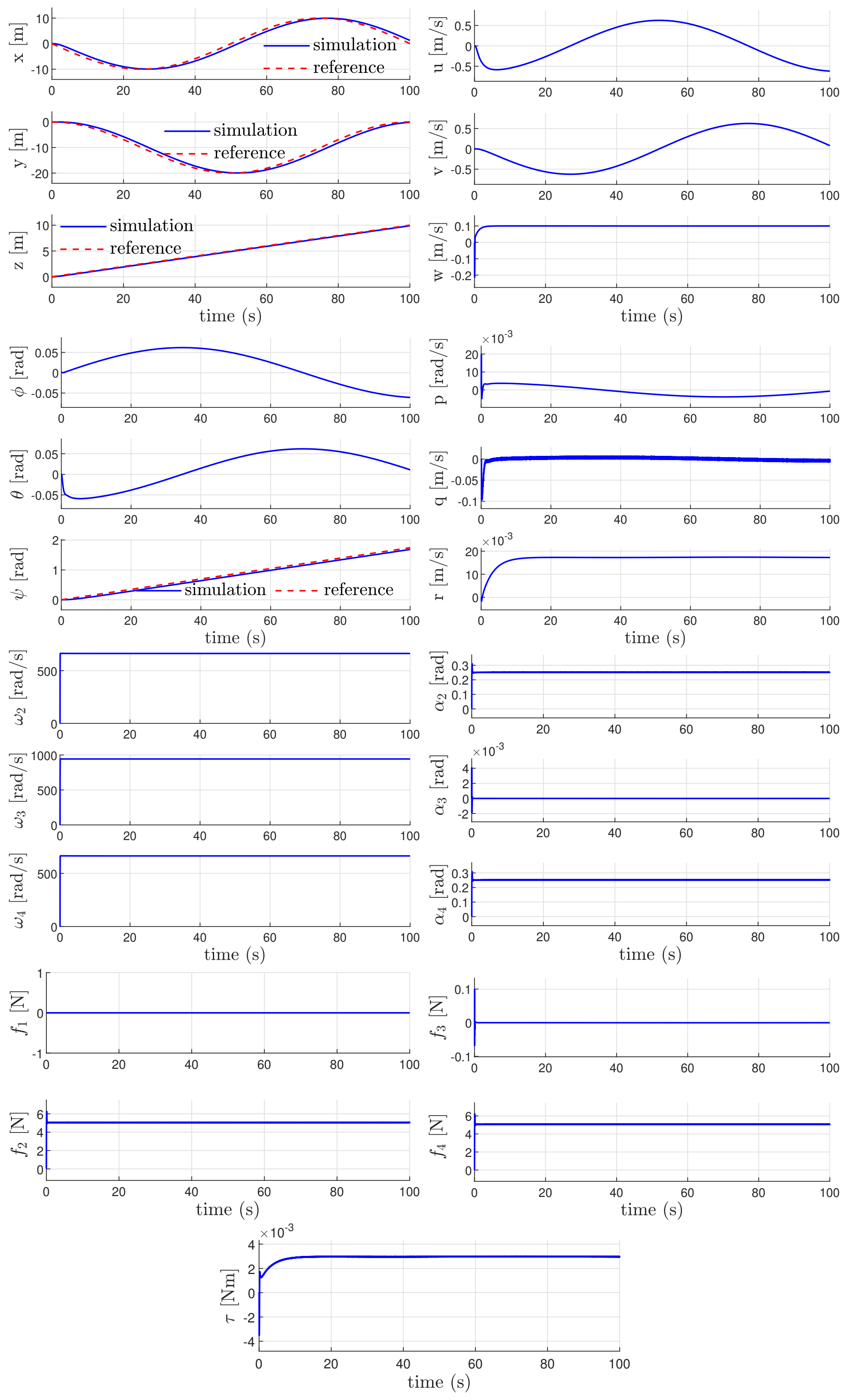


Figure 4.2. Simulation results for scenario 1.

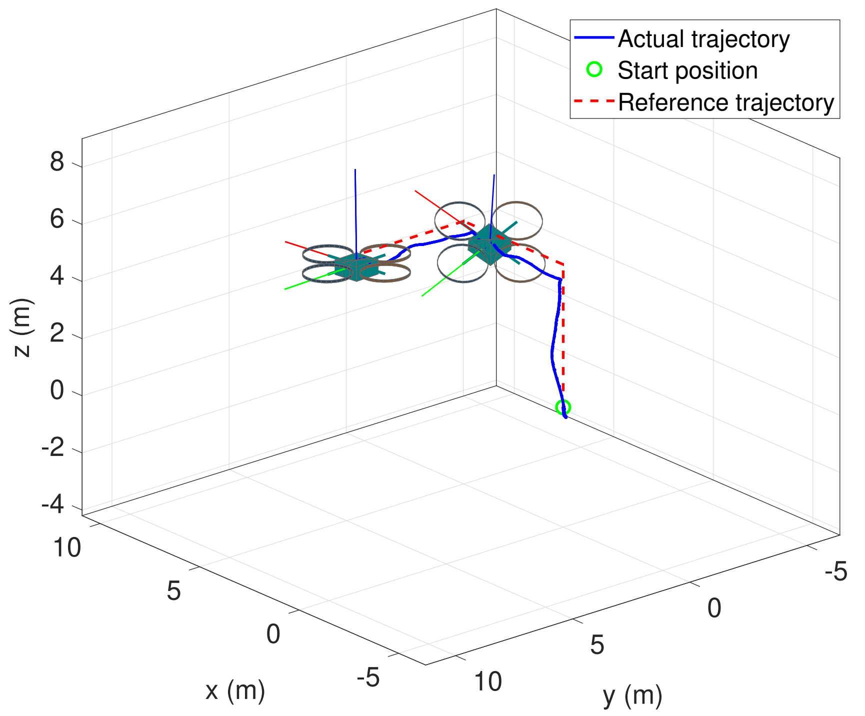


Figure 4.3. 3D trajectory in scenario 2.

Figures 4.3 and 4.4 show the tracking performance of the VPP quadcopter. The quadcopter can quickly track to the reference in the presence of noises.

Scenario 3: Noise and External Disturbance

In scenario 3, we suppose all the state measurements are corrupted by 15 dB signal-to-noise-ratio noises. More importantly, wind disturbance is considered. In particular, the wind disturbance is 4 m/s along x-axis and 4 m/s along y-axis when $t \geq 1$. The desired states are all set to 0.

Figure 4.5 presents the reaction of a VPP quadcopter facing an external disturbance and measurement noises. As can be seen in Fig. 4.6, the pitch and roll angles of the quadcopter converge to constant non-zero values to counter the wind disturbance.

4.5 Summary

In this chapter, we study the control of an SVPP quadcopter in the presence of a propeller failure. If one VPP is faulty, an SVPP quadcopter still has six independent control inputs (i.e., three motor spinning speeds and three pitch angles),

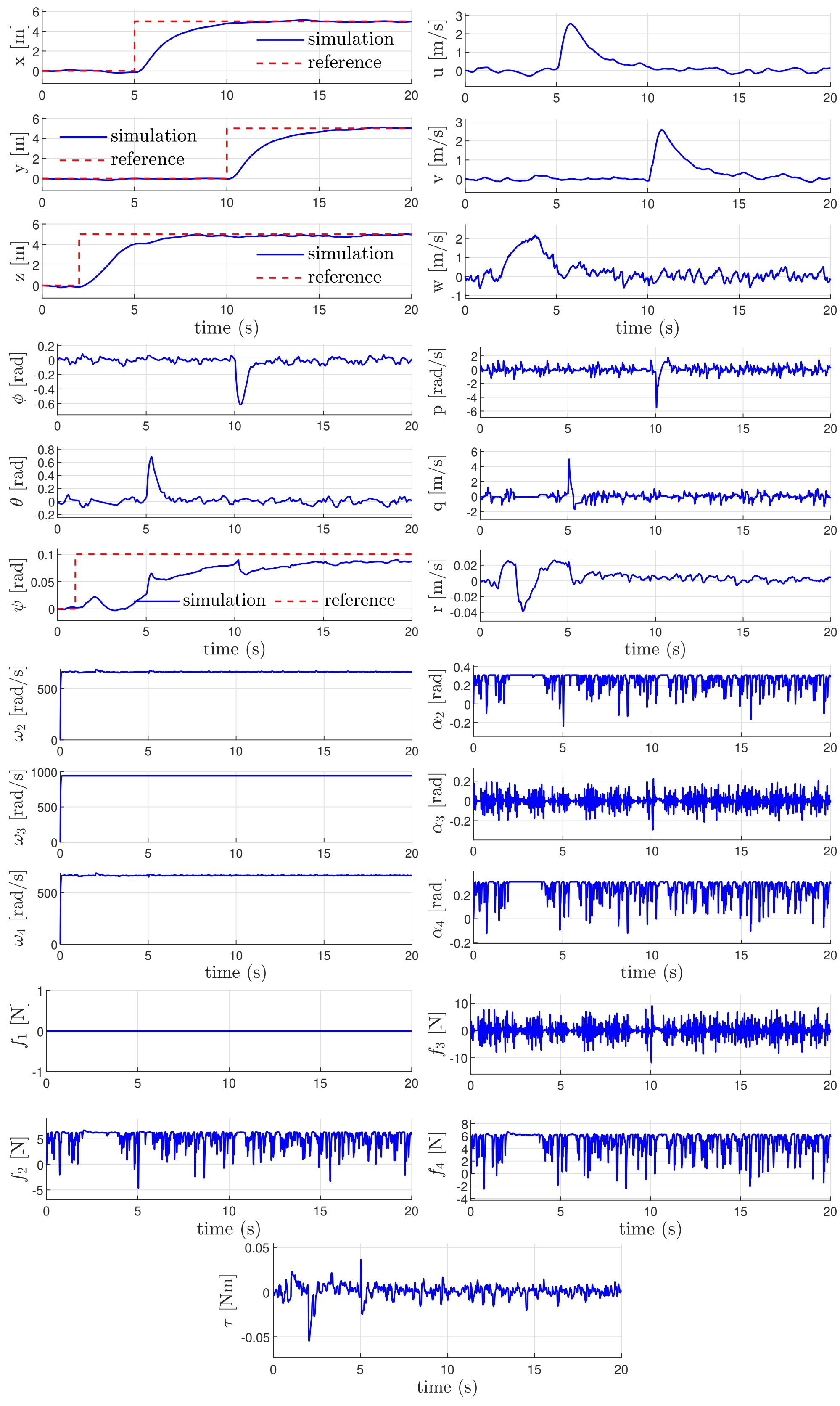


Figure 4.4. Simulation results for scenario 2.

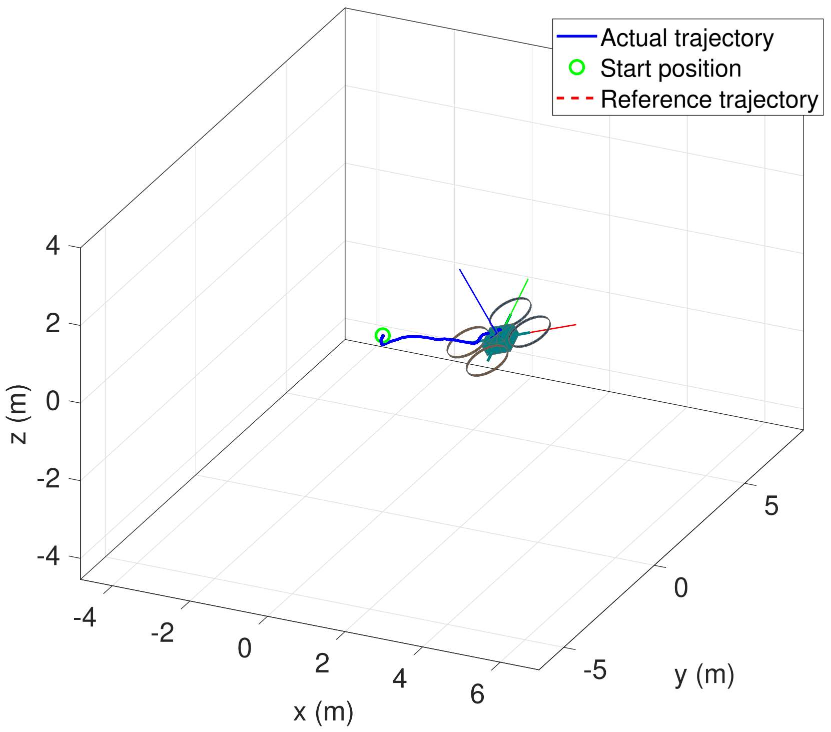


Figure 4.5. 3D trajectory in scenario 3.

and the entire system remains controllable. Although the controllability analysis and controller design are both based on the linearised model, numerical simulation incorporating external disturbances and measurement noise has verified that the theoretical findings are still valid for the nonlinear dynamical model. It is suggested that VPP quadcopters provide a promising platform for various aerial tasks.

The contributions of this chapter are as follows. First, we conduct the controllability analysis of a VPP quadcopter with one propeller failure based on the linearised dynamical model. It is shown that, when one propeller fails, all states of the quadcopter remain controllable. For comparison purposes, we also analyse the controllability of fixed-pitch quadcopters with one rotor failure and show that fixed-pitch quadcopters are uncontrollable with rotor failures. Second, we identify the equilibrium point and derive the linearised dynamical model of VPP quadcopters. Based on the linearised model, we design an LQR controller to handle propeller failure. Based on such a simple controller, the VPP quadcopter could accurately track a given trajectory subject to wind disturbances or noises. This property is important to safety-critical tasks such as flying taxis, where it is necessary to transport the human passengers safely back to a safe place in the presence of propeller failures. Thanks to this property, SVPP quadcopters provide an important alternative type of platform for safety-critical aerial tasks.

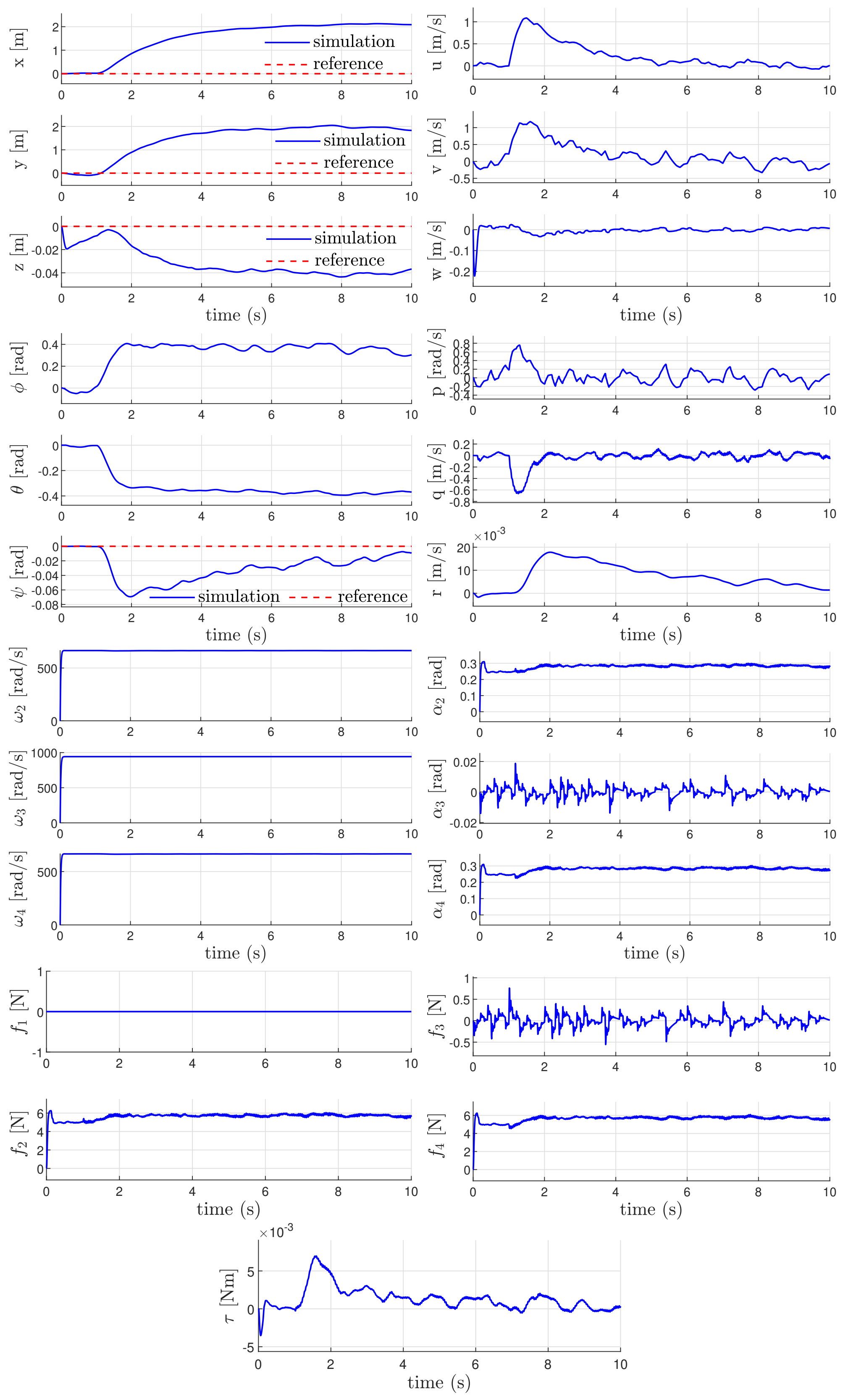


Figure 4.6. Simulation results for scenario 3.

Chapter 5

Control of Centralized-Powered Variable Pitch Propeller Quadcopters Subject to Propeller Faults

5.1 Introduction

According to the review of the VPP quadcopter structure, we could find that the CVPP quadcopter is another popular type of VPP quadcopter. It is a generalised variant of the SVPP quadcopter, which retains the feature of the VPP actuator that can realize the inverse thrust, and integrates all independent power units into a total one. For a CVPP quadcopter, all propellers spin at the same constant speed and they are driven by the same motor located at the centre of the body [24, 168]. Therefore, a CVPP quadcopter will still be able to fly upside down steadily and recover swiftly from largely disturbed attitudes back to stable hovering conditions, which is important considering the safety of the system. In addition, a CVPP quadcopter equipped with a fuel engine can lift the energy consumption constraints caused by the battery.

Although SVPP and CVPP quadcopters have similar mechanism actuator structures, their dynamic inputs are quite different. As we have presented in Chapter 4, if one actuator is faulty, an SVPP quadcopter still has six independent control inputs (i.e., three motor spinning speeds and three pitch angles), and the entire system remains controllable. By contrast, if one propeller of a CVPP quadcopter fails, there would only be three independent control inputs, and the entire system becomes uncontrollable, so it is more challenging to deal with than the SVPP case is. But according to our analysis shown in Chapter 3, it still exhibits great potential for safe and agile flight in future applications. This research has not been studied in the literature.

In this chapter, we present the flight control of a CVPP quadcopter in the presence of a propeller fault. We analyse the equilibrium trajectory, identify the uncontrollable modes, and propose a linear controller. It is shown that the yaw angle and angular velocity become uncontrollable in the presence of a VPP fault, yet the quadcopter can still accurately track a desired trajectory. It is also discovered that the quadcopter exhibits different and favourable behaviour due to its structure, such as slow self-spinning speed. We also analyse the relationship under certain parameter conditions and identify the parameter conditions that lead to zero self-spinning. Our analysis could contribute to the development of high-performance quadcopters that are both agile and robust with respect to faults. Simulation results are presented to verify the theoretical findings.

5.2 Problem Setup

We consider a CVPP quadcopter with a “plus” configuration where propellers 1 and 3 rotate clockwise and propellers 2 and 4 rotate counter-clockwise. In this chapter, fault detection or isolation is not considered, which is, however, important for the research. We consider the faulty scenario where the pitch angle of propeller 1 becomes stuck at a constant value c. Substituting $\alpha_1 = c$ to Eq. (3.8) gives

$$u_1 = b_0(c + \alpha_2 + \alpha_3 + \alpha_4), \tag{5.1}$$

$$u_2 = b_0(\alpha_3 - c), \tag{5.2}$$

$$u_3 = b_0(\alpha_4 - \alpha_2), \tag{5.3}$$

$$u_4 = b_1(-c^2 + \alpha_2^2 - \alpha_3^2 + \alpha_4^2) + b_2(-c + \alpha_2 - \alpha_3 + \alpha_4). \tag{5.4}$$

Table 5.1. CVPP quadcopter parameters.

Parameters	Name	Value	Unit
Mass	m	1	kg
Gravitational acceleration	g	9.8	m/s^2
Inertia about x_b, y_b	I_x,I_y	0.125	kg·m^2
Inertia about z_b	I_z	0.25	kg·m^2
Arm length	l	1	m
Force coefficient	K_0	$2.86 \cdot 10^{-5}$	-
Torque coefficient	K_1	$1 \cdot 10^{-8}$	-
	K_2	$6.56 \cdot 10^{-7}$	-
	K_3	$2.29 \cdot 10^{-5}$	-
Pitch angle limit	α_{sat}	0.3142	rad
Pitch angle rate limit	d_α	11.62	rad/s
Motor speed limit	$\omega_{\max}$	1047.19	rad/s
Motor speed rate limit	d_ω	6047	rad/s^2
Linear resistance coefficient	K_β	0.01	N·s/m
Rotation resistance coefficient	K_γ	0.05	N·s·m

In this case, u_1, u_2, u_3, u_4 are not independent anymore since there are merely three degrees of freedom.

The overall dynamic model for a CVPP quadcopter is identical to that of a fixed-pitch propeller quadcopter. The translational motion of the quadcopter is given by Eq. (3.12) and the rotational motion of the quadcopter is described by Eq. (3.16).

The parameter values are shown in Table 5.1.

5.3 Equilibrium Analysis

In this section, we first identify the equilibrium of the nonlinear faulty system, and then present a quantitative study of the equilibrium points which reveals some interesting and unique properties of CVPP quadcopters.

5.3.1 Equilibrium Trajectory

In the presence of a faulty actuator, the CVPP quadcopter would not be able to remain at any static equilibrium point. However, it is able to remain along a family of equilibrium points. In particular, consider a state trajectory

$$\begin{aligned}\mathbf{x}^*(t) &= [x^*, y^*, z^*, \phi^*, \theta^*, \psi^*(t), p^*, q^*, r^*, u^*, v^*, w^*]^T \\ &= [x^*, y^*, z^*, 0, 0, \psi^*(t), 0, 0, r^*, 0, 0, 0]^T,\end{aligned}$$

where x^*, y^*, z^*, r^* are constant values and $\psi^*(t) = r^*t + \psi_0^*$. The state trajectory $\mathbf{x}^*(t)$ corresponds to the case where the yaw angle $\psi^*(t)$ varies at a constant value r^* while all the other states are constant.

Along $\mathbf{x}^*(t)$, we have $\dot{u}^*, \dot{v}^*, \dot{w}^*, \dot{p}^*$, and $\dot{q}^*$ equal to 0. It then follows from Eqs. (3.12) and (3.16) that $u_1^* = mg$, $u_2^* = 0$, and $u_3^* = 0$, substituting which into Eqs. (5.1)-(5.3) gives

$$\alpha_2^* = \alpha_4^* = \frac{mg}{2b_0} - c, \quad \alpha_3^* = c. \tag{5.5}$$

Furthermore, substituting $\alpha_2^*, \alpha_3^*, \alpha_4^*$ into Eq. (5.4) yields

$$u_4^* = 2b_2\left[\left(\frac{mg}{2b_0} - c\right)^2 - c^2\right] + 2b_3\left[\left(\frac{mg}{2b_0} - c\right) - c\right]. \tag{5.6}$$

Moreover, substitute $\dot{r}^* = 0$ to Eq. (3.16) gives

$$r^* = sgn(u_4^*)\sqrt{\frac{|u_4^*|l}{K_\gamma}}, \tag{5.7}$$

where sgn(·) denotes the sign function.

In terms of control inputs, since there are only three functional servos, we choose the three pitch angles as the new inputs. Along the equilibrium trajectory, we have

$$\begin{aligned}\mathbf{u}_F^* &= [\alpha_2^*, \alpha_3^*, \alpha_4^*]^T \\ &= \left[\frac{mg}{2b_0} - c, c, \frac{mg}{2b_0} - c\right]^T,\end{aligned}$$

where the subscript F represents 'fault'.

5.3.2 Quantitative Study of the Equilibrium

Along the equilibrium trajectory, the quadcopter would rotate at a constant rate of r^* as given in Eq. (5.7). It is of great interest to see the specific values of r^*, especially whether r^* could be zero.

Substituting $b_0 = K_0\omega_0^2$, $b_2 = K_1\omega_0^2$, and $b_2 = K_2\omega_0$ into Eq. (5.6) gives

$$u_4^* = \frac{K_2 m^2 g^2}{2K_1^2\omega_0^2} + \frac{K_3 mg}{K_1\omega_0} - \left(\frac{2mgK_2}{K_1} + 4K_3\omega_0\right)c, \tag{5.8}$$

substituting which into Eq. (5.7) yields

$$r^* = \sqrt{\left[\frac{K_2 m^2 g^2}{2K_1^2 \omega_0^2} + \frac{K_3 mg}{K_1 \omega_0} - \left(\frac{2mgK_2}{K_1} + 4K_3\omega_0\right) c\right] \frac{l}{k_\gamma}}. \tag{5.9}$$

Although we assume ω_0 is fixed in each control process, we would like to analyse the impact of different values of ω_0 on r^*. First, suppose that the physically achievable interval of ω_0 is $[0, \omega_{\max}]$. Second, since ω_0 must satisfy Eq. (5.5), we have

$$\omega_0 = \sqrt{\frac{mg}{2K_1(\alpha_2^* + c)}}. \tag{5.10}$$

Suppose the physically achievable pitch angle is saturated by $[-\alpha_{\text{sat}}, \alpha_{\text{sat}}]$, where $\alpha_{\text{sat}} = 0.314$ rad following [20]. Since $\alpha_2^* > -c$ by Eq. (5.5), we know $\alpha_2 \in (-c, \alpha_{\text{sat}}]$. It then follows from Eq. (5.10) that $\omega_0 \in \Omega_2 = \left[\sqrt{\frac{mg}{2K_1(\alpha_{\text{sat}}+c)}}, +\infty\right)$. Thus, ω_0 must satisfy

$$\begin{aligned} \omega_0 \in \Omega_1 \cap \Omega_2 &= \left[\sqrt{\frac{mg}{2K_1(\alpha_{\text{sat}} + c)}}, \omega_{\max}\right] \\ &:= [\omega_{\min}, \omega_{\max}]. \end{aligned} \tag{5.11}$$

To determine the admissible interval of c, we note that Eq. (5.11) implies

$$0 \leq \frac{mg}{2K_1(\alpha_{\text{sat}} + c)} \leq \omega_{\max}^2,$$

which gives $c \geq \frac{mg}{2K_1\omega_{\max}^2} - \alpha_{\text{sat}}$. Furthermore, since $c \in [-\alpha_{\text{sat}}, \alpha_{\text{sat}}]$, we have

$$c_{\min} := \frac{mg}{2K_1\omega_{\max}^2} - \alpha_{\text{sat}} \leq c \leq \alpha_{\text{sat}}. \tag{5.12}$$

The interpretation is that, if the faulty propeller became stuck at the angle $c < c_{\min}$, then Eq. (5.10) could not hold and hence the system could not reach the equilibrium point.

When c varies in $[c_{\min}, \alpha_{\text{sat}}]$, the values of r^* against w_0 are shown in Fig. 5.1. This figure suggests two important properties. First, the values of r^* for different c are within a small interval: $[-0.13, 0.13]$ rad/s. As a result, the quadcopter self-rotates slowly at the equilibrium. Second, surprisingly, we could tune w_0 to make $r^* = 0$ for certain c (see blue dotted curves in Fig. 5.1). In this case, a CVPP quadcopter could hover steadily without self-rotation in the presence of a propeller fault.

To examine the case of $r^* = 0$ more closely, substituting $r^* = 0$ into Eq. (5.9) gives

$$\frac{K_2 m^2 g^2}{2K_1^2\omega_0^2} + \frac{K_3 mg}{K_1\omega_0} = \left(\frac{2mgK_2}{K_1} + 4K_3\omega_0\right)c, \tag{5.13}$$

which implies

$$c = \frac{mg}{4K_1\omega_0^2} \tag{5.14}$$

or equivalently

$$\omega_0^* = \sqrt{\frac{mg}{4K_1 c}}. \tag{5.15}$$

Therefore, given a specific value of ω_0, if angle c at which propeller 1 became stuck satisfies Eq. (5.14), then we have $r^* = 0$. On the other hand, given a specific value of c, if ω_0 satisfies Eq. (5.15), then we have $r^* = 0$.

Since $c \in [c_{\min}, \alpha_{\text{sat}}]$ and $\omega_0 \in [\omega_{\min}, \omega_{\max}]$, we need to examine under what conditions Eq. (5.14) holds. Since $\omega_0 \in [\omega_{\min}, \omega_{\max}]$, it follows from Eq. (5.14) that

$$c = \frac{mg}{4K_1\omega_0^2} \leq \frac{mg}{4K_1\omega_{\min}^2} = \alpha_{\text{sat}}, \tag{5.16}$$

$$c = \frac{mg}{4K_1\omega_0^2} \geq \frac{mg}{4K_1\omega_{\max}^2} := c^\dagger \approx 0.05. \tag{5.17}$$

Inequality Eq. (5.16) always holds because $c \leq \alpha_{\text{sat}}$. Then, it follows from Eq. (5.17) that if

$$c \geq c^\dagger,$$

then r^* could be zero by setting ω_0 as in Eq. (5.15). This result is illustrated by Fig. 5.1, when $c \geq c^\dagger$ (blue dotted curves), there exists ω_0 that could achieve $r^* = 0$.

5.3.3 Linearised Model

Define $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^*$ and $\bar{\mathbf{u}}_F = \mathbf{u}_F - \mathbf{u}_F^*$. Since $\dot{\mathbf{x}}^* = \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*)$, we have

$$\begin{aligned}\dot{\bar{\mathbf{x}}} = \dot{\mathbf{x}} - \dot{\mathbf{x}}^* &= \mathbf{f}(\bar{\mathbf{x}} + \mathbf{x}^*, \bar{\mathbf{u}}_F + \mathbf{u}_F^*) - \dot{\mathbf{x}}^* \\ &\approx \mathbf{f}(\mathbf{x}^*, \mathbf{u}_F^*) + \mathbf{A}(\psi^*)\bar{\mathbf{x}} + \mathbf{B}\bar{\mathbf{u}}_F - \dot{\mathbf{x}}^* \\ &= \mathbf{A}(\psi^*)\bar{\mathbf{x}} + \mathbf{B}\bar{\mathbf{u}}_F\end{aligned} \tag{5.18}$$

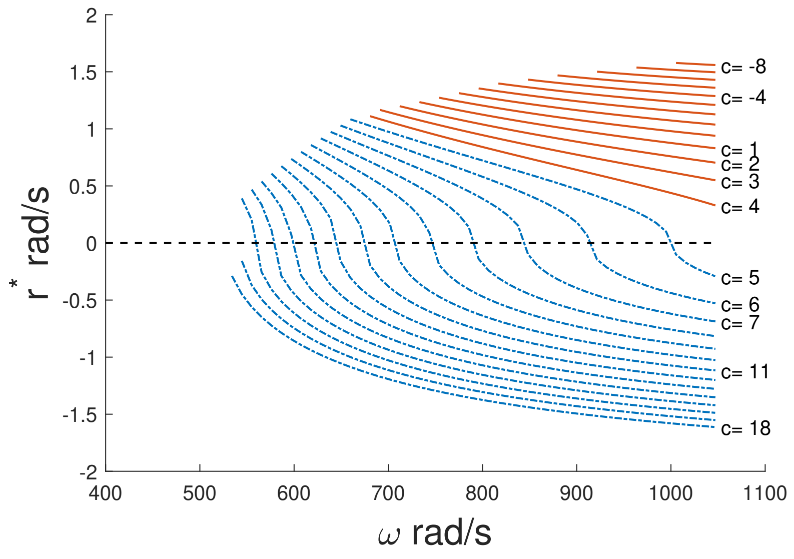


Figure 5.1. The relationship between a VPP quadcopter's self rotation speed r^* and its centrally-controlled propeller spinning speed w_0, as expressed by Eq. (5.9). Shown for different angles c (in degree) at which faulty propeller 1 became stuck. The interval of ω_0 for each curve is given in Eq. (5.11).

where

$$\begin{aligned}\mathbf{A}(\psi^*) &= \left.\frac{\partial \mathbf{F}(\bar{\mathbf{x}},\bar{\mathbf{u}}_F)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*,\mathbf{u}_F=\mathbf{u}_F^*}\\ &= \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{E}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\end{bmatrix} \in \mathbb{R}^{12\times12},\end{aligned}$$

with $\mathbf{I}_{3\times3}$ as the identity matrix and

$$\mathbf{E}_{3\times3} = \begin{bmatrix} g\sin\psi^*(t) & g\cos\psi^*(t) & 0\\ -g\cos\psi^*(t) & g\sin\psi^*(t) & 0\\ 0 & 0 & 0\end{bmatrix},$$

and

$$\begin{aligned}\mathbf{B} &= \left.\frac{\partial \mathbf{F}(\bar{\mathbf{x}},\bar{\mathbf{u}}_F)}{\partial \mathbf{u}_F}\right|_{\mathbf{x}=\mathbf{x}^*,\mathbf{u}_F=\mathbf{u}_F^*}\\ &=[\mathbf{0}_{3\times3}, \mathbf{0}_{3\times3}, \mathbf{Q}_{c1}^T, \mathbf{Q}_{c2}^T]^T \in \mathbb{R}^{12\times3}\end{aligned}$$

where

$$\mathbf{Q}_{c1} = \begin{bmatrix} 0 & \frac{b_0 l}{I_x} & 0 \\ -\frac{b_0 l}{I_y} & 0 & \frac{b_0 l}{I_y} \\ \frac{n_2}{I_z} & -\frac{n_3}{I_z} & \frac{n_4}{I_z} \end{bmatrix}, \mathbf{Q}_{c2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{b_0}{m} & \frac{b_0}{m} & \frac{b_0}{m} \end{bmatrix},$$

and $n_i = 2\alpha_i^* b_2 + b_3$.

5.3.4 Controllability Analysis

The controllability matrix $\mathbf{Q}_c$ of system Eq. (5.18) is

$$\begin{aligned} \mathbf{Q}_c &= \left[\mathbf{B}\ \mathbf{AB}\ \mathbf{A}^2\mathbf{B}\ \dots\ \mathbf{A}^{11}\mathbf{B}\right] \\ &= \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{Q}_{c2} & \mathbf{0}_{3\times3} & \mathbf{Q}_{c3} & \\ \mathbf{0}_{3\times3} & \mathbf{Q}_{c1} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{12\times24} \\ \mathbf{Q}_{c1} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \\ \mathbf{Q}_{c2} & \mathbf{0}_{3\times3} & \mathbf{Q}_{c3} & \mathbf{0}_{3\times3} & \end{bmatrix} \in \mathbb{R}^{12\times36}. \end{aligned}$$

where

$$\mathbf{Q}_{c3} = \begin{bmatrix} -\frac{P_c}{I_y} & \frac{P_s}{I_x} & \frac{P_c}{I_y} \\ -\frac{P_s}{I_y} & -\frac{P_c}{I_x} & \frac{P_s}{I_y} \\ 0 & 0 & 0 \end{bmatrix},$$

$$P_c = b_0 l g \cos\psi^*(t), \quad P_s = b_0 l g \sin\psi^*(t).$$

Matrix $\mathbf{Q}_c$ is sparse and highly structured. It can be verified that $\text{rank}(\mathbf{Q}_c) = 10$ when the rated speed $\omega_0 > 0$. Since there are 12 states, we know that two modes are uncontrollable.

To further identify the uncontrollable modes, we conduct a controllability decomposition as follows. Following [167, 169], we design

$$\mathbf{T} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{Q}_{c2} & \mathbf{0}_{3\times2} & \mathbf{Q}_{c4} & \mathbf{0}_{3\times2} \\ \mathbf{0}_{3\times3} & \mathbf{Q}_{c1} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times2} & \mathbf{Q}_{c5} \\ \mathbf{Q}_{c1} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times2} & \mathbf{Q}_{c6} \\ \mathbf{Q}_{c2} & \mathbf{0}_{3\times3} & \mathbf{Q}_{c4} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times2} \end{bmatrix},$$

$$\mathbf{Q}_{c4} = \begin{bmatrix} -\frac{P_c}{I_y} & \frac{P_s}{I_x} \\ -\frac{P_s}{I_y} & -\frac{P_c}{I_x} \\ 0 & 0 \end{bmatrix},$$

$$\mathbf{Q}_{c5} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{Q}_{c6} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix},$$

where $\mathbf{T} \in \mathbb{R}^{12\times12}$ is invertible. Then we define

$$\tilde{\mathbf{x}} = \mathbf{T}^{-1}\bar{\mathbf{x}}. \tag{5.19}$$

Substituting Eq. (5.19) into Eq. (5.18) leads to $\dot{\tilde{\mathbf{x}}} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\tilde{\mathbf{x}} + \mathbf{T}^{-1}\mathbf{B}\bar{\mathbf{u}}_F$. Let $\tilde{\mathbf{A}} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}$ and $\tilde{\mathbf{B}} = \mathbf{T}^{-1}\mathbf{B}$. Then, we have $\dot{\tilde{\mathbf{x}}} = \tilde{\mathbf{A}}\tilde{\mathbf{x}} + \tilde{\mathbf{B}}\bar{\mathbf{u}}_F$, whose sub-block matrix form is

$$\begin{bmatrix} \dot{\tilde{\mathbf{x}}}_c \\ \dot{\tilde{\mathbf{x}}}_{\bar{c}} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{A}}_c & \tilde{\mathbf{A}}_{12} \\ \mathbf{0}_{2\times10} & \tilde{\mathbf{A}}_{\bar{c}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_c \\ \tilde{\mathbf{x}}_{\bar{c}} \end{bmatrix} + \begin{bmatrix} \tilde{\mathbf{B}}_c \\ \mathbf{0}_{2\times3} \end{bmatrix} \bar{\mathbf{u}}_F, \tag{5.20}$$

where

$$\tilde{\mathbf{A}}_c = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times2} \\ \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times2} \\ \mathbf{0}_{2\times3} & \mathbf{H}_{2\times3} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times3} & \mathbf{0}_{2\times3} & \mathbf{I}_{2\times2} & \mathbf{0}_{2\times2} \end{bmatrix},$$

$$\mathbf{H}_{2\times3} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \quad \tilde{\mathbf{A}}_{12} = \mathbf{0}_{10\times2},$$

$$\tilde{\mathbf{A}}_{\bar{c}} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad \tilde{\mathbf{B}}_c = \begin{bmatrix} \mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} \\ \mathbf{0}_{2\times3} \\ \mathbf{0}_{2\times3} \end{bmatrix}.$$

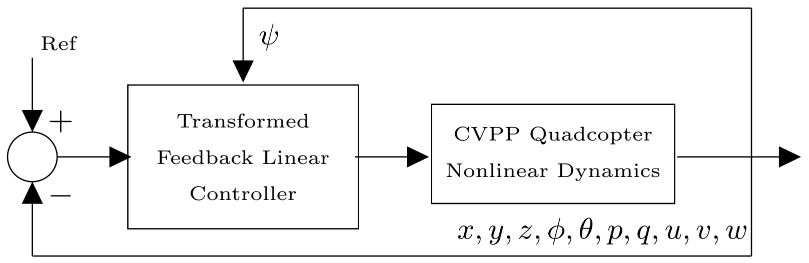

Ref
ψ
$+$
$-$
Transformed Feedback Linear Controller
CVPP Quadcopter Nonlinear Dynamics
$x, y, z, \phi, \theta, p, q, u, v, w$

Figure 5.2. Block diagram of the CVPP quadcopter control scheme.

Equation (5.20) shows that the system is decomposed into an uncontrollable subsystem $\dot{\tilde{\mathbf{x}}}_{\bar{c}} = \tilde{\mathbf{A}}_{\bar{c}}\tilde{\mathbf{x}}_{\bar{c}}$ and a controllable subsystem $\dot{\tilde{\mathbf{x}}}_{c} = \tilde{\mathbf{A}}_{c}\tilde{\mathbf{x}}_{c} + \tilde{\mathbf{A}}_{12}\tilde{\mathbf{x}}_{\bar{c}} + \tilde{\mathbf{B}}_{c}\bar{\mathbf{u}}_{F}$. The transformed states are

$$\begin{bmatrix} \tilde{\mathbf{x}}_{c} \\ \tilde{\mathbf{x}}_{\bar{c}} \end{bmatrix} = \mathbf{T}^{-1}\bar{\mathbf{x}} = \begin{bmatrix} \frac{mx_{12}}{2b_0} - \frac{I_x x_7}{2lb_0} - \frac{I_y x_8}{2lb_0} \\ \frac{I_x x_7}{lb_0} \\ \frac{mx_{12}}{2b_0} - \frac{I_x x_7}{2lb_0} + \frac{I_y x_8}{2lb_0} \\ \frac{mx_3}{2b_0} - \frac{I_x x_4}{2lb_0} - \frac{I_y x_5}{2lb_0} \\ \frac{I_x x_4}{lb_0} \\ \frac{mx_3}{2b_0} - \frac{I_x x_4}{2lb_0} + \frac{I_y x_5}{2lb_0} \\ -\frac{I_y(x_{10}\cos\psi^* + x_{11}\sin\psi^*)}{lb_0 g} \\ -\frac{I_x(x_{11}\cos\psi^* - x_{10}\sin\psi^*)}{lb_0 g} \\ -\frac{I_y(x_1\cos\psi^* + x_2\sin\psi^*)}{lb_0 g} \\ -\frac{I_x(x_2\cos\psi^* - x_1\sin\psi^*)}{lb_0 g} \\ x_9 + x_{12}k_{\bar{c}_1} - x_8 k_{\bar{c}_2} + x_7 k_{\bar{c}_3} \\ x_6 + x_3 k_{\bar{c}_1} - x_5 k_{\bar{c}_2} + x_4 k_{\bar{c}_3} \end{bmatrix}, \tag{5.21}$$

$$k_{\bar{c}_1} = \frac{m(n_2 - n_4)}{2I_z b_0}, \; k_{\bar{c}_2} = \frac{I_y(n_2 + n_4)}{2I_z b_0},$$
$$k_{\bar{c}_3} = \frac{I_x(2n_3 - n_2 + n_4)}{2I_z b_0},$$

where x_i (i ranges from 1 to 12) are the elements of $\bar{\mathbf{x}}$, and $\tilde{\mathbf{x}}_c$ and $\tilde{\mathbf{x}}_{\bar{c}}$ are controllable and uncontrollable modes, respectively.

As can be seen from Eq. (5.21), the uncontrollable states in $\tilde{\mathbf{x}}_{\bar{c}}$, which correspond to the last two elements of the vector, contain x_9 and x_6 that correspond to r and ψ, respectively. As a result, the yaw angle and its angular rate are uncontrollable.

5.3.5 A Linear Controller Design

Figure 5.2 shows a block diagram of the control scheme for a faulty CVPP quadcopter. In this chapter, we present an H_∞ controller to enhance the robustness of the system.

Other control methods could also be used. According to the faulty system Eq. (5.20), the H_∞ controller is designed as

$$\begin{aligned}\dot{\tilde{\mathbf{x}}} &= \tilde{\mathbf{A}}\tilde{\mathbf{x}} + \tilde{\mathbf{B}}_1\mathbf{w} + \tilde{\mathbf{B}}_2\mathbf{u},\\ \mathbf{z}_\infty &= \tilde{\mathbf{C}}_1\tilde{\mathbf{x}} + \mathbf{D}_{12}\mathbf{u},\\ \tilde{\mathbf{y}} &= \tilde{\mathbf{C}}_2\tilde{\mathbf{x}},\end{aligned} \tag{5.22}$$

where

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times2} \\ \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times2} \\ \mathbf{0}_{2\times3} & \mathbf{H}_{2\times3} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times3} & \mathbf{0}_{2\times3} & \mathbf{I}_{2\times2} & \mathbf{0}_{2\times2} \end{bmatrix}, \quad \tilde{\mathbf{B}}_1 = \begin{bmatrix} \mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} \\ \mathbf{0}_{2\times3} \\ \mathbf{0}_{2\times3} \end{bmatrix}, \quad \tilde{\mathbf{B}}_2 = \begin{bmatrix} \mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} \\ \mathbf{0}_{2\times3} \\ \mathbf{0}_{2\times3} \end{bmatrix},$$

$$\tilde{\mathbf{C}}_2 = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{Q}_{c2} & \mathbf{0}_{3\times2} & \mathbf{Q}_{c4} \\ \mathbf{0}_{2\times3} & \mathbf{S} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{S} & \mathbf{0}_{2\times3} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{Q}_{c2} & \mathbf{0}_{3\times3} & \mathbf{Q}_{c4} & \mathbf{0}_{3\times2} \end{bmatrix}, \quad \tilde{\mathbf{C}}_1 = \begin{bmatrix} I_{10\times10} \\ \mathbf{0}_{3\times10} \end{bmatrix}, \quad \mathbf{D}_{12} = \begin{bmatrix} \mathbf{0}_{10\times3} \\ I_{3\times3} \end{bmatrix},$$

$$\mathbf{S} = \begin{bmatrix} 0 & \frac{b_1 l}{I_x} & 0 \\ -\frac{b_1 l}{I_y} & 0 & \frac{b_1 l}{I_y} \end{bmatrix}.$$

It can be found in Eq. (5.22) that, $\tilde{\mathbf{x}}$ is the decomposed state, $\mathbf{w}$ is the disturbance, $\mathbf{u}$ represents the control signal, and $\mathbf{z}_\infty$ is the error value that needs to be minimized.

5.4 Simulation Validation

This section presents two simulation experiments to verify the theoretical findings. The simulation is conducted in a physical environment built in Simscape in Matlab. It combines detailed physical component models based on first principles. We use a low pass filter with a 10 Hz cut-off frequency to represent the dynamics of the propeller actuator. The input of the low pass filter is the expected value of the pitch angle and the output is the response value. Noise with 20 dB signal-to-noise ratio is added to all states as measurement noise.

5.4.1 Scenario 1: Trajectory Tracking

In the first experiment, it is assumed that the propeller 1 has malfunctioned from the beginning with a fixed pitch angle of 0.18 rad and a spinning speed of 707.7 rad/s. Moreover, we assume that these values can be measured and used by the controller.

The quadcopter is required to track a trajectory simulating taking off, moving along a square trajectory with 10 m length on each side, and landing, shown in Fig. 5.3.

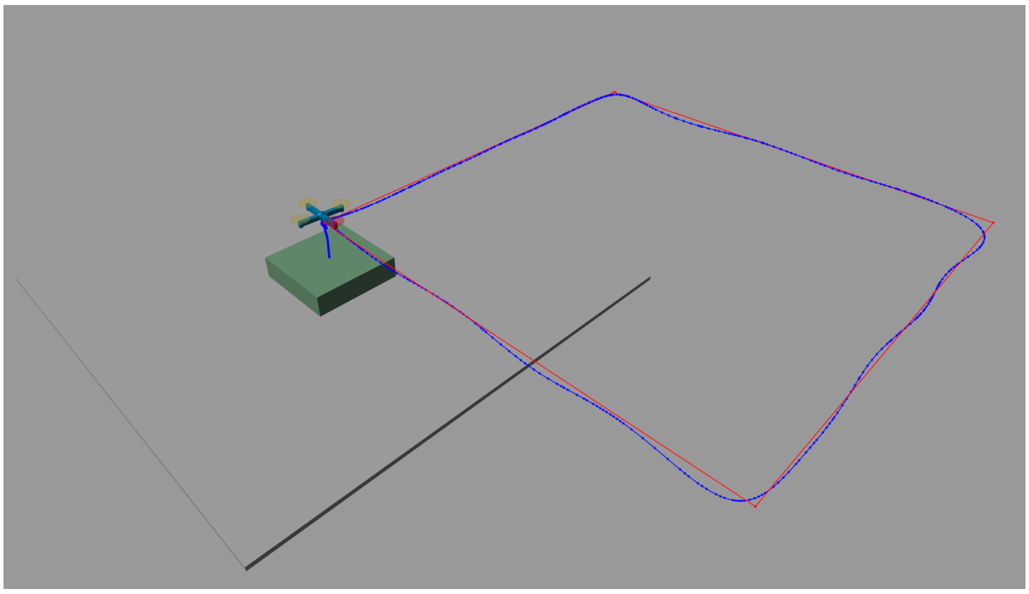


Figure 5.3. Trajectory tracking results. The blue dotted curves represent the actual flying trajectory and the red line is the reference square trajectory with 10 m length on each side. The faulty propeller is highlighted in red.

The results are shown in Fig. 5.4. As can be seen, although the yaw angle and its rate are uncontrollable, a CVPP quadcopter with a faulty actuator is still able to accurately track the desired position trajectory and the value of r remains close to zero, which is consistent with our analysis.

5.4.2 Scenario 2: Random Initialization

In the second experiment, we examine the performance of the proposed controller given a variety of initial conditions. The initial state is $[x_0, y_0, z_0, \phi_0, \theta_0] = [0, 0, 0, \phi_r, \theta_r]$ and the target state is $[x^*, y^*, z^*, \phi^*, \theta^*] = [0, 0, 0, 0, 0]$, where the values of ϕ_r and θ_r are generated randomly from a uniform distribution in [-1.57,1.57] rad. Assume the quadcopter VPP 1 stuck at 0.05 rad. When the state gets sufficiently close to the target state after 5 s, the quadcopter is regarded as stable; otherwise it is unstable.

We conducted 100 trials, each with a different initial condition (ϕ_r and θ_r). Figure 5.5 shows that the controller works well under a wide range of initial conditions, with a success rate of 95 percent. We observe that the quadcopter does not cope well when both pitch (θ) and roll (ϕ) are large negative angles. At this point, the quadcopter needs to be relied upon to generate a high rotation speed and thus stabilize its attitude. However, due to the propeller faults, the angular velocity and

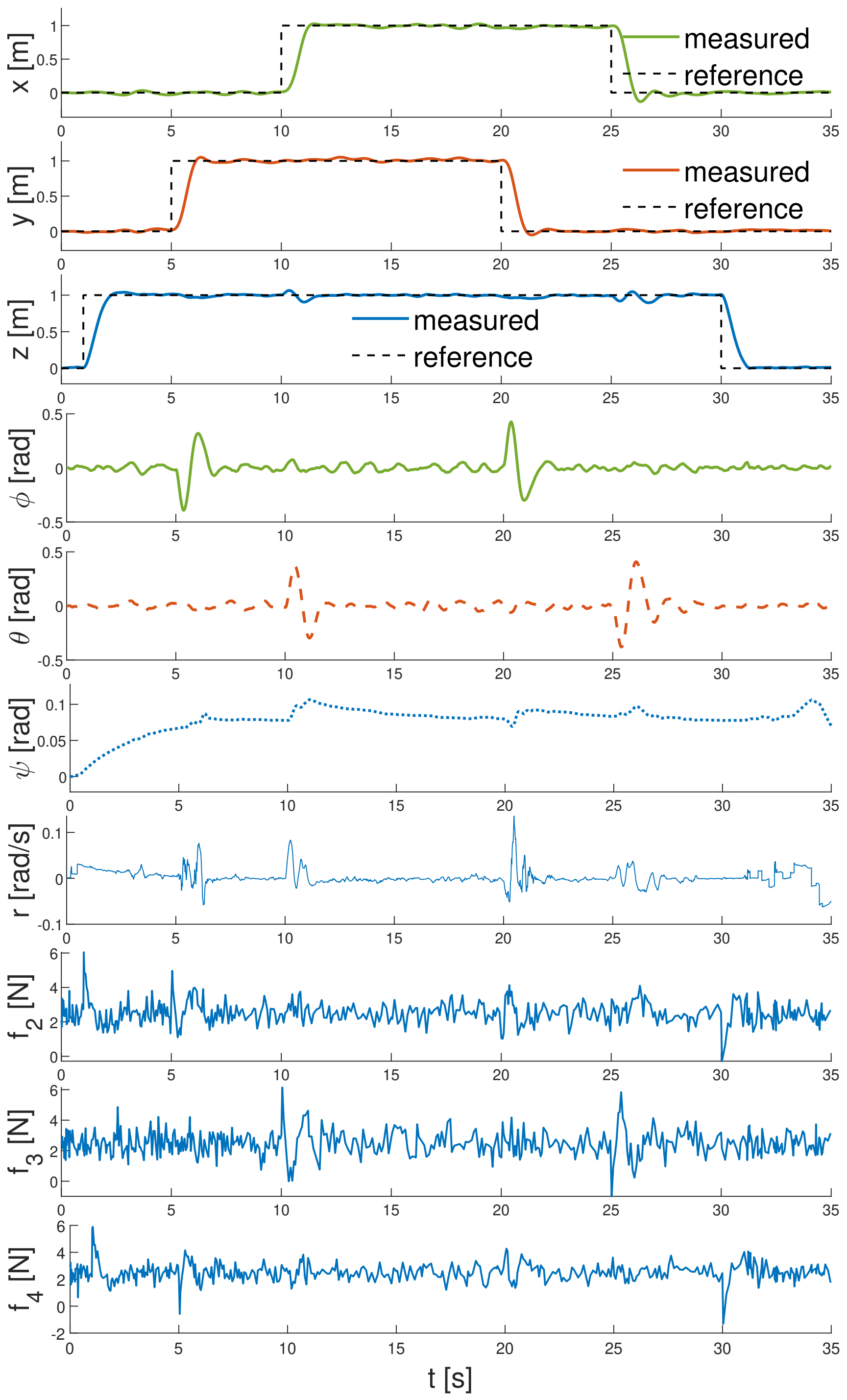


Figure 5.4. Simulation results of a CVPP quadcopter subject to propeller faults. The reference trajectory is defined by points [0,0,1], [0,1,1], [1,1,1], [1,0,1], [0,0,1], [0,0,0] at 1 s, 5 s, 10 s, 20 s, 25 s, 30 s, respectively. The yaw angle ψ and angular rate r are uncontrollable but approach the theoretical prediction.

angular acceleration that the faulty quadcopter can achieve becomes small, causing it to fall and then crash on the ground, shown in Fig. 5.6. Figure 5.7 shows the results of a typical successful simulation trial, demonstrating the control law to work well.

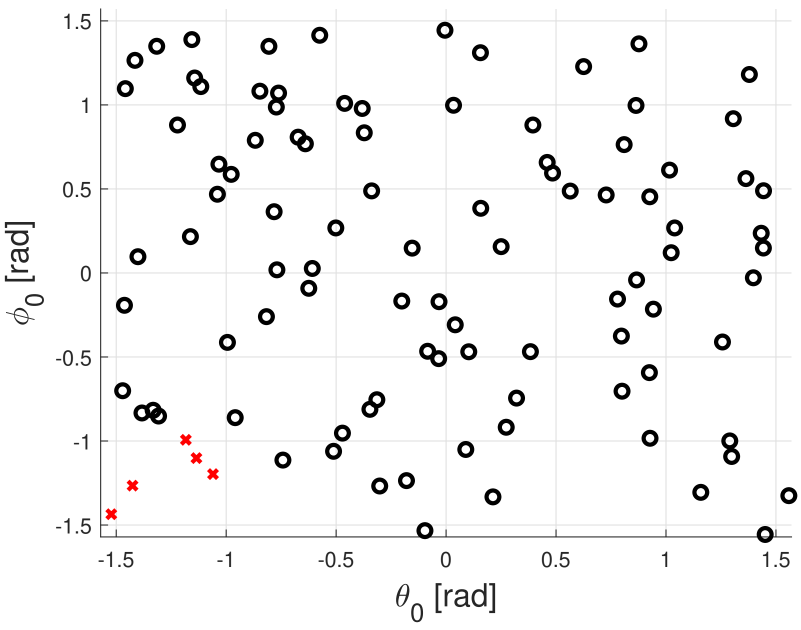


Figure 5.5. Results from physics-based simulations for 100 random initial states. The x-axis and y-axis show the initial pitch (θ_0) and roll (ϕ_0) angles respectively. A black circle indicates that the faulty quadcopter achieved stability, whereas the red cross indicates that it lost stability.

It can be found in Figs. 5.4 and 5.7 that the yaw rate is not zero during the translational flight but reaches 0 when hovering. It indicates that the relation between the attainable yaw rate and propeller speeds for several propeller pitch angles (shown in Fig 5.1) is only valid for hovering. Because this work only considers the dynamic system with 3 control inputs (3 available pitch angles) where the rotating speed remains constant during operation. If the quadcopter has to change the centre rotor speed to have a 0 yaw rate in translation, this leads to the dynamic system changing from three control inputs to four control inputs, which can be further researched in the future.

5.5 Summary

In this chapter, we study the dynamics and control of CVPP quadcopters in the presence of a faulty propeller, that is, a propeller that can no longer adjust its pitch angle. We analyse the equilibrium trajectory, identify the uncontrollable modes, and propose a linear controller. The correlation between the self-rotation speed and the common propeller spinning rate is characterized. The specific condition under which a faulty CVPP quadcopter could hover steadily without self-rotation is given. We believe the new findings could further enhance the development of

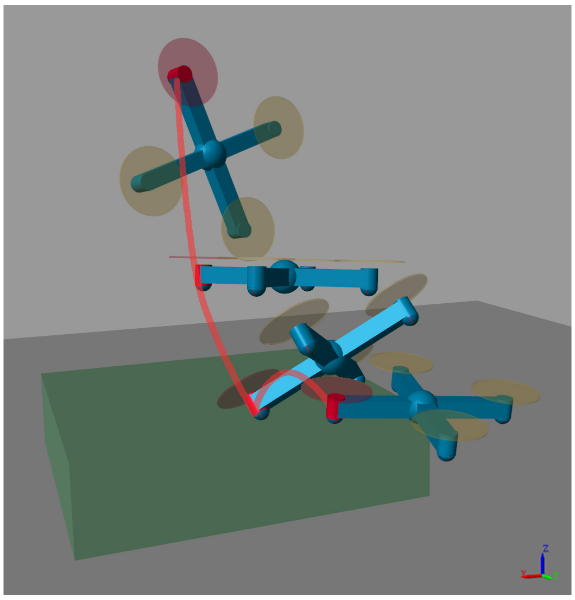


Figure 5.6. Result of the failure trail (crash to the ground), where the initial state is $[x, y, z, \phi, \theta] = [0, 0, 0, -1.2, -1.2]$

CVPP quadcopters for high-performance flights in the future. The finds are further validated by physics-based simulation experiments.

The main theoretical contribution of this chapter is to discover that a CVPP quadcopter can exhibit different but favourable behaviour such as a slow self-spinning speed in the presence of a propeller fault. We also analyse the relationship between the self-spinning speed and the parameter conditions and identify the parameter conditions that lead to zero self-spinning in the presence of one propeller fault. Such a discovery is favourably surprising. The theoretical findings are supported by analysing the equilibrium trajectory, identifying the uncontrollable modes, proposing an H_∞ controller and simulating it in two validation experiments. The aforementioned properties make CVPP quadcopters an attractive platform and may stimulate further research and application in the future.

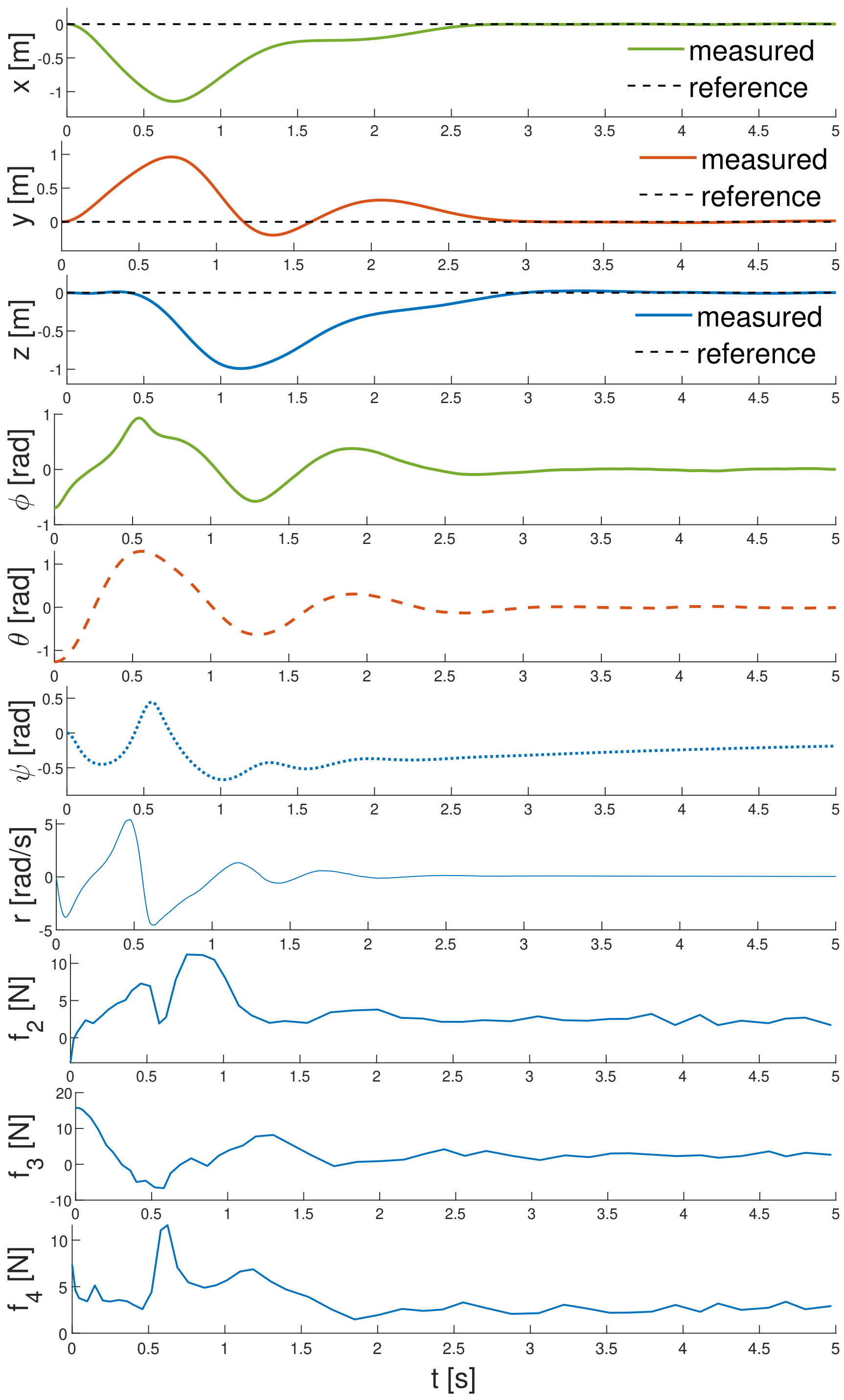


Figure 5.7. Simulation results of random initial angle scenario where the initial state is $[x, y, z, \phi, \theta] = [0, 0, 0, -0.7, -1.26]$ and the target state is $[x^*, y^*, z^*, \phi^*, \theta^*] = [0, 0, 0, 0, 0]$.

Chapter 6

Aerobatic Tic-Toc Control of Variable Pitch Propeller Quadcopters via Reinforcement Learning

6.1 Introduction

Although VPP quadcopters have received increasing attention in recent years, studies mainly focus on fault-tolerant control [170, 171]. The great potential in manoeuvring flight has not been well explored up to now. In fact, VPP quadcopters are suitable for a variety of aerobatic flight manoeuvres. Exploring these manoeuvres could broaden the flight envelope and help address complex flight scenarios. They could be relevant in entertainment and military applications as well as in applications requiring aircraft to navigate narrow confined spaces, all of which have received increased attention in recent years.

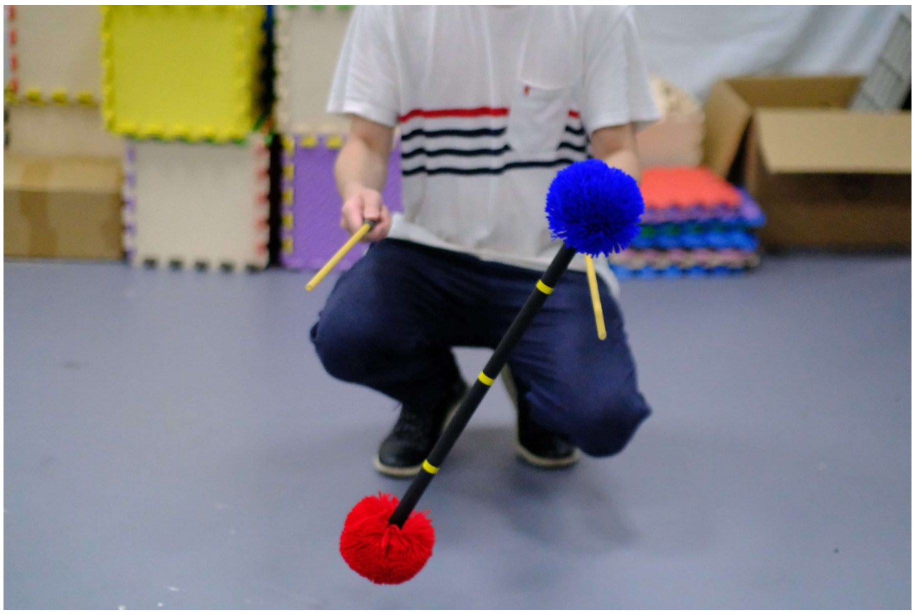


Figure 6.1. A typical tic-toc (also known as “the pendulum”) manoeuvrer of the devil sticks.

Among aerobatic flight manoeuvres, tic-toc is one of the most challenging to be achieved autonomously. The tic-toc manoeuvre attempts to fly the UAV in a vertical plane rather than a horizontal plane. As the UAV is not able to fly steadily in a vertical plane due to the lack of vertical lift, it has to periodically swing back and forth to approximately keep a vertical flight pose. Such a periodic movement can be observed in juggling (see Fig. 6.1). It is a typical aerobatic manoeuvre of helicopters [172]. The work in [114, 35] realized autonomous tic-toc control of a helicopter using inverse reinforcement learning. Such a method, however, requires data of tic-toc trajectories generated by expert pilots in advance.

Up to now, an autonomous tic-toc manoeuvre of a quadcopter has not yet been reported in the literature. Moreover, how to realize it by self-learning without data generated by a skilled pilot is still an open problem. This chapter studies this problem. As the dynamical system is extremely complex, we consider a simplified planar quadcopter model to simulate a VPP quadcopter [173, 174]. Even though the dynamic model is simplified to be two-dimensional, we still face many of the challenges. In particular, as the tic-toc movement is not around any equilibrium point, equilibrium-based control approaches are not applicable.

As far as the authors are aware, the tic-toc manoeuvre has not been achieved by any conventional control approaches in the literature. In this chapter, we design controllers for tic-toc aerobatic flight for planar VPP quadcopters via reinforcement learning (RL). RL is a method that enables an agent to use the reward obtained from its interactions with the environment to generate its control policy [32]. It has received significant attention in recent years due to its potential to address problems that are challenging to solve by conventional control approaches [33, 34]. Although RL has been applied to the control of multi-rotor drones, it is mainly used to achieve flight near the equilibrium point, such as throw-and-hover [36] and attitude control [37].

We use the deep deterministic policy gradient (DDPG) approach to train RL controllers based on carefully designed rewards. The obtained RL controllers are shown to generate two flight modes: spin and tic-toc. The flight performance of either mode is carefully analysed. Then, we evaluate and screen out unfavourable RL controllers by a non-dominated sorting approach [175]. Finally, we extend the remaining RL controller by introducing a compensation control, so that the tic-toc motion can follow a moving reference point, and an LQR-based recovery control, so that the quadcopter can recover from tic-toc to hovering flight. A series of studies are conducted in simulation to verify the proposed approach.

6.2 Tic-toc Manoeuvrer

6.2.1 Introduction

Among the aerobatic flight movements, tic-toc is one of the most challenging to be achieved autonomously, shown in Fig. 6.2. Tic-toc manoeuvrer is to fly the helicopter in a vertical plane rather than a horizontal plane. Since the helicopter cannot fly steadily in a vertical plane due to the lack of vertical lift, the aircraft has to periodically swing back and forth around the vertical plane in order to approximately maintain a vertical flight status. Since the main propeller is located in the centre of the helicopter, the thrust always acts on its centre of mass, so the helicopter always has a large position movement during the tic-toc manoeuvrer. Therefore, the tic-toc manoeuvrer standard of a helicopter is generally to evaluate the displacement of its tail rotor as it acts as its fulcrum during the manoeuvrer while the fuselage is rotating about the vertical axis by 90° back and forth.

Such a manoeuvrer is not achievable for the ordinary quadcopters since their actuators can not provide inverse forces. With the reverse thrust capability of the VPP actuators, a VPP quadcopter can try to achieve such a manoeuvrer. However, different from helicopters, where the thrust acts in the centre of the UAV, the quadcopters generate thrust from the ends of their frames. As a result, the VPP quadcopter may not have such a large displacement during the tic-toc manoeuvring. Such a periodic movement can be observed in juggling (see Fig. 6.1). The performer

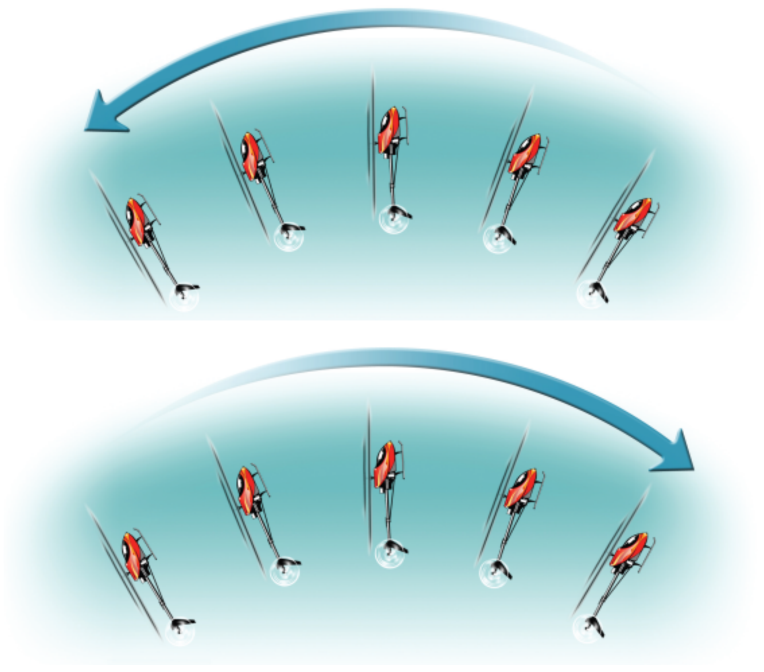


Figure 6.2. A helicopter tic-toc manoeuvrer handbook, reprinted from [176].

uses three sticks (two in his hands namely a beating stick and one with coloured wool balls on each end in the air namely a flying stick). Each time, the player takes turns using the left and right beating sticks to strike at the end of the flying stick, so that it swings in the air without falling. Moreover, the flying stick can maintain its position without large displacement. Inspired by this juggling, we build a planar VPP quadcopter model where only one side of the quadcopter can provide thrust. The aim of this environment is to perform such a tic-toc manoeuvrer by a planar quadcopter to keep its centroid position as constant as possible.

6.2.2 Problem Statement

The planar VPP quadcopter is modelled as a stick with uniform mass distribution which is illustrated in Fig. 6.3. A force f is applied at one end of the stick. Its direction is always perpendicular to the stick. Its sign can be positive or negative (see Fig. 3.1). We consider only a single force as doing so is already sufficient to achieve the tic-toc manoeuvre as demonstrated by Fig. 6.1. Interestingly, a single force acting on the stick end is not sufficient for a quadcopter to hover. Tic-toc is one of a few flight modes that a quadcopter could use to stay in the air under these constraints.

Let d be the distance between the centre point and a reference point (i.e., the red point in Fig. 6.3). The control objective is to design f such that d is as small as

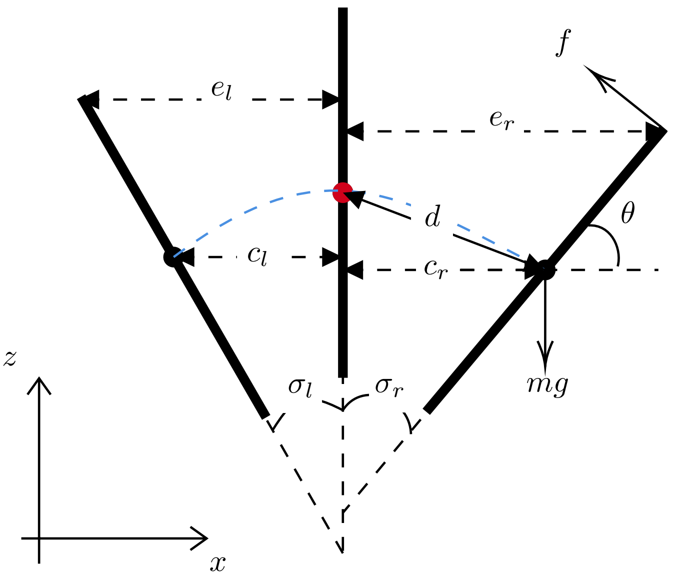


Figure 6.3. The schematic diagram of the whole period of the VPP quadcopter tic-toc manoeuvrer.

possible. As there are no equilibrium states, it is challenging to formally define the target state. We will later quantify the objective by using rewards when designing RL algorithms.

It is shown that different types of indicators and features can describe the manoeuvrer. Some of them, such as periodic time, the centre point of the motion, maximum angle, position deviation and force requirement, can be obtained by direct measurement. Other indicators, such as the space occupation requirements (c_l and c_r), need further calculated. In addition, some of these indicators can be measured every moment, such as the position and angle deviation, while some of them can only be measured at the end of the operation, such as periodic time and maximum force. Since there are no equilibrium states, it is challenging to accurately define the target state. Since the tic-toc manoeuvrer standard is hard to describe, we simplify the control objective to design a f to make d as small as possible while keeping σ_l and σ_r close to 0. We will later quantify the objectives by using rewards when designing RL algorithms.

In the following part, the states of the stick and the dynamic model are presented. The position and velocity of the centre point of the stick are $[x, z]$ and $[u, w]$, respectively. The attitude of the planar quadcopter is described by θ, which is the angle between the stick and the x-axis. The spinning rate is q. Let m and l denote the mass and half length of the stick, respectively, I is the moment of inertia, g is the gravitational constant, and f_T is the total thrust, and τ_T is the torque. Then, the overall state vector is $[x, z, \theta, u, w, q]$ and the dynamic model is .

The position and velocity of the centre point of the stick are $[x, z]$ and $[u, w]$, respectively. The attitude of the planar quadcopter is described by θ, which is the angle between the stick and the x-axis. The spinning rate is q. Let m and l denote the mass and half length of the stick, respectively, I the moment of inertia, g the gravitational constant, and f_T and τ_T the total thrust and torque. Then, the overall state vector is $[x, z, \theta, u, w, q]$

Recall Eqs. (3.28) and (3.30), the dynamic models of the simplified 2D VPP quadcopter are given by

$$\begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ \dot{u} \\ \dot{w} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} u \\ w \\ q \\ -f_T \sin\theta/m \\ f_T \cos\theta/m - g \\ -\tau_T/I \end{bmatrix}. \tag{6.1}$$

and

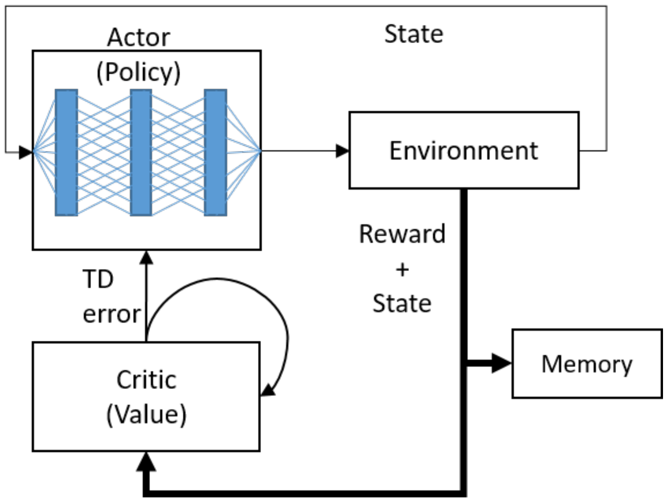
$$\begin{aligned} f_T &= f, \\ \tau_T &= fl. \end{aligned} \tag{6.2}$$

6.3 Reinforcement Learning Controller: Comparison

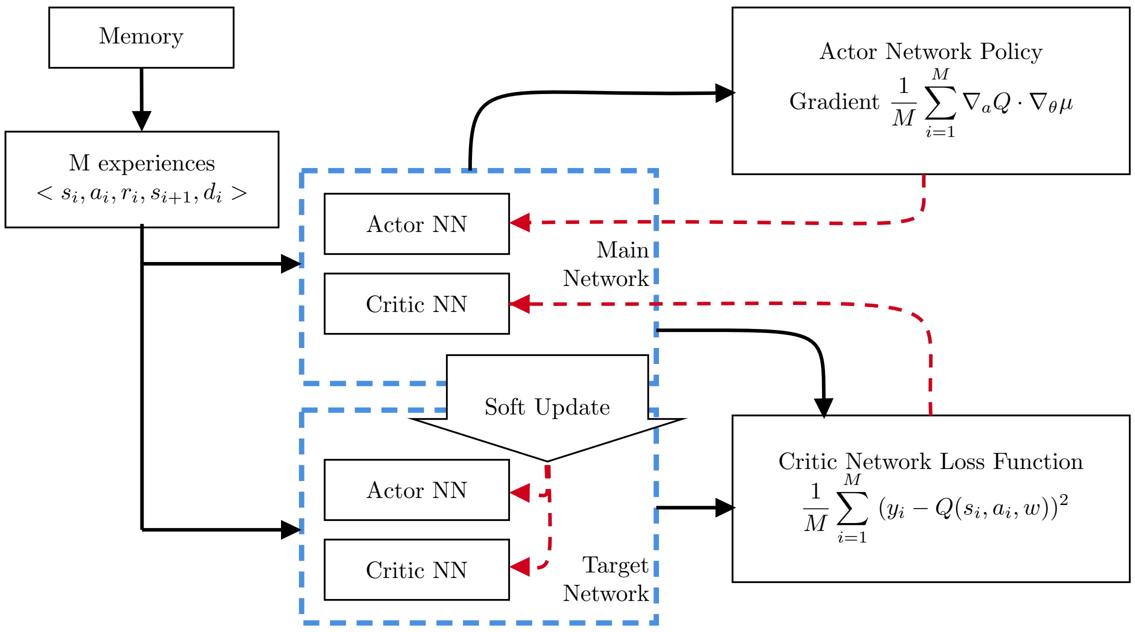
Based on the literature, this work selects four typical algorithms for comparison, namely DDPG, PPO, SAC and TD3. This section mainly introduces the DDPG algorithm, and the contents of other algorithms are detailed in Appendix B.

6.3.1 Deep Deterministic Policy Gradient Methodology

We apply the DDPG approach reported in [33] to train the RL controller. The approach comprises two parts, an actor Neural Network (NN) and a critic NN. The actor NN is an agent that works in the environment whereas the critic NN evaluates the performance of the agent. The actor NN decides which action should be taken based on the state get from the environment. All experiences, including rewards, states, and actions that the actor gathered from the environment, are stored in



(a) Actor Critic structure of the DDPG.


Memory
M experiences
$< s_i, a_i, r_i, s_{i+1}, d_i >$
Actor Network Policy
Gradient $\frac{1}{M}\sum_{i=1}^{M}\nabla_a Q \cdot \nabla_\theta \mu$
Actor NN
Main
Network
Critic NN
Soft Update
Actor NN
Critic NN
Target
Network
Critic Network Loss Function
$\frac{1}{M}\sum_{i=1}^{M}(y_i - Q(s_i, a_i, w))^2$

(b) The DDPG update rule.

Figure 6.4. DDPG NN working example.

memory. The critic NN evaluates the performance of the actor NN based on the experiences randomly drawn from the memory buffer and generate the temporary difference error to guide the update of the actor NN. A deterministic policy gradient algorithm is used to update the actor NN. The critic NN is a value based deep Q-learning NN that uses state feedback and reward as input while its output is a temporal-difference error used to evaluate the performance of the actor. The overall network structure is presented in Fig. 6.4(a).

The whole training process can be divided into three parts, as shown in Figure 6.4 (b). Firstly, the parameters of the NNs are initialized. We choose $Q(s, a, \boldsymbol{w})$

and $\mu(a|s,\boldsymbol{\theta})$ to represent the main critic and actor network, respectively. Here, $\boldsymbol{w}$ and $\boldsymbol{\theta}$ are the weights of the critic and actor networks. Since DDPG uses target networks to improve the stability of the optimization, we use $Q'(s,a,\boldsymbol{w}')$ and $\mu'(a|s,\boldsymbol{\theta}')$ to represent the target critic and actor network respectively. These two target networks are updated periodically.

Secondly, data in the environment is sampled. Suppose the current time is t, define the state of the model to be s_t, action as a_t and the reward value as r_t. After that, set s_{t+1} to be the next state of taking action a_t at states s_t. Since DDPG uses an experiences buffer $\boldsymbol{R}$ to store experiences, design an experience set $< s_i, a_i, r_i, s_{i+1}, d_i >$ and store it in $\boldsymbol{R}$.

Finally, the main NN is periodically updated by selecting a batch of M experience sets from experiences buffer $\boldsymbol{R}$, where the update frequency is a hyperparameter. The actor network is updated according to Eq. (2.3), given by

$$\nabla_{\boldsymbol{\theta}} \approx \frac{1}{M}\sum_{i=1}^{M}\left(\nabla_a Q(s,\mu(s,\boldsymbol{\theta}),\boldsymbol{w})\nabla_{\boldsymbol{\theta}}\mu(s_i,\boldsymbol{\theta})\right).$$

The first ∇_a part describes how the actor gets a high Q value according to the critic network. The second $\nabla_{\boldsymbol{\theta}}$ part presents how to change the parameters to allow the actor network to select the action with high probability [33]. The critic network updates the policy with two Q networks to minimize the loss according to Eq. (2.2), given by

$$L = \frac{1}{M}\sum_{i=1}^{M}(r_i + \gamma Q'(s_{i+1},\mu'(s_{i+1},\boldsymbol{\theta}'),\boldsymbol{w}') - Q(s_i,a_i,\boldsymbol{w}))^2,$$

where γ is the discounting factor. At last, target network is updated by

$$\begin{aligned}\boldsymbol{\theta}' &\leftarrow \tau\boldsymbol{\theta}' + (1-\tau)\boldsymbol{\theta},\\ \boldsymbol{w}' &\leftarrow \tau\boldsymbol{w}' + (1-\tau)\boldsymbol{w}.\end{aligned}$$

The experience set $< s_i, a_i, r_i, s_{i+1}, d_i >$ comes from the memory set, and the policy will be trained according to the selected mini-batch. The larger the batch size used for training, the better the results presented, but the slower the calculation speed is. The pseudocode is shown in Algorithm 1.

Algorithm 1: Deep Deterministic Policy Gradient

```
Input: episodes $K$, steps per episode $T$, batch size $M$
Init: actor policy NN $\mu(a|s,\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$, critic value NN $Q(s,a,\boldsymbol{w})$
      with parameters $\boldsymbol{w}$, replay experience $\boldsymbol{R}$
Init: target policy NN parameters $\mu'(a|s,\boldsymbol{\theta}') \leftarrow \mu(a|s,\boldsymbol{\theta})$, target value NN
      parameters $Q'(s,a,\boldsymbol{w}') \leftarrow Q(s,a,\boldsymbol{w})$
begin
    for $k \in K$ do
        Reset the environment and the noise model $N$
        for $t \in T$ do
            Take action $a_t = \mu(s,\boldsymbol{\theta}) + \epsilon, \epsilon \in N$ by observe state $s_t$
            Observe next time step states $s_{t+1}$, done signal $d_t$ and reward $r_t$
            Store experience set $< s_t, a_t, r_t, s_{t+1}, d_t >$ in $R$
            if *it is time to update* then
                Sample $M$ random experience sets $< s_i, a_i, r_i, s_{i+1}, d_i >$ from $R$
                Compute targets
                    if $d_t == done$ then $y_i = r_i$
                    else $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1},\boldsymbol{\theta}'), \boldsymbol{w}')$
                Update the critic value NN paramters $\boldsymbol{w}$ to minimize the loss
                 (gradient descent) by
                    $L = \frac{1}{M}\sum_{i=1}^{M}(y_i - Q(s_i,a_i,\boldsymbol{w}))^2$
                Update the actor policy NN paramters $\boldsymbol{\theta}$ to maxmize the
                 expected reward (gradient ascent) by
                    $\nabla_{\boldsymbol{\theta}} \approx \frac{1}{M}\sum_{i=1}^{M}(\nabla_a Q(s,\mu(s,\boldsymbol{\theta}),\boldsymbol{w})\nabla_{\boldsymbol{\theta}}\mu(s_i,\boldsymbol{\theta}))$
                Soft target update:
                    $\boldsymbol{\theta}' \leftarrow \tau\boldsymbol{\theta}' + (1-\tau)\boldsymbol{\theta}$
                    $\boldsymbol{w}' \leftarrow \tau\boldsymbol{w}' + (1-\tau)\boldsymbol{w}$
            if $d_t == done$ then quit episode
```

6.3.2 Reinforcement Learning Methods Comparison

Besides DDPG, detailed pseudocodes of the other three RL algorithms are presented in Appendix B.

Since our target may be too hard for the agent NN to learn and the time requirement is so long, we use the standard pendulum environment instead of the planar quadcopter tic-toc environment for a quick comparison. The standard pendulum environment is essentially a rotation pendulum with one end fixed in a gravity environment. It hangs in a downward position in the initial. The target is to train the NN to control the pendulum to stand upward without falling by applying a limited torque. The closer the pendulum is to the target upward position, the larger the reward is given. A slight punishment coefficient is given for the torque action. It is an environment that the RL NN has to learn from failure and can hardly attempt to complete the holding target through exploration.

In order to avoid the interference of hyperparameters, all these algorithms have the same neural network structure. Both the actor and the critic NN are double hidden layer NNs, each layer has 400 and 300 nodes, and all nodes between layers are fully connected. Among them, due to the TD3 algorithm design, it has two critic NNs with the same structure.

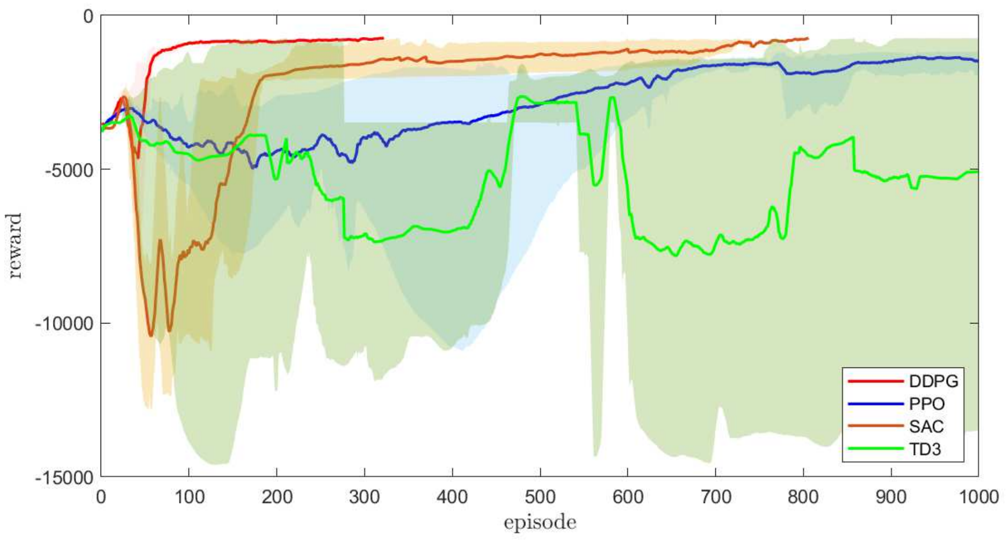


Figure 6.5. Comparison of the learning process of four RL algorithms. The colour of red, blue, brown and green represents the result of DDPG, PPO, SAC and TD3 algorithms, respectively. The translucent areas indicate the upper and lower reward limitation of the multi-trials training process, and the solid line represents the mean reward calculated from 6 trials.

To give an overall evaluation of these RL algorithms, we train each algorithm 6 trials with different random seeds where all RL algorithms use the same sequence of random seeds. Each trial has 10000 episodes and each episode is a complete plane quadcopter movement from the beginning (start from initialization) to the end (reach the end of time). Each episode contains 20 seconds with 0.05 second a step.

The overall comparison for pendulum environment among these four RL algorithms is given in Fig. 6.5 and the detailed results are shown in Appendix B.

The test results show that all RL algorithms can learn how to control the system to reach our target, but with different speeds. Compare with the on-policy algorithm (PPO), the off-policy algorithms (DDPG, TD3 and SAC) can train NN more efficiently because they can use the experience stored in the experience buffer to train the NN as well.

It is worth noting that PPO does not seem to have reached its target reward, but we find the NN is still possible to stabilize the pendulum after training. Because of its randomness, PPO has the lowest reward value when it can stabilize the pendulum, which makes it still considered to have learned how to stabilize the pendulum.

It can be found that although TD3 improves the stability of the DDPG by adding a double critic NN, it also makes the algorithm sensitive to hyperparameters. TD3 can quickly learn to the desired position in some of the trials, but it performs the worst in other trials that start with a different initialization random seed.

In conclusion, DDPG is more suitable for the use of our project compare with the other three popular RL algorithms.

6.3.3 Flight Control Comparison

Because the control of the pendulum environment is successful, we also implement a training section for DDPG to learn to control the plane quadcopter to achieve balance. In order to have an evaluation standard for comparison, we first designed an LQR controller through a linearised model.

LQR based Recovery Controller Comparison

The study starts with a linear control algorithm design based on the input-output linearisation process [177, 178]. In order to control the 2D model shown in Fig. 3.13, it is better to change the output function as the position (x, y) . So the new output will be

$$y = h(x) = [s_x, s_y]^T.$$

Then, if the control state x is available, it is possible to find a fully new feedback control law

$$u = \alpha(x) + \beta(x)v, \tag{6.3}$$

where v is a new input defined later.
Create a new series number r_i (i is the number of the output) which means the relative degree of the system

$$[y_1^{r_1}, y_2^{r_2}]^T = b(x) + \phi(x)u,$$

where

$$\phi(x) = \begin{bmatrix} L_{g1}L_f^{r_1-1}h_1(x) & L_{g2}L_f^{r_1-1}h_1(x) \\ L_{g1}L_f^{r_2-1}h_2(x) & L_{g2}L_f^{r_2-1}h_2(x) \end{bmatrix}, \tag{6.4}$$

$$b(x) = \begin{bmatrix} L_f^{r_1}h_1(x) \\ L_f^{r_2}h_2(x) \end{bmatrix}. \tag{6.5}$$

The system is solvable only if the matrix $\phi(x)$ is nonsingular. Therefore, the equation (6.3) with

$$\alpha(x) = -\phi^{-1}(x)b(x),$$
$$\beta(x) = \phi^{-1}(x).$$

For the system (3.29) above, the parameters are

$$r1 = r2 = 2,$$

with

$$\phi(x) = \begin{bmatrix} -\frac{\sin\theta}{m} & 0 \\ \frac{\cos\theta}{m} & 0 \end{bmatrix}.$$

It is easy to find that $\phi(x)$ is singular for all x. In order to find a nonsingular $\phi(x)$, it is an ideal to get an extra parameter instead of u_1. So consider a new set of the input values

$$\begin{aligned} u_1 &= \zeta, \\ \dot{\zeta} &= \xi, \\ \dot{\xi} &= \overline{u}_1, \\ u_2 &= \overline{u}_2. \end{aligned}$$

The new state space will be presented by

$$\dot{\overline{x}} = f(\overline{x}) + \sum_{i=1}^{2} \overline{g}_i(\overline{x})\overline{u}_i, \tag{6.6}$$

where

$$\begin{aligned} \overline{x} &= [s_x, s_y, v_x, v_y, \zeta, \xi, \theta, w]^T, \\ \overline{u} &= [\overline{u}_1, \overline{u}_2]^T, \\ f(x) &= \begin{bmatrix} v_x \\ v_y \\ -\frac{\sin\theta}{m}\zeta \\ -g + \frac{\cos\theta}{m}\zeta \\ w \\ \xi \\ 0 \\ 0 \end{bmatrix}, \end{aligned}$$

and

$$\begin{aligned} g_1(x) &= [0,0,0,0,0,0,1,0]^T, \\ g_2(x) &= [0,0,0,0,0,0,0,-\frac{L}{2I}]^T. \end{aligned}$$

As a result, the system (6.6) is solvable with the relative degree

$$r1 = r2 = 4,$$

and

$$[y_1^{r_1}, y_2^{r_2}]^T = b(\bar{x}) + \phi(\bar{x})u.$$

Recall the calculation formula (6.4), the feedback control law $\bar{u} = \alpha(\bar{x}) + \beta(\bar{x})v$ can be computed and shown in Fig. 6.6.

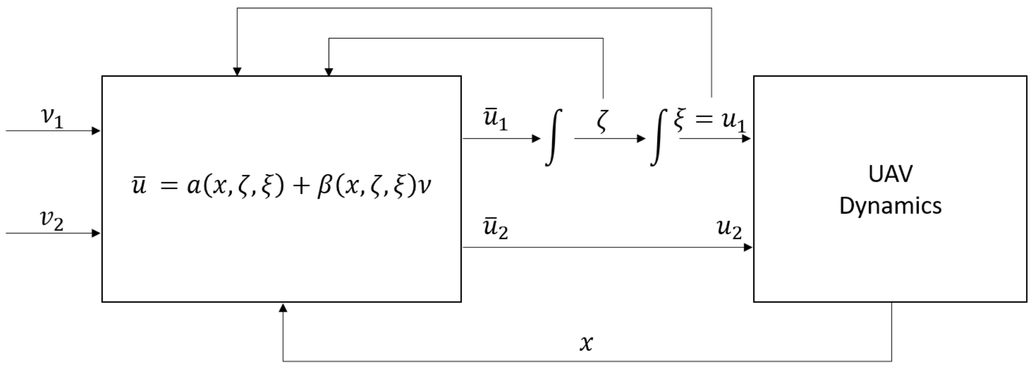


Figure 6.6. The control law sketch.

The new system (6.6) has 8 dimensions which is equal to the sum of r_i. Then, the new system (6.6) is fully linear and controllable. The new coordinate $z = \Phi(\bar{x})$ is

$$\begin{aligned} z_1 &= h_1(x) = x, & z_5 &= h_2(x) = y, \\ z_2 &= L_f h_1(x) = \dot{x}, & z_6 &= L_f h_2(x) = \dot{y}, \\ z_3 &= L_f^2 h_1(x) = \ddot{x}, & z_7 &= L_f^2 h_2(x) = \ddot{y}, \\ z_4 &= L_f^3 h_1(x) = \dddot{x}, & z_8 &= L_f^3 h_2(x) = \dddot{y}, \end{aligned}$$

which is linear and also controllable.

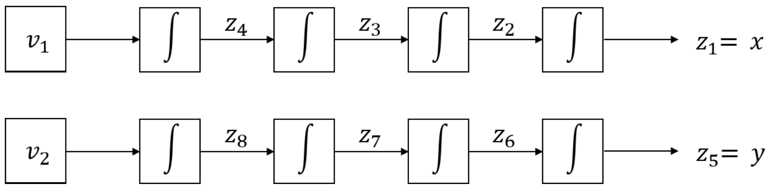


Figure 6.7. The block diagram of the linear control system.

After the coordinate changed, the system shown in Fig. 6.7 is changed to

$$\begin{aligned} \dot{z} &= \mathbf{A}z + \mathbf{B}v \\ y &= \mathbf{C}z \end{aligned} \tag{6.7}$$

with

$$\begin{aligned} z &= [z_1, z_2, ..., z_8]^T, \\ v &= [v_1, v_2]^T. \end{aligned}$$

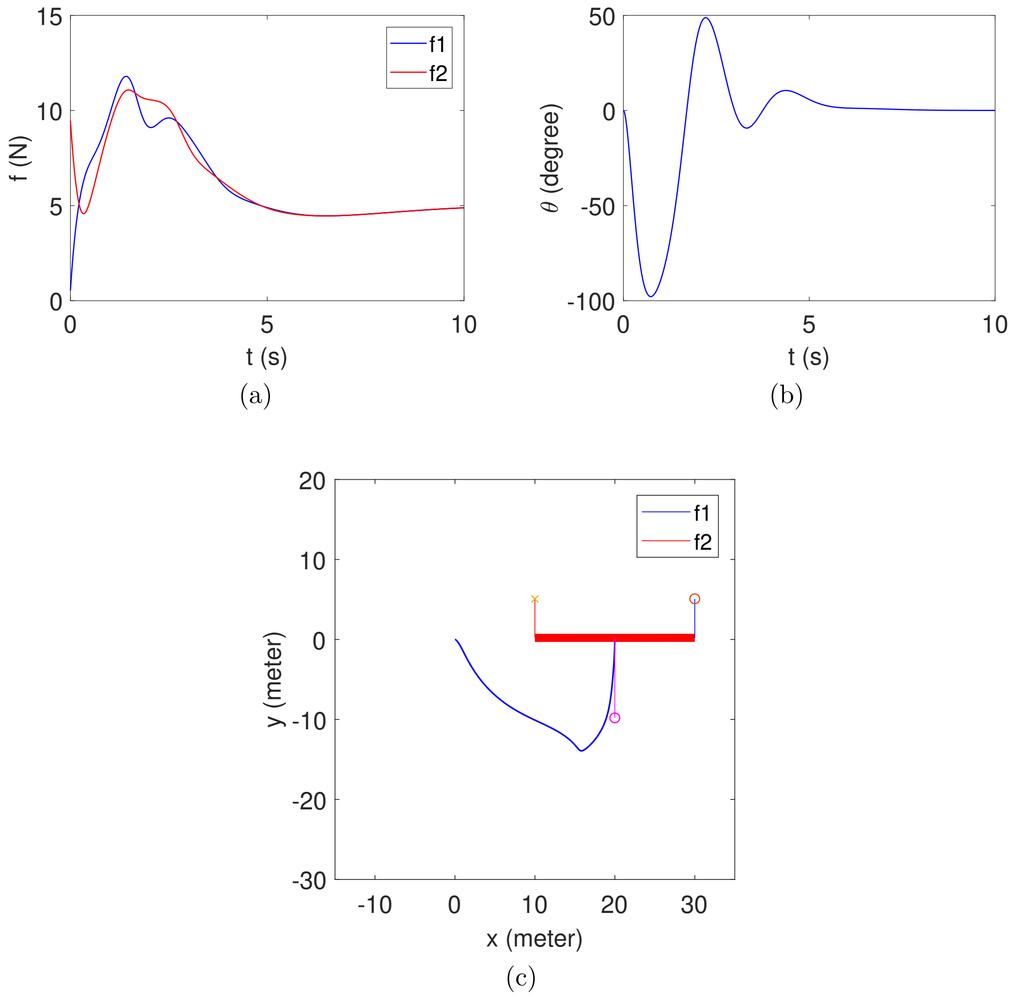


Figure 6.8. Result of LQR based on the simple model experiment. (a) is the force changing with time; (b) is the degree changing with time; (c)is the trajectory of the stick, the blue line is the trajectory and the thick red line is the stick.

$$A = \begin{bmatrix} A_1 & 0 \\ 0 & A_1 \end{bmatrix}, \quad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \qquad C = \begin{bmatrix} C_1 & 0 \\ 0 & C_1 \end{bmatrix},$$

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

$$C_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}.$$

In the linear Eq. (6.7), a feedback controller like LQR can be used. After the calculation, the LQR controller can be placed in the Matlab environment. The result is shown in Fig. 6.8, which can achieve a good result. However, the controller can not self-improve the control performance. Therefore, we will test if the RL algorithm can handle the same experiment environment.

RL based Recovery Controller

According to the comparison result shown in the preview section, we choose DDPG as our basic RL algorithm [33]. The experiment environment is shown in Fig. 6.9 (d). The environment is set to be the one only affected by gravity and ignore the aerodynamic resistance. The 2D quadcopter is a simplified model that needs to be controlled, which is shown as the red stick. The blue line is the trajectory of the quadcopter and the red square is the target place. In every episode, the quadcopter is set to start from the original point with all states being set to 0. At the beginning of each episode, instead of changing the quadcopter initial state, the target is set to a random position within a circular range of 50 m from the centre. Inputs are two vertical forces from two ends which can be set to any value smaller than 15 N, up or down direction.

The basic network structure we use is a four-layered perception structure, including an input layer, two hidden layers and an output layer, all layers are fully connected. The input layer collects states patterns, and the output layer has action signals to which input patterns may map. Two hidden layers have 400 and 300 nodes respectively. This neural network structure design achieves a good balance between the time efficiency and performance. Both actor and critic NN use this network structure.

In the experiment, the reward is designed by

$$r = 0.1t - 0.1\sqrt{s_x^2 + s_y^2} - 0.1\theta^2, \tag{6.8}$$

where t is the running time, s_x and s_y are the normalized position distance of between the centre of the stick and the target position along x and y axis and θ is the angle.

Inputs of the DDPG algorithm contain 6 states error values (2 positions, 2 velocities and 2 accelerations), 3 attitudes error values (1 angle, 1 angle velocities and 1 angle acceleration), 2 forces form equation and 1 labelled reward value (show if the quadcopter reaches the target). The output values are the forces on the ends.

Figure 6.9 (a),(b) and (c) claim that after 150 seconds running, the model reaches to period of oscillation. Every time the stick reaches the target with non-zero speed, which makes it fly away from the target state. Despite this, the stick knows how to

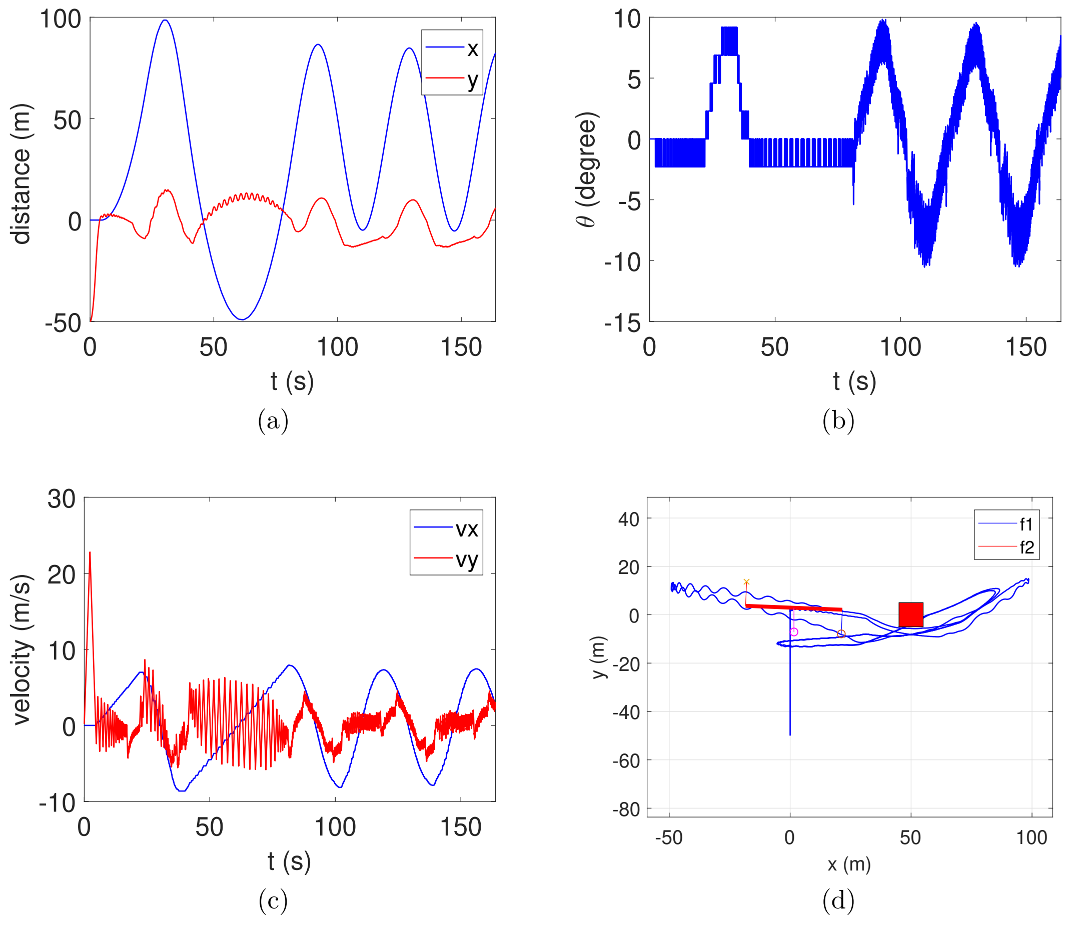
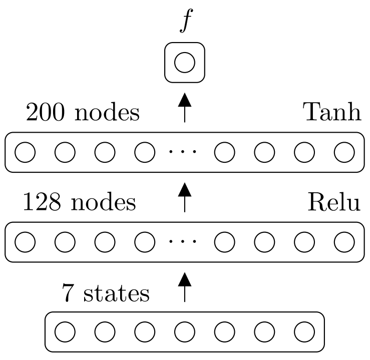


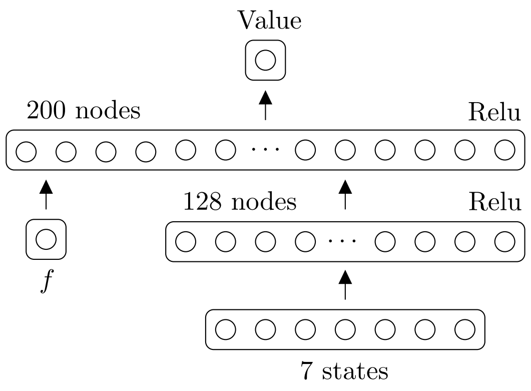
Figure 6.9. The result of the RL planar quadcopter position tracking experiment. (a) is the distance changing with time; (b) is the degree changing with time; (c) is the velocity changed with time; (d) shows the trajectory of the stick.

control itself to reach the target by self-learning. Such performance can be further improved by tuning the weight parameters of the reward function to increase the punishment importance of the velocity. However, there is no general standard for the design of the reward function, and most adjustments are based on experience.

Overall, comparing between Figs. 6.8 and 6.9, it can be found that the RL control performance is not as good as the LQR controller for the hover control of planar quadcopter. This indicates that for problems that are easy to implement by using control theory, designing a controller based on control theory is a more effective method. However, to some difficult problems, it is better to use RL to design a controller and enhance its performance by adding other extension algorithms, such as the Tic-toc manoeuvrer.



(a) Actor NN structure



(b) Critic NN structure

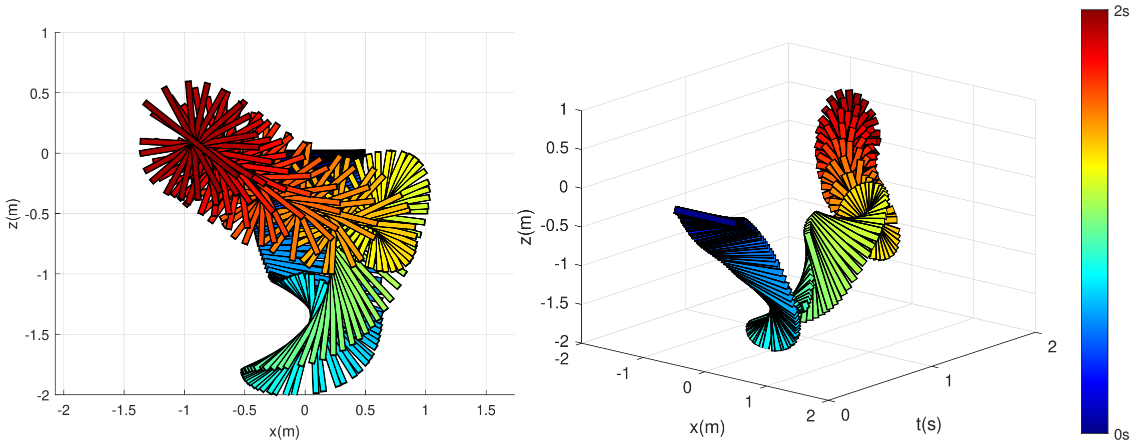
Figure 6.10. The neural network of the DDPG training algorithm.

6.4 Reinforcement Learning Controller: Training and Analysis

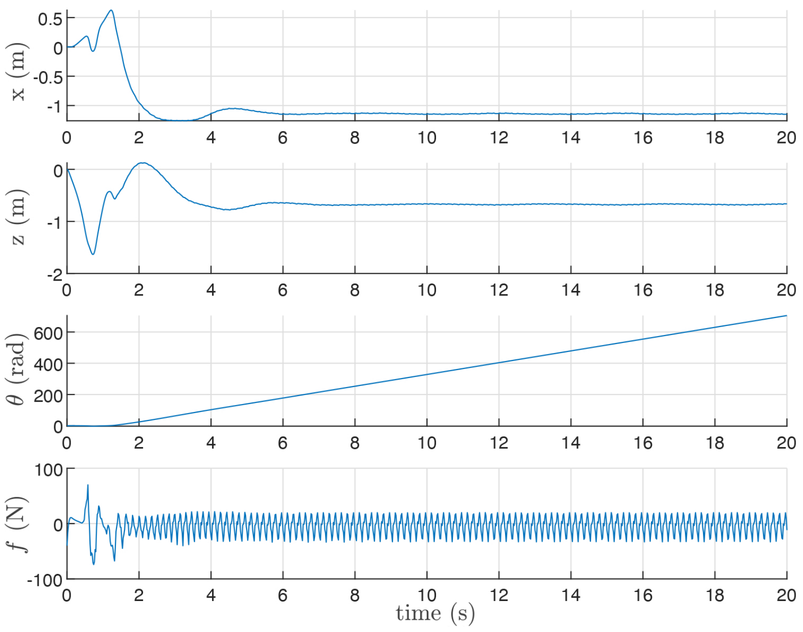
6.4.1 Algorithm Structure

We apply the DDPG approach reported in [33] to train the RL controller. The approach comprises two parts, an actor Neural Network (NN) and a critic NN. The actor NN is an agent that works in the environment whereas the critic NN evaluates the performance of the agent. A deterministic policy gradient algorithm is used to update the actor NN. The critic NN is a value-based deep Q-learning NN that uses state feedback and reward as input while its output is a temporal-difference error used to evaluate the performance of the actor.

The NN structures are shown in Fig. 6.10. Subsequent layers are fully connected. The output of the actor NN is the force f that acts at the end of the planar quadcopter. To avoid non-linearity, we use $\cos\theta$ and $\sin\theta$ instead of θ as the environmental output states. As a result, there are 7 feedback states which are obtained by the agent from its interaction with the environment: $\left[\sin(\theta), \cos(\theta), \dot{\theta}, x, z, \dot{x}, \dot{z}\right]$. These feedback



(a)



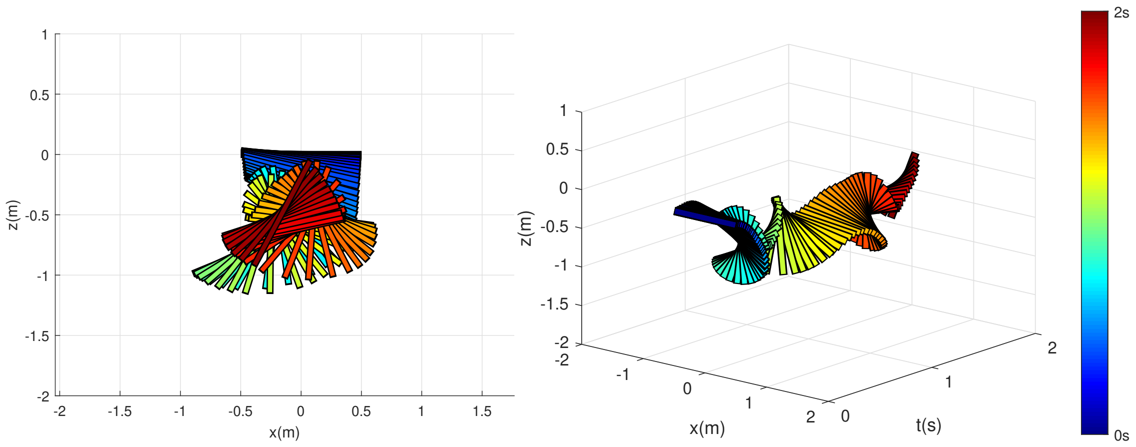
(b)

Figure 6.11. Position holding simulation results of the spin mode, where colour bar represents time. (a) shows the first 2 seconds timeline simulation results of a spin mode NN and (b) shows the full 20 seconds states simulation results of spin mode. It takes about 2 seconds for the initialization phase to achieve the manoeuvre phase.

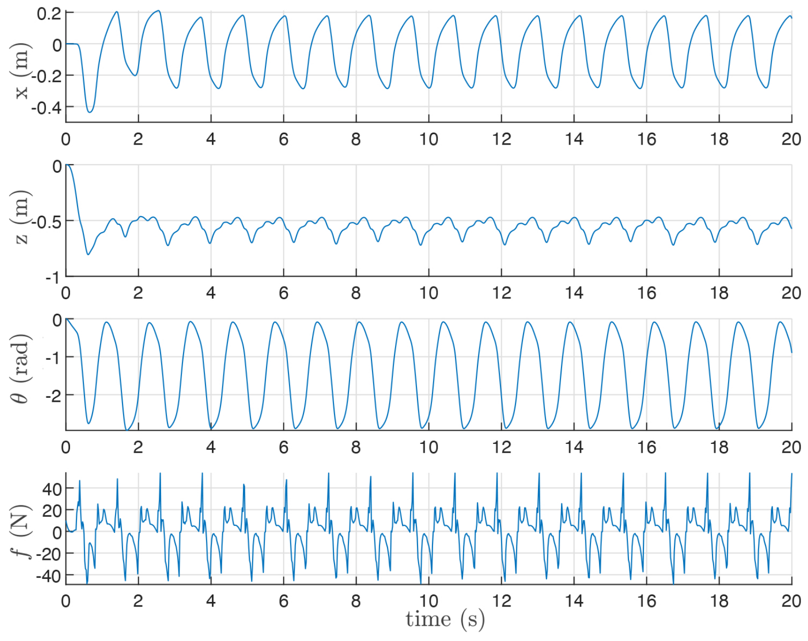
states are also used to design rewards. The whole NN is built in the Matlab Deep Reinforcement Learning toolbox environment.

6.4.2 Training Process

Since there is no target equilibrium states, we must design appreciate reward to reflect our control objective. The rewards are designed as below. Positive and negative rewards, respectively, represent encouragement and penalty of certain behaviour.



(a)



(b)

Figure 6.12. Simulation result comparison of the tic-toc mode where colour bar represents time. (a) shows the simulation results of first 2 seconds timeline for a tic-toc mode NN and (b) shows the full 20 seconds states results of it. The planar quadcopter quickly switched to the tic-toc manoeuvrer with very little initialization time.

1) To reward the centre point of the quadcopter for approaching the target location, we design the following distance deviation penalty function:

$$r(d) = -0.1d^2 - 100d_{\text{far}},$$

where

$$d_{\text{far}} = \begin{cases} 0, & \text{if } d < 4; \\ 1, & \text{if } d \geq 4. \end{cases}$$

Here, d_{far} gives a strong penalty when the centre point gets too far away from the reference point. When $d_{\text{far}} = 1$, the episode is stopped.

2) To reward the quadcopter for assuming a vertical attitude (i.e. θ close to $-\pi/2$), we use reward function

$$r(\theta) = -0.01(\theta + \pi/2)^2,$$

where $\theta \in [-\pi, \pi)$.

3) To minimize the required force magnitude, a force penalty is designed as

$$r(f) = -0.01f^2.$$

4) To encourage the quadcopter to remain in the air for a long time, we increase the reward with the flying time by

$$r(t) = 0.1t,$$

where t is the time elapsed since the start of the episode.

In total, the reward function is

$$R = r(d) + r(\theta) + r(f) + r(t). \tag{6.9}$$

According to Eq. (6.1), we can estimate whether the attitude of the plane quadcopter meets our expectation by the position and angle. Relatively speaking, the deviation of position can be compensated by the later design of the trajectory planning system, but the compensation of angle error is difficult to achieve. Therefore, the consideration of angle needs to be the most important one among all reward designs.

For each episode, the quadcopter starts from the initial hovering state, which is $x_0 = 0, z_0 = 0, \theta_0 = 0$. The target reference position is randomly generated, $x_t, z_t \in [-1, 1]$. The randomness helps strengthen the RL controller's generalization ability and stability [179]. Within each episode, the quadcopter tries to reach, and remain close to the target position. If the quadcopter flies too far away (more than 4 m), the episode will be marked as a failure and stop. We train the DDPG agent for 30000 episodes, with each episode lasting at most 10 s with 0.02 s sampling time. All episodes whose returns are greater than -50 are saved for further analysis.

6.4.3 Result Analysis

The trained RL controllers exhibit two flight modes. The first is a spin mode, where the planar quadcopter spins around a fixed position, therefore θ varies from 0 to 360 degrees (see Fig. 6.11). The second is a tic-toc mode, where the planar quadcopter swings around a point back and forth, therefore θ is constrained in a bounded interval (see Fig. 6.12).

In either of the flight modes, the entire flight can be split into two phases. The first is a settling phase, in which the planar quadcopter starts from a horizontal attitude and then gradually converges to a steady periodic motion. During this phase, there will be large position and attitude deviations. The second is a steady periodic phase, in which the states vary in a steady periodic manner.

In order to quantify the settling time, we use trigonometric functions to fit the steady periodic curve. As illustrated in Fig. 6.13, the distance d between the quadcopter centre point and the reference point is shown by the blue curve. We can fit the blue curve in the steady periodic phase by

$$d_f(t) = A_0 + A_1 \cos(\omega t) + A_2 \sin(\omega t).$$

Once $|d(t) - d_f(t)|$ remains below a threshold (for example, 0.1) the settling phase ends and the corresponding time is the settling time. We have checked all the trained controllers and noticed that it takes a longer settling time for the spin mode (around 6 s) than for the tic-toc mode (around 3 s).

By comparing the performance of the spin and tic-toc modes as shown in Figs. 6.11 and 6.12, the tic-toc mode does not exhibit significant fluctuations during the settling phase and reaches the steady periodic phase faster. In terms of space occupation, the spin mode takes more space than the tic-toc mode (observed from the XZ plane). In terms of required force, the spin mode requires high force (70 N maximum) in the settling phase and then much less force (25 N maximum) in the steady periodic phase. As a comparison, the required force of the tic-toc mode does not vary significantly during different phases and the maximum value is 50 N.

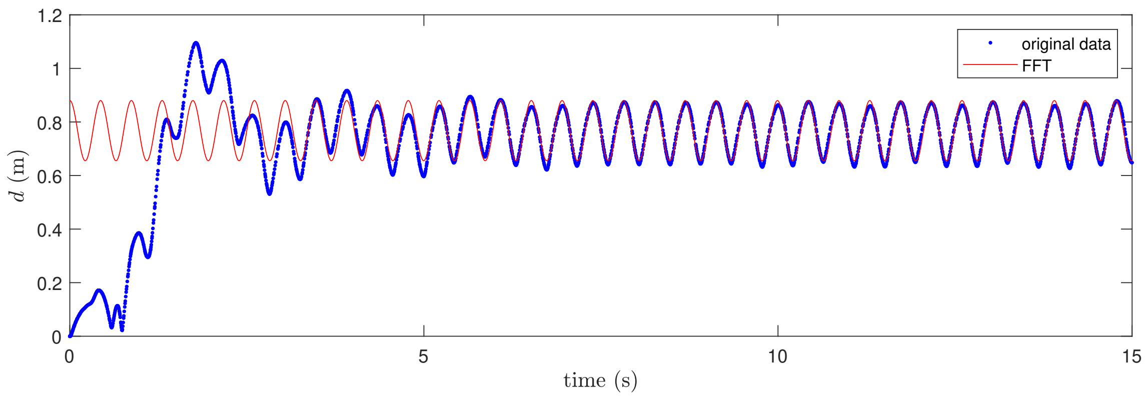


Figure 6.13. Settling time quantification through comparison of the original distance curve and the Fourier fitting curve. To avoid interference via the initialization phase, we exclude the first 5 seconds when generating the Fourier fitting curve. The fitting distance function shown in the figure is $d_f(t) = 0.76 + 0.11\cos(14.46t) - 0.001\sin(14.46t)$, and the settling time is around 2.8 s.

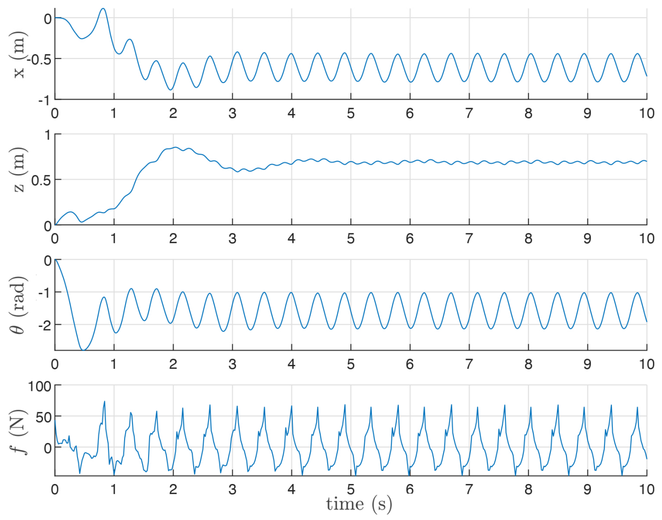
Table 6.1. Comparison among three tic-toc manoeuvrer NNs with different speeds.

	High-speed	Middle-speed	Low-speed
Swing Angle (rad)	1.12	1.38	2.26
Time Period (s)	0.45	0.56	0.94
Maximum Force (N)(+)	73.68	50.34	32.65
Maximum Force (N)(-)	-46.43	-55.28	-48.34

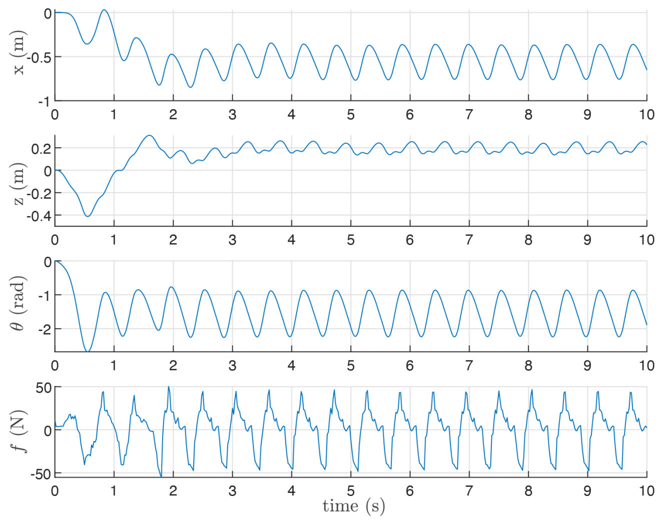
While we have incorporated an angle penalty reward function $r(\theta)$, why could the controller still learn the spin mode? This is because the total reward function considers other properties as well and we stored all the RL NNs whose overall rewards were greater than the set threshold.

6.5 Reinforcement Learning Controller: Further Evaluation and Screening

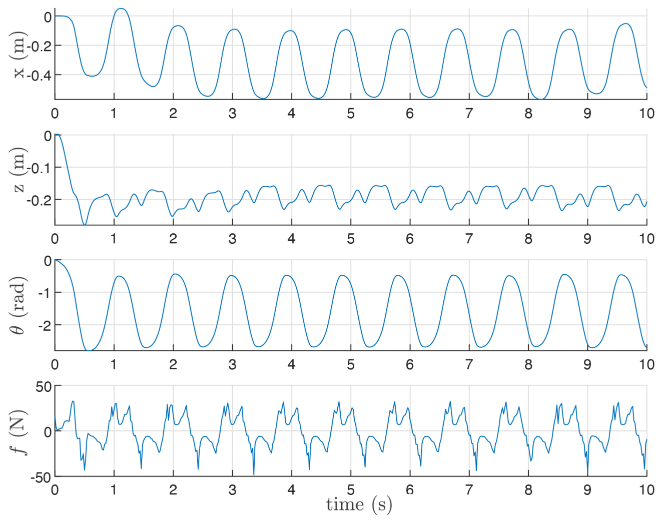
In the last section, we show that the tic-toc flight mode has less settling time and lower space occupation than the spin mode. It should be noted that different training episodes can lead to different controllers. These controllers may all implement the tic-toc manoeuvre, but their performance is very different. The inherent diversity of solutions (and performance) is fundamentally due to the total reward function being composed of a mixture of metrics, and the trained controllers may place different emphasis on different metrics. In this section, we evaluate different tic-toc controllers and show how to screen out solutions according to additional metrics.



(a) A high-speed tic-toc mode NN, where the manoeuvre frequency is 3 Hz.



(b) A middle-speed tic-toc mode NN, where the manoeuvre frequency is 2 Hz.



(c) A low-speed tic-toc mode NN, where the manoeuvre frequency is 1 Hz.

Figure 6.14. Simulation result comparison of different NNs that have similar reward value. It indicates that it is difficult to distinguish different performance qualities only by the reward function.

Figures 6.14 shows examples to demonstrate the performance of different controllers. The three examples could be classified to be high-speed, medium-speed, low-speed controllers based on the periodic time of their steady phases. The specific values of the maximum swing angles, periodic time, and maximum forces of these examples are given in Table 6.1. The sign of force only represents the direction. As can be seen, the smaller the swing angle and the shorter the time period, the larger the force that is required.

In the rest of the section, we introduce three metrics to evaluate different RL controllers and propose a method to screen the unfavourable ones out.

6.5.1 Evaluation Metrics

To give an overall evaluation of the performance of the NN, we use the following three metrics:

1) The first metric is the mean distance between the centre and reference points. It is denoted as d_{mean}.
2) The second metric is the maximum space occupation $s_{\max}$, which is defined as

$$s_{\max} = \max\left(|e_r|, |e_l|\right), \tag{6.10}$$

   where e_r and e_l denote respectively the rightmost and leftmost distance of the stick's top end from the vertical plane (see Fig. 6.3).
3) The third metric, $f_{\max}$, is the maximum magnitude of the force during the entire control process including both settling and steady periodic phases. This metric would be relevant for practical realizations of the RL controller.

It must be noted that the three metrics could not be designed as rewards during the training process. That is because they are defined for the entire control process and can not be used for timely feedback to evaluate the performance of the training NN controller.

It is favourable if d_{mean}, $s_{\max}$, and $f_{\max}$ are small; the smaller the better.

In general, to fully describe the qualities of the manoeuvrer, we can construct other 4 different objective functions based on indicators and features of the tic-toc manoeuvrer as follows:

1) Maximum angle $\theta_{\max}$. It is determined by the larger value between θ_l and θ_r, where $\theta_{\max} = \max(\theta_l, \theta_r)$.

2) Reward value R. It is the reward value to measure the performance of the action in RL training.

3) Maximum central distance deviation c_{max}. It is a value that can describe the horizontal movement of the centre of mass.

4) Initialization phase time t_i. It is the time of the initialization phase which is calculated by the first time entering periodic movement.

It is favourable if d_{mean}, s_{max}, and f_{max} are small. Next, we will use a multi-objective optimization method to further screen the trained NNs.

6.5.2 Network Screening

This subsection addresses how to screen a large number of RL controllers based on the aforementioned three metrics.

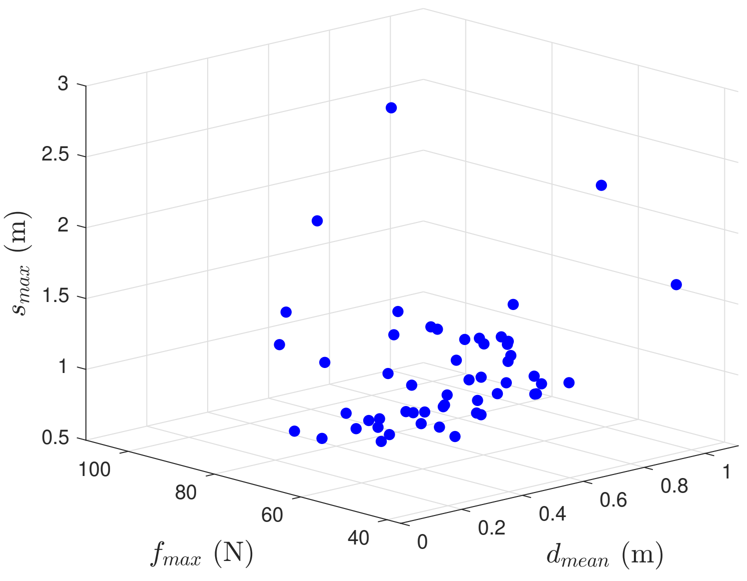
One approach is to assign different weights to the three metrics according to ones' own preferences and then use a weighted summation of the three metrics as a single metric. Since many networks perform well on one metric but worse on another, one overall metric may not be sufficient to choose a suitable controller. Therefore, we use a screening algorithm to identify the set of best NNs from all the trained NNs. Here "best" refers to NNs that are not dominated by other NNs.

Figure 6.15(a) shows all trained NNs as a point cloud in the metric space, where the axes correspond to the three metrics $d_{\text{mean}}, s_{\text{max}}$ and f_{max}, respectively. Figure 6.15(b) shows the corresponding Pareto-efficient frontier.

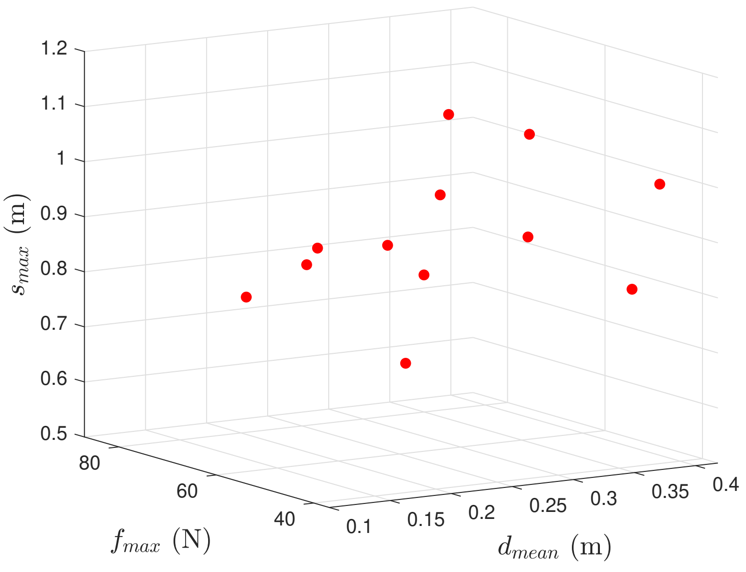
While this method screens out a large portion of unfavourable NNs, we can further reduce the selection based on our preferences. For example, if we want the space occupation to be as small as possible, we could choose the bottom-left red dot in Fig. 6.15(b). The corresponding curves of the states for this RL controller are given in Fig. 6.16. In addition, we can choose NNs from anywhere on the Pareto optimal surface to meet multiple metric requirements.

Based on the requirements from the task, we could determine which NN meets our requirements from the optimized result. In most cases, we may pay more attention to the maximum force requirements and the maximum space occupation requirements. Therefore, the experimental results of the comparative test for two preferences are shown in Fig. 6.16, where the summarized comparison is given as follows:

1) Minimum force preference. In this scenario, the actuator power of the planar quadcopter is insufficient, and it is difficult to provide a large thrust. We aim



(a)

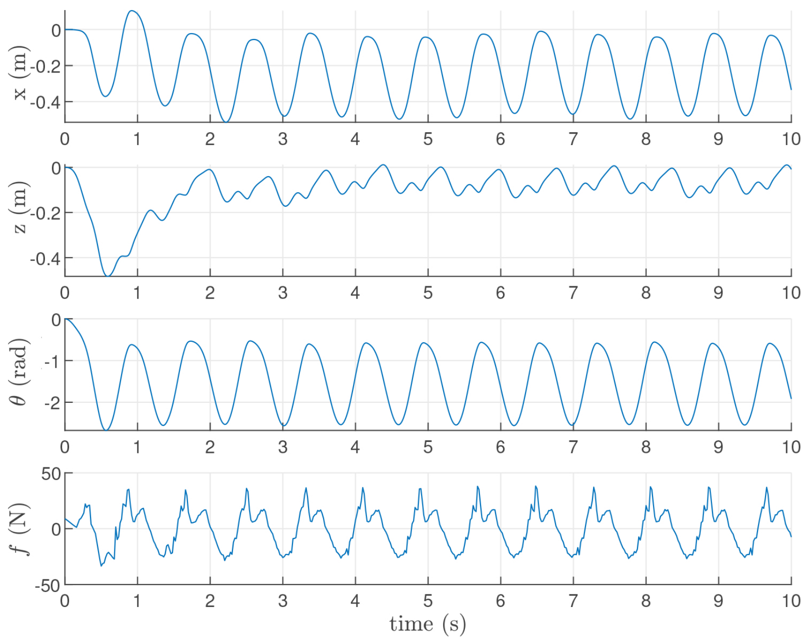


(b)

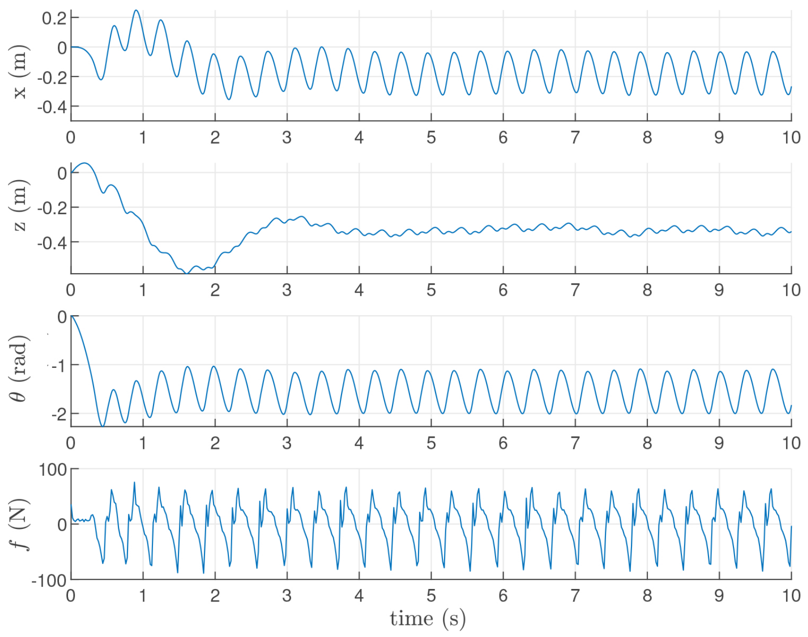
Figure 6.15. Screening of trained NN controllers. (a) shows the Point cloud of all trained NNs and (b) shows the Pareto-efficient frontier point cloud of all trained NNs.

to control the planar quadcopter that can move around the target position with minimum force, and space constraints are weak. Simulation results are shown in Fig. 6.16(a), where the maximum force is less than 40 N, but the space occupation requirement is larger than 1.16 m.

2) Minimum space occupation preference. In this scenario, the quadcopter needs to pass through an extremely narrow passage. We aim to find the minimum



(a)



(b)

Figure 6.16. Simulation results of the planar quadcopter controlled by RL NN performing tic-toc manoeuvrer with different tendencies. Performance of an RL controller selected from the Pareto-efficient frontier. (a) is the controller that uses the minimum force (less than 35 N) and (b) is the controller that uses the minimum space occupation (about 0.74 m).

space occupation along with a release of force requirement and error constraint.

Simulation results are shown in Fig. 6.16(b), where the space occupation requirement is about 0.56 m and the force requirement reaches up to 89 N.

6.6 Reinforcement Learning Controller: An Extension

This section extends the RL controllers to further improve their performance. In addition, a recovery controller is designed to restore hovering flight and a NN migration method is provided to realize the control of different dynamic models by the target NN.

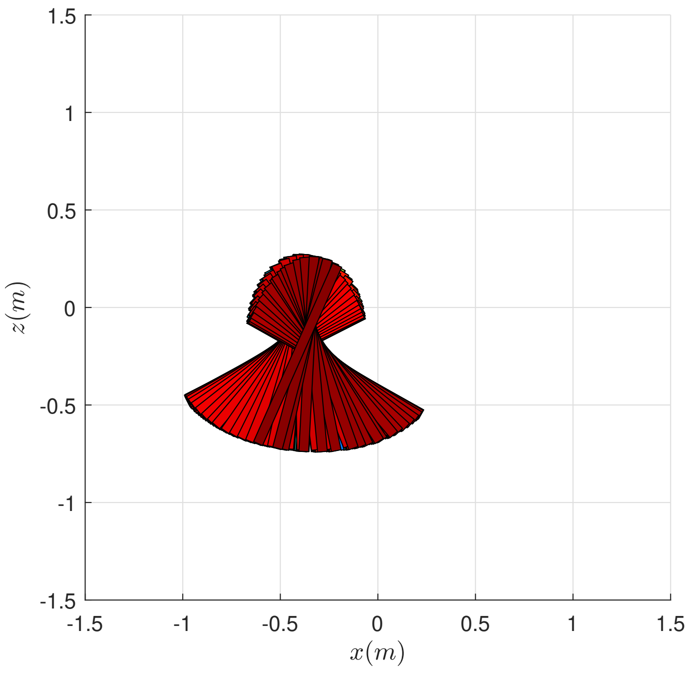
6.6.1 Trajectory Compensation

A problem of the RL controllers is that there may exist steady-state errors between the average position of the centre point and the reference point. We now seek to suppress this steady error. Moreover, we seek to further decrease the maximum space occupation $s_{\max}$.

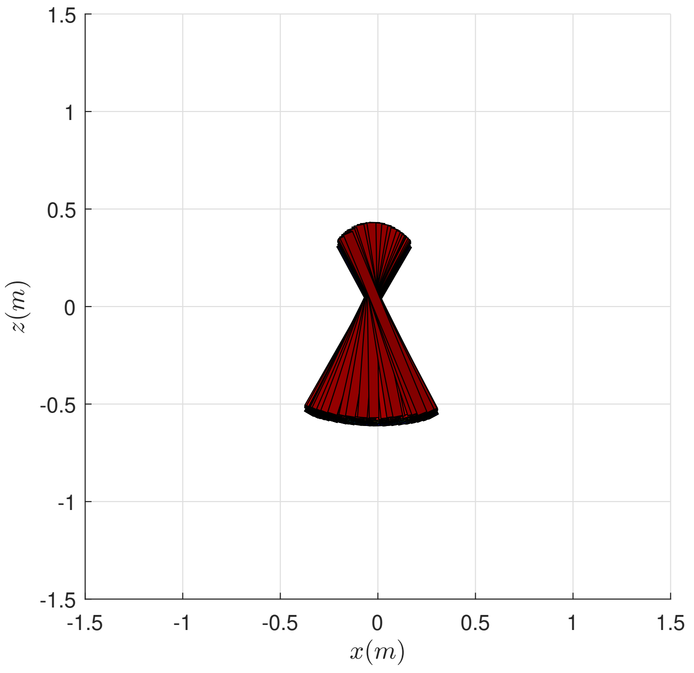
We design a trajectory compensation method that uses a PID controller to compensate the steady-state error and reduce $s_{\max}$. The idea of this method is to design a swing trajectory to offset unnecessary displacement, thereby reducing the maximum manoeuvrer distance deviation. In our scenario, the error between actual and desired trajectories is given to the PID controller as the input where the output value is the trajectory that needs to be compensated.

Figure 6.17 verifies the effectiveness of the proposed controller. As can be seen, the compensation method can reduce the average steady-state error from $[-0.54, -0.22]$ to $[-0.05, -0.12]$. Moreover, $s_{\max}$ decreased from 0.98 m to 0.51 m. However, this result comes at the cost of the maximum force increasing from 78 N to 165 N.

The advantage of this compensation controller is that we can flexibly enhance the flight performance of an existing trained RL controller according to our needs, instead of training new RL controllers.



(a)



(b)

Figure 6.17. Simulation results (steady periodic phase) of the planar quadcopter with or without trajectory compensation. The trained NN strikes a good balance among all three objective functions (d_{mean}, $s_{\max}$ and $f_{\max}$). (a) shows the simulation results of the NN controller without trajectory compensation and (b) shows the simulation results of the NN controller with trajectory compensation. The overall maximum space occupation is decreased by 53%.

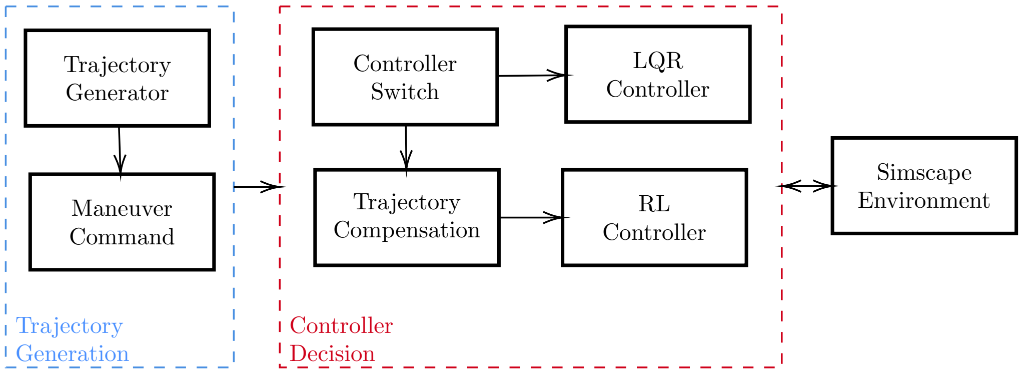


Figure 6.18. Overall control system structure used for the narrow space passing simulation. The structure is divided into three parts, trajectory generation (shown in blue block), controller decision (shown in red block) and simulation environment.

6.6.2 Recovery Controller System

Our trained RL controller can only achieve tic-toc flight. If the planar quadcopter wants to restore hovering flight, a new controller would need to be designed and integrated.

We introduce an LQR controller for hovering flight control based on a modified version of Eqs. (6.2) and (3.5). The modification is to change the single force as in Eq. (6.2) to two forces applied on the two ends of the stick. As a result, Eq. (6.2) becomes

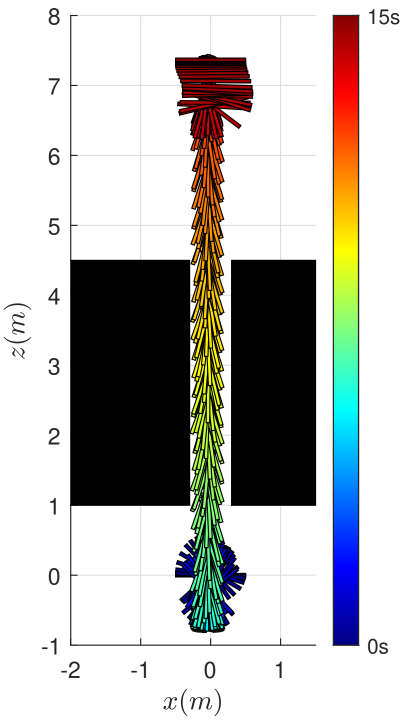
$$\begin{aligned} f_T =& f_r + f_l, \\ \tau_T =& (f_r - f_l) l, \end{aligned}$$

where f_r and f_l are the forces acting on right and left ends of the VPP quadcopter, respectively. The design of the LQR controller is standard and omitted here.

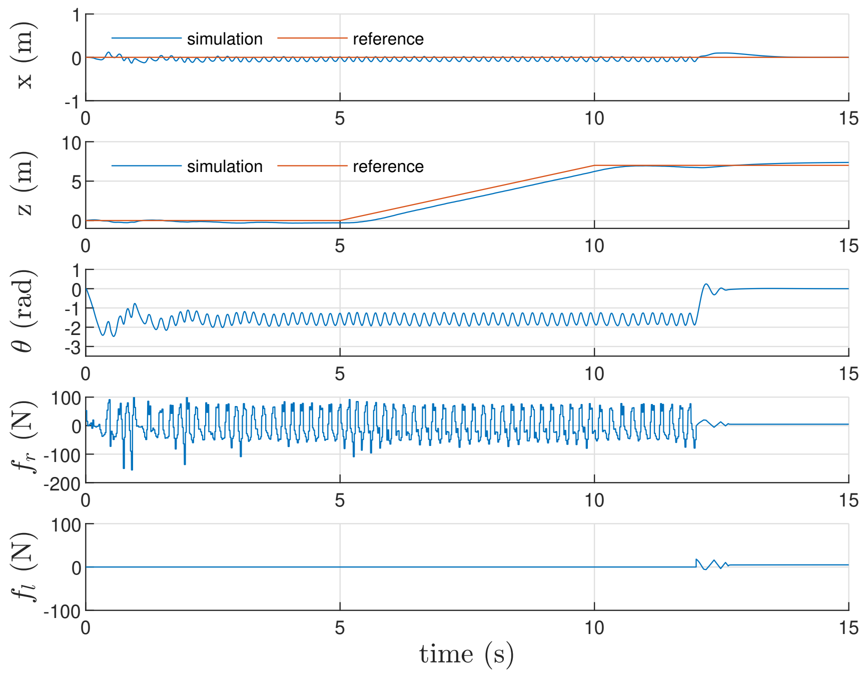
By combining such a controller with the RL controller, a quadcopter could start from a hovering position, switch to the tic-toc flight, and finally switch back to the hovering mode. It should be noted that two forces are required for hovering and only one force is needed for tic-toc. The overall control structure is illustrated in Fig. 6.18.

6.6.3 Network Migration

Developing artificial neural networks and porting them from a learnt environment to a new environment is one of the current research emphases in the field of reinforcement



(a)



(b)

Figure 6.19. Simulation results of passing through vertical obstacles with the extension NN controller, where overall maximum force is 164 N and space occupation is 0.52 m. (a) shows the 2D rainbow movement and (b) shows its feedback states and forces.

learning [180, 181]. There are three different models within the whole learning based control system deployment process:

1. The first model is what the control system has learnt about the world. Because we use a model-free reinforcement learning method, the control system has to learn not only the controller but also the model of the environment (the dynamics of the planar quadcopter in our situation).
2. The second model is what we build in the simulation environment. It is the model used for RL based controller training.
3. The third model is the new validation model. It is a different platform on which we aim the controller to be deployed.

And since the model mismatch exists, apply the trained policy from simulation to a new environment will inevitably generate errors. This chapter provides two different methods to reduce these errors and enhance the policy migration ability.

1. The first aspect is the learning mismatch, where the trained control system can not fully learn the model of the simulation environment. This helps to train the controller to explore more states and situations.
2. The second aspect is the model parameters mismatch, where parameter values of the simulation model may differ from those in the target situation. We can adjust the parameters within Eq. (6.11) to approach the deployment situations in reality.
3. The third mismatch comes from the model structure mismatch between the training environment and the target environment. There are methods that can potentially reduce this mismatch. After deploying the trained controller, data feedback collected from the target flying tests can be used to train the controller to reduce the model mismatch [182].

In our work, we initialize the training agent target at a random position to reduce the first type of mismatches.

To solve the second type of mismatches, we could adjust the force and torque generated by the trained RL controller according to the specific parameters of the target platform to

$$\begin{aligned} f_n &= \frac{m_n}{m_o} f_o, \\ \tau_n &= \frac{l_o I_n}{I_o l_n} \tau_o, \end{aligned} \tag{6.11}$$

where parameters and variables with subscript o and n correspond to the original and new parameters, respectively.

Substituting the migration equations Eq. (6.11) to Eq. (3.5) gives

$$\begin{aligned} f_o \sin\theta / m_o &= f_n \sin\theta / m_n, \\ f_o \cos\theta / m_o &= f_n \cos\theta / m_n, \\ \tau_o l_o / I_o &= \tau_n l_n / I_n. \end{aligned}$$

Therefore, once the parameters of the standard training model and the target model are known, we can migrate the RL controller to the target model to make their performance consistent.

Since the third method requires to use the data collected from real experiments, this method can be considered to reduce the model mismatch in further practical deployment research.

6.6.4 Simulation Validation

We study three simulation scenarios to examine the performance of the integrated system, comprising the RL controller, compensation controller, network migration subsystem and hovering recovery controller.

The simulation is conducted in Simscape, a physical simulation environment in Matlab. We discard aerodynamic forces caused by the obstacles. Table 6.2 lists the physical parameters of the VPP quadcopter model as used in training (with subscript o) and simulation (with subscript n).

In the first scenario, as shown in Fig. 6.20, from $t = 0$ to $t = 5$, the quadcopter switches from a hovering position to a tic-toc flight mode. Starting from $t = 5$, it tracks a moving reference point upward thereby passing a narrow passage. At $t = 12$, it switches successfully back to hovering.

In the second scenario, noises and wind disturbance are considered during validation. Noise with a signal-to-noise ratio of 15 dB is added to all the feedback states. The wind disturbance is 2 N along x-axis and 2 N along z-axis from 3 s to 6 s. Moreover, actuator constraints are added to the simulation where the maximum thrust of the actuator is limited to 25 N and the maximum thrust changing rate is limited to 1000 N/s [183]. The result presented in Fig. 6.21 indicates that our designed control system has good robustness (green lines represent the duration of the wind disturbance).

Table 6.2. Planar quadcopter parameters

Parameters	Symbols	Values	Unit
Original Mass	m_o	1	kg
Mass	m_n	0.665	kg
Gravitational acceleration	g	9.8	m/s^2
Original Moment of inertia	I_o	0.083	kg·m^2
Moment of inertia	I_n	0.007	kg·m^2
Original Length	l_o	1	m
Length	l_n	0.44	m
Force Limit	$f_{\max}$	200	N

In the third scenario, we assume that the parameters of the real system can not be measured accurately. The values of the parameters used in (6.11) are mismatched, with different degrees of uncertainty. In particular, these parameters are sampled from uniform distributions within a mismatch percentage as follows $a_n = a_r(1 + k_a), k_a \in [-b, b]$, where a_r is the real value of the parameter, a_n is the inaccurate value of the parameter used in Eq. (5), k_a is a random variable drawn uniformly from $[-b, b]$, and b is the model parameters mismatch percentage. We test the controller with different model parameters mismatch percentage $b \in \{0.1, 0.2, \ldots, 0.8\}$. For each tested mismatch percentage, we repeated the simulation for 100 episodes and calculate the average success rate. The episode is marked as success when the agent can perform the tic-toc manoeuvrer for 5 seconds. The result is shown in Fig. 6.22. It indicates that our trained controller reliably performs the tic-toc manoeuvre if the mismatch is not large. This suggests that the reinforcement learning based control system has certain generalization ability to handle model mismatches.

The findings demonstrate the potential application of the proposed control approach. To the best of our knowledge, no other methods has previously achieved autonomous tic-toc manoeuvrer with a quadcopter model.

6.7 Summary

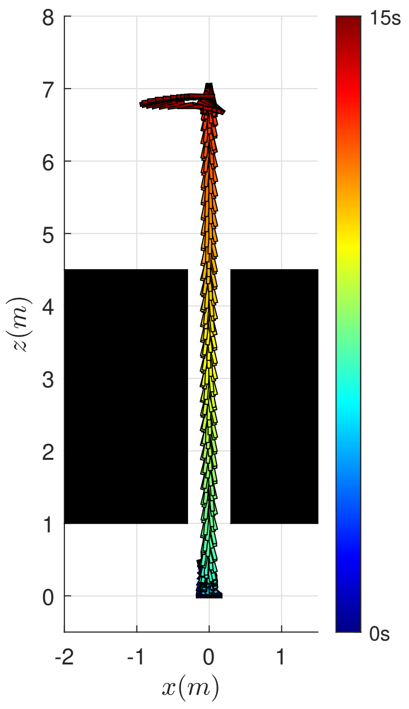
In this chapter, we employ the DDPG approach to train RL controllers based on well-designed rewards. While the obtained RL controllers could generate two flight modes, spin and tic-toc, we further analyse the properties of these flight modes and screen unfavourable RL controllers out by multi-objective optimization. Finally, the RL controller is strengthened by additional PID and LQR controllers to achieve better flight performance such as tracking a moving reference point and recovering to hovering flight status. The proposed control system is validated by a quadcopter in a physical environment named Simscape. Although the tic-toc manoeuvrer controller is designed in the 2D frame, the experiments were simulated in a simplified 3D environment with no noise or errors. The results demonstrate that the designed controller can achieve our manoeuvrer target and can be further extended. Although this is the first step to achieve tic-toc manoeuvrer control, there are still many challenges to overcome in order to deploy such a controller in reality.

Alternative solutions for real implementation could include 1) an agile quadcopter platform that includes the dynamics, perception and control system, and 2) appropriate methods to extend the controller from 2D environment to 3D environment.

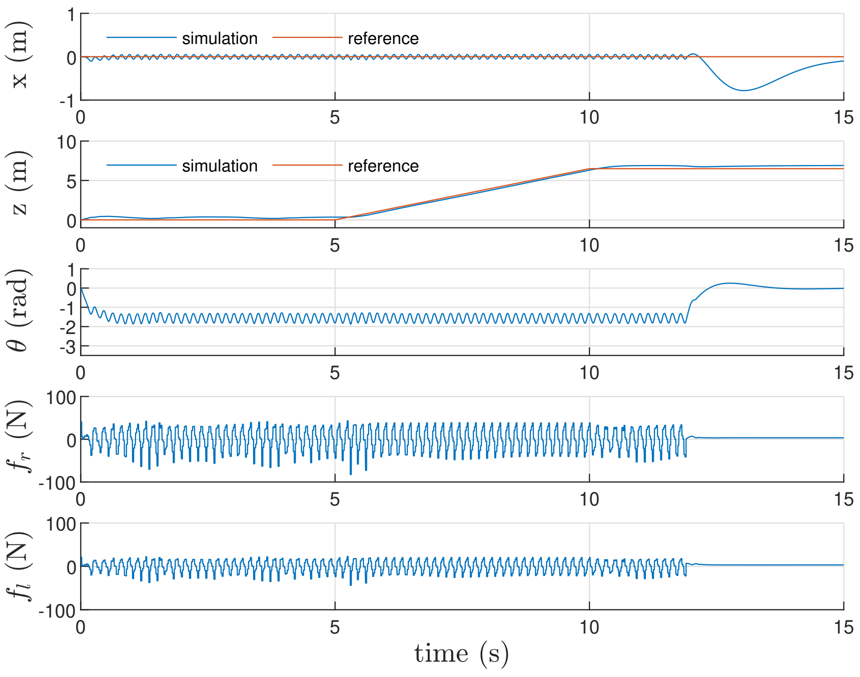
1) As mentioned in Section 2.1, quadcopter with rapid dynamic response and corresponding accurate sensing system is a key concern in actual implementation. In particular, the SVPP quadcopter is more suitable for manoeuvring than the CVPP quadcopter, because it can provide precise and independent thrust and torque control.

2) Methods like ILC and deep RL can be efficient to extend the controller to adapt to the practical 3D environment (e.g. [103]). Since tic-toc manoeuvrer is an iterative motion, other learning-based methods (such as RL) can also be used to extend the controller to stabilize the quadcopter flying in the air.

In summary, the main theoretical contribution of this chapter is to present for the first time an RL NN controller which was trained on a planar quadcopter model to successfully perform the tic-toc manoeuvre. Moreover, we extended the controller and demonstrated its ability to perform position tracking and narrow vertical tunnel passage in a simulation environment.



(a)



(b)

Figure 6.20. Simulation results of passing through vertical obstacles with the NN migration method, where the overall maximum force was 84 N and maximum space occupation was 0.4 m. (a) shows the 2D rainbow movement and (b) shows its feedback states and forces.

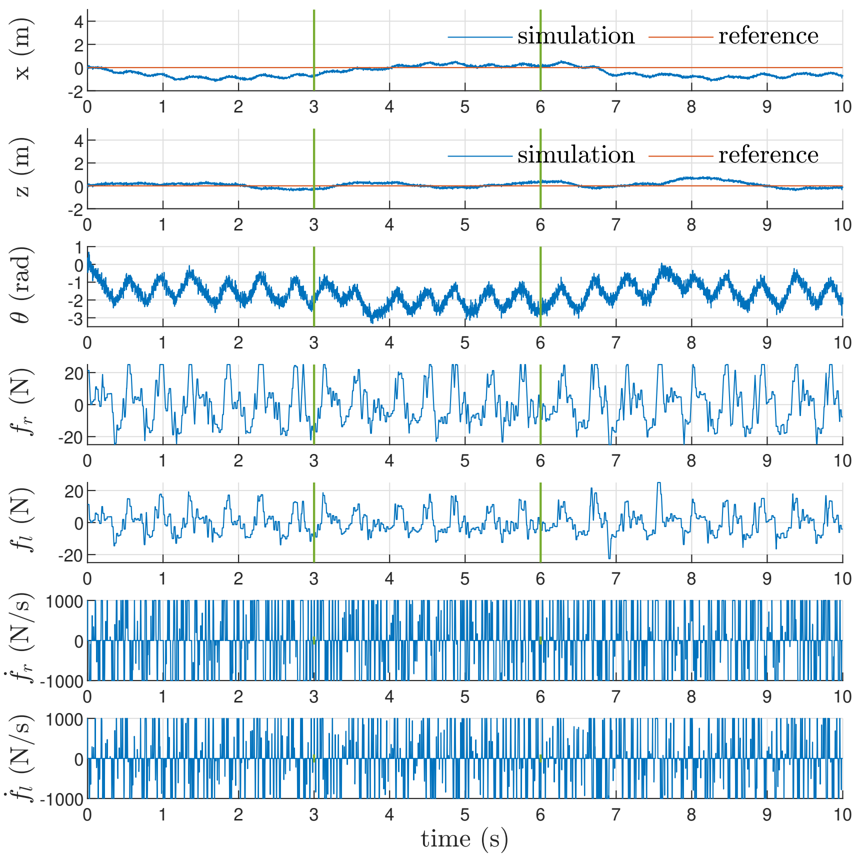


Figure 6.21. Results of a VPP quadcopter in the tic-toc mode facing external disturbances and noises.

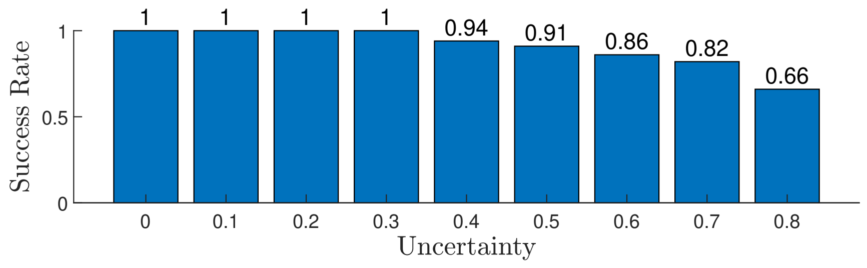


Figure 6.22. Results of a VPP quadcopter holding its position with tic-toc mode under different parameter mismatches.

Chapter 7

Summary and Future Work

7.1 Thesis Summary

The VPP quadcopter is a promising aerial platform with many potential applications. However, the existing research has not fully explored its capabilities. In this study, we investigate one theoretical contribution and three novel control systems that enhance the flight safety and agility of VPP quadcopters.

The first novel theoretical contribution of this study is to discover that the thrust and torque control of a VPP quadcopter actuator can be independently controlled, as presented in Chapter 3. This enhancement lays a foundation for the subsequent VPP quadcopter flight safety control.

Another novel concept is the design of a fault tolerance controller for SVPP quadcopters. We find that an SVPP quadcopter can fly with three available actuators. Therefore, as described in Chapter 4, we design a fault tolerance control system for an SVPP quadcopter with one inoperative actuator. Simulations indicate that a faulty SVPP quadcopter system with three available actuators can accurately track the desired trajectory, even when subject to wind disturbance or noise. Subsequent demonstrations verify that an SVPP quadcopter using our new controller possesses great potential for safety-critical tasks.

Additionally, in this thesis, we study the case where one propeller of a CVPP quadcopter is stuck and cannot adjust its pitch angle. This is one of the most common failures that occurs in CVPP quadcopters and it is also a unique fault that only occurs on VPP quadcopters. A unique fault tolerance control system is devised after decomposing the uncontrollable modes from the system. In Chapter 5, we describe how the proposed controller is validated in a simulated environment to demonstrate its stabilising ability on the faulty CVPP quadcopter.

Finally, we investigate the agility control of VPP quadcopters in Chapter 6. This chapter includes tic-toc manoeuvre metrics analysis, self-learning control system design, and systematic solutions for control performance adaption. Results from simulations show that a planar quadcopter can perform tic-toc manoeuvres in the simulated environment. The simulation results also prove that the learning-based control system we designed can enhance the flight agility of VPP quadcopters with good generalisation ability. It also provides the additional possibility for an autonomous autopilot system to learn manoeuvres by itself without human demonstration.

However, the design of this control system is based on the assumption that a planar quadcopter can be considered as a simplified real quadcopter. It is difficult to implement this assumption in reality, and the findings of the current study are not a perfect solution for actual VPP quadcopters to perform the tic-toc manoeuvre. Nevertheless, the proposed method, an RL-based controller that can achieve such a manoeuvre within a 2D frame, is one step closer to demonstrating that the manoeuvre is possible in practice.

In this study, we demonstrate that the VPP quadcopter is an aircraft with both agility and safety potential. Although the mechanical structure of the VPP quadcopter is more complicated than conventional models, the complexity can be overcome by designing appropriate control systems. We believe that our work provides a good foundation for realising the safe and agile flight of VPP quadcopters in the future.

7.2 Future Work

In this thesis, we have proposed several different methods that may improve the flight performance of the VPP quadcopter. However, some of them have not been fully investigated and deserve further research, such as optimal control for the power distribution of a VPP quadcopter and practical validation experiments for existing algorithms. Therefore, in the future, the following research directions may be studied:

1. In the VPP dynamic model described in Chapter 3, we only use second-order polynomial equations to model the thrust and torque dynamics. In the future, we can use machine learning algorithms to identify more complex models using data that have been collected.

2. In Chapter 4, we confirm that an SVPP quadcopter can maintain its position when only three actuators are functional. Although the analysis in Chapter 4 is based on a manually selected actuator spinning speed, the spinning speed can actually be selected from an interval. Further research can analyse how to optimally select this spinning speed to maximise the endurance of the VPP quadcopter.

3. Although the actuator rotation speed is considered as a fixed value for the CVPP quadcopter in Chapter 5, some CVPP quadcopters can vary their speed. For this case, we can design new controllers that adjust the propeller speed to reduce the yaw angle offset. Such a method can further improve safety performance.

4. In Chapter 6, we propose an extension method that can migrate the tic-toc manoeuvre to another quadcopter. This method also adjusts the tic-toc manoeuvre performance by setting different parameter values. Further research can use algorithms to optimise the parameter values and delicately adjust the manoeuvrability. As a result, the control system can be employed in applications with different requirements, such as maximum thrust or horizontal occupation limitation.

5. We present the tic-toc manoeuvre of a VPP quadcopter within the 2D plane in Chapter 6. This is the basis for accomplishing the tic-toc manoeuvre in a practical environment. With further research, we can transfer the designed control system from a 2D simulation environment to a 3D environment. Besides, methods such as RL or iterative learning control can be applied.

Overall, we believe that the approaches, designs, implementation, and analysis demonstrated in this thesis pave the way for the future realisation of VPP quadcopters.

Bibliography

[1] Andrew Zulu and Samuel John. A review of control algorithms for autonomous quadrotors. *Open Journal of Applied Sciences*, 4:547–556, 2014.

[2] Mark W Mueller, Seung Jae Lee, and Raffaello D'Andrea. Design and control of drones. *Annual Reviews*, 5:1–18, 2021.

[3] Pierre-Jean Bristeau, François Callou, David Vissiere, Nicolas Petit, et al. The navigation and control technology inside the AR. drone micro UAV. In *Proceedings of the 18th IFAC world congress*, volume 18, pages 1477–1484, 2011.

[4] Jeremy H Gillula, Haomiao Huang, Michael P Vitus, and Claire J Tomlin. Design and analysis of hybrid systems, with applications to robotic aerial vehicles. In *Robotics Research*, pages 139–149. Springer, 2011.

[5] S Norouzi Ghazbi, Y Aghli, M Alimohammadi, and AA Akbari. Quadrotors unmanned aerial vehicles: A review. *International Journal on Smart Sensing & Intelligent Systems*, 9(1), 2016.

[6] Chris A Wargo, Gary C Church, Jason Glaneueski, and Mark Strout. Unmanned aircraft systems (UAS) research and future analysis. In *Proceedings of 2014 IEEE Aerospace Conference*, pages 1–16. IEEE, 2014.

[7] Antonio Barrientos, Julian Colorado, Jaime del Cerro, Alexander Martinez, Claudio Rossi, David Sanz, and João Valente. Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. *Journal of Field Robotics*, 28(5):667–689, 2011.

[8] Nathan Michael, Shaojie Shen, Kartik Mohta, Vijay Kumar, Keiji Nagatani, Yoshito Okada, Seiga Kiribayashi, Kazuki Otake, Kazuya Yoshida, Kazunori Ohno, et al. Collaborative mapping of an earthquake damaged building via ground and aerial robots. In *Proceedings of Field and Service Robotics*, pages 33–47, 2014.

[9] Kimon P Valavanis and George J Vachtsevanos. Future of unmanned aviation. In *Handbook of unmanned aerial vehicles*, pages 2993–3009. Springer, 2015.

[10] LYU Xin-ming. Military UAV development and countermeasures. *National Defense Science and Technology*, 1:003, 2013.

[11] PY Huang, LR Zeng, Chuan Yang, YX Peng, and CB Yu. Design of a new style four-axis aerial photography aircraft for disaster relief. *J. Sichuan Ordnance*, 6 (1):035, 2014.

[12] Vinay Pandit and Arun Poojari. A study on amazon prime air for feasibility and profitability: A graphical data analysis. *IOSR Journal of Business and Management*, 16(11):06–11, 2014.

[13] Alec Momont. Alec Momont, 2018. URL http://www.alecmomont.com/projects/dronesforgood. Accessed: 2018-12-18.

[14] Ehang. Ehang 184, 2018. URL http://http://ehang.com/ehang184/index. Accessed: 2018-12-18.

[15] Christopher Silva, Wayne R Johnson, Eduardo Solis, Michael D Patterson, and Kevin R Antcliff. Vtol urban air mobility concept vehicles for technology development. In *2018 Aviation Technology, Integration, and Operations Conference*, page 3847, 2018.

[16] Arne Devos, Emad Ebeid, and Poramate Manoonpong. Development of autonomous drones for adaptive obstacle avoidance in real world environments. In *Proceedings of 2018 21st Euromicro Conference on Digital System Design (DSD)*, pages 707–710, 2018.

[17] Ziquan Yu, Youmin Zhang, Bin Jiang, Chun Yi Su, Jun Fu, Ying Jin, and Tianyou Chai. Decentralized fractional-order backstepping fault-tolerant control of multi-UAVs against actuator faults and wind effects. *Aerospace Science and Technology*, 104:105939, 2020.

[18] Shidong Xu, Hao Wen, and Zheng Huang. Robust fuzzy sampled-data attitude control of spacecraft with actuator saturation and persistent disturbance. *Aerospace Science and Technology*, 101:105850, 2020.

[19] Sam Allison, He Bai, and Balaji Jayaraman. Wind estimation using quadcopter motion: A machine learning approach. *Aerospace Science and Technology*, 98: 105699, 2020.

[20] Mark Cutler and Jonathan How. Actuator constrained trajectory generation and control for variable-pitch quadrotors. In *Proceedings of AIAA Guidance, Navigation, and Control Conference*, pages 4777–4792, 2012.

[21] Mark Cutler, Nazim-Kemal Ure, Bernard Michini, and Jonathan How. Comparison of fixed and variable pitch actuators for agile quadrotors. In *Proceedings of AIAA Guidance, Navigation, and Control Conference*, pages 6406–6423, 2011.

[22] Mark Cutler and Jonathan P How. Analysis and control of a variable-pitch quadrotor for agile flight. *Journal of Dynamic Systems, Measurement, and Control*, 137(10):101002–101016, 2015.

[23] Namrata Gupta, Mangal Kothari, and Abhishek. Flight dynamics and nonlinear control design for variable-pitch quadrotors. In *Proceedings of 2016 American Control Conference (ACC)*, pages 3150–3155. IEEE, 2016.

[24] Shouzhao Sheng and Chenwu Sun. Control and optimization of a variable-pitch quadrotor with minimum power consumption. *Energies*, 9(4):232, 2016.

[25] Remus C Avram, Xiaodong Zhang, and Jonathan Muse. Nonlinear adaptive fault-tolerant quadrotor altitude and attitude tracking with multiple actuator faults. *IEEE Transactions on Control Systems Technology*, 26(2):701–707, 2017.

[26] Mark W Mueller and Raffaello D'Andrea. Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. In *Proceedings of 2014 IEEE International Conference on Robotics and Automation*, pages 45–52, 2014.

[27] Iman Sadeghzadeh and Youmin Zhang. A review on fault-tolerant control for unmanned aerial vehicles (UAVs). In *Infotech@ Aerospace 2011*, page 1472, 2011.

[28] Tiago P Nascimento and Martin Saska. Position and attitude control of multi-rotor aerial vehicles: A survey. *Annual Reviews in Control*, 48:129–146, 2019.

[29] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012.

[30] Mark Müller, Sergei Lupashin, and Raffaello D'Andrea. Quadrocopter ball juggling. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5113–5120, 2011.

[31] Dario Brescianini, Markus Hehn, and Raffaello D'Andrea. Quadrocopter pole acrobatics. In *Proceedings of the Intelligent Robots and Systems International Conference*, pages 3472–3479, 2013.

[32] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction.* MIT press, 1998.

[33] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *Proceedings of 2016 International Conference on Learning Representations (ICLR)*, 2016.

[34] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

[35] Pieter Abbeel, Adam Coates, and Andrew Y Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010.

[36] Jemin Hwangbo, Inkyu Sa, Roland Siegwart, and Marco Hutter. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103, 2017.

[37] William Koch, Renato Mancuso, Richard West, and Azer Bestavros. Reinforcement learning for UAV attitude control. *ACM Transactions on Cyber-Physical Systems*, 3(2):1–21, 2019.

[38] Josh Villbrandt et al. The quadrotor's coming of age. *Illumin. University of Southern California*, 2011.

[39] Kenneth Munson and John William Wood. *Helicopters and other rotorcraft since 1907.* Blandford, 1968.

[40] Giorgio Apostolo. *The illustrated encyclopedia of helicopters.* Bonanza Books New York, 1984.

[41] Shweta Gupte, Paul Infant Teenu Mohandas, and James M Conrad. A survey of quadrotor unmanned aerial vehicles. In *Proceedings of the 2012 IEEE Southeastcon,*, pages 1–6, 2012.

[42] Martin Detert and Volker Weitbrecht. A low-cost airborne velocimetry system: proof of concept. *Journal of Hydraulic Research*, 53(4):532–539, 2015.

[43] M Sakti Alvissalim, Big Zaman, Z Ahmad Hafizh, M Anwar Ma'sum, Grafika Jati, Wisnu Jatmiko, and Petrus Mursanto. Swarm quadrotor robots for telecommunication network coverage area expansion in disaster area. In *Proceedings of the SICE Annual Conference*, pages 2256–2261, 2012.

[44] Chunhua Sheng and Jim C Narramore. Computational simulation and analysis of bell boeing quad tiltrotor aero interaction. *Journal of the American Helicopter Society*, 54(4):42002–42002, 2009.

[45] W Dale Brown. Dusty's flying taxi. *Carolina Quarterly*, 46(2):38, 1994.

[46] Gabe Hoffmann, Dev Gorur Rajnarayan, Steven L Waslander, David Dostal, Jung Soon Jang, and Claire J Tomlin. The stanford testbed of autonomous rotorcraft for multi agent control (STARMAC). In *Proceedings of The 23rd Digital Avionics Systems Conference*, volume 2, pages 12–E. IEEE, 2004.

[47] Haomiao Huang, Gabriel M Hoffmann, Steven L Waslander, and Claire J Tomlin. Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In *Proceedings of 2009 IEEE International Conference on Robotics and Automation*, pages 3277–3282, 2009.

[48] Jeremy H Gillula, Haomiao Huang, Michael P Vitus, and Claire J Tomlin. Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice. In *Proceedings of 2010 IEEE International Conference on Robotics and Automation*, pages 1649–1654. IEEE, 2010.

[49] Sergei Lupashin, Angela Schöllig, Michael Sherback, and Raffaello D'Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, pages 1642–1648, 2010.

[50] Markus Hehn and Raffaello D'Andrea. A flying inverted pendulum. In *Proceedings of 2011 IEEE International Conference on Robotics and Automation*, pages 763–770. IEEE, 2011.

[51] Sergei Lupashin, Markus Hehn, Mark W Mueller, Angela P Schoellig, Michael Sherback, and Raffaello D'Andrea. A platform for aerial robotics research and demonstration: The flying machine arena. *Mechatronics*, 24(1):41–54, 2014.

[52] Robin Ritz, Mark W Müller, Markus Hehn, and Raffaello D'Andrea. Cooperative quadrocopter ball throwing and catching. In *Proceedings of 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4972–4978. IEEE, 2012.

[53] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.

[54] Markus Hehn and Raffaello D'Andrea. Quadrocopter trajectory generation and control. *IFAC Proceedings Volumes*, 44(1):1485–1491, 2011.

[55] Hikaru Otsuka. Stability of unmanned aerial vehicles in heavy winds can be improved through rotor placement and angle, according to a team from Tohoku University and Kanazawa Institute of Technology, 2018. URL https://www.sciencedaily.com/releases/2018/02/180209100730.htm. Accessed: 2018-12-18.

[56] Steven Yoon, William M Chan, and Thomas H Pulliam. Computations of torque-balanced coaxial rotor flows. In *Proceedings of 55th AIAA Aerospace Sciences Meeting*, page 0052, 2017.

[57] Adam Bondyra, Stanisław Gardecki, Przemysław Gasior, and Wojciech Giernacki. Performance of coaxial propulsion in design of multi-rotor UAVs. In *Proceedings of International Conference on Automation*, pages 523–531. Springer, 2016.

[58] DJI. DJI INSPIRE 2, 2018. URL https://www.dji.com/uk/inspire-2. Accessed: 2018-12-18.

[59] 3DR. 3DR X-8 Quadcopter, 2021. URL https://uavsystemsinternational.com/products/3d-robotics-x8. Accessed: 2021-08-18.

[60] Align. TREX 450 PRO Helicopter, 2021. URL http://www.align.com.tw/helicopter/trex450/.

[61] Xiaonan Wu. *Design and Development of Variable Pitch Quadcopter for Long Endurance Flight*. PhD thesis, Oklahoma State University, 2018.

[62] Pang Tao. *Design, prototyping and autonomous control of gasoline-engine variable-pitch quadcopter*. PhD thesis, National University of Singapore (Singapore), 2016.

[63] Michael E McKay, Robert Niemiec, and Farhan Gandhi. Performance comparison of quadcopters with variable-rpm and variable-pitch rotors. *Journal of the American Helicopter Society*, 64(4):1–14, 2019.

[64] M. Lovera F. Riccardi, P. Panizza. Identification of the attitude dynamics for a variable-pitch quadrotor uav. https://home.aero.polimi.it/lovera/ea/5_2.pdf, 2014. Accessed: 2021–08-14.

[65] Karl Johan Åström and Tore Hägglund. *PID controllers: theory, design, and tuning*, volume 2. Instrument society of America Research Triangle Park, NC, 1995.

[66] A Tayebi and S McGilvray. Attitude stabilization of a four-rotor aerial robot. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, volume 2, pages 1216–1221, 2004.

[67] Tim Puls and Andreas Hein. 3d trajectory control for quadrocopter. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 640–645. IEEE, 2010.

[68] İ Can Dikmen, Aydemir Arisoy, and Hakan Temeltas. Attitude control of a quadrotor. In *Recent Advances in Space Technologies, 2009. RAST'09. 4th International Conference on*, pages 722–727, 2009.

[69] Deepak Gautam and Cheolkeun Ha. Control of a quadrotor using a smart self-tuning fuzzy pid controller. *International Journal of Advanced Robotic Systems*, 10(11):380, 2013.

[70] Mohammad Hadi Amoozgar, Abbas Chamseddine, and Youmin Zhang. Fault-tolerant fuzzy gain-scheduled pid for a quadrotor helicopter testbed in the presence of actuator faults. *IFAC Proceedings Volumes*, 45(3):282–287, 2012.

[71] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[72] Mehmet Onder Efe. Neural network assisted computationally simple pid control of a quadrotor UAV. *IEEE Transactions on Industrial Informatics*, 7 (2):354–361, 2011.

[73] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes.* Routledge, 2018.

[74] Arthur E Bryson. Optimal control-1950 to 1985. *IEEE Control Systems*, 16 (3):26–33, 1996.

[75] LA Sanchez, Omar Santos, Hugo Romero, Sergio Salazar, and Rogelio Lozano. Nonlinear and optimal real-time control of a rotary-wing UAV. In *Proceedings of 2012 American Control Conference (ACC)*, pages 3857–3862, 2012.

[76] Agus Budiyono and Singgih S Wibowo. Optimal tracking controller design for a small scale helicopter. *Journal of bionic engineering*, 4(4):271–280, 2007.

[77] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

[78] Kaan Taha Öner, Ertuğrul Çetinsoy, Mustafa Ünel, Mahmut Faruk Akşit, Ilyas Kandemir, and Kayhan Gülez. Dynamic model and control of a new quadrotor unmanned aerial vehicle with tilt-wing mechanism. *World Academy of Science, Engineering and Technology (WASET)*, 2008.

[79] Lucas M Argentim, Willian C Rezende, Paulo E Santos, and Renato A Aguiar. Pid, lqr and lqr-pid on a quadcopter platform. In *Proceedings of the 2013 International Conference on Informatics, Electronics & Vision (ICIEV)*, pages 1–6, 2013.

[80] Samir Bouabdallah, Andre Noth, and Roland Siegwart. Pid vs lq control techniques applied to an indoor micro quadrotor. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004.(IROS 2004)*, volume 3, pages 2451–2456, 2004.

[81] Shahida Khatoon, Dhiraj Gupta, and LK Das. Pid & lqr control for a quadrotor: Modeling and simulation. In *Proceedings of Communications and Informatics (ICACCI, 2014 International Conference on Advances in Computing*, pages 796–802, 2014.

[82] Huibert Kwakernaak and Raphael Sivan. *Linear optimal control systems*, volume 1. Wiley-Interscience New York, 1972.

[83] Yann Ameho, Fabien Niel, François Defaÿ, Jean-Marc Biannic, and Caroline Bérard. Adaptive control for quadrotors. In *Proceedings of 2013 IEEE International Conference on Robotics and Automation*, pages 5396–5401. IEEE, 2013.

[84] Gabriel Hoffmann, Haomiao Huang, Steven Waslander, and Claire Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Proceedings of Navigation and Control Conference and Exhibit AIAA Guidance*, page 6461, 2007.

[85] M Elena Antonio-Toledo, Edgar N Sanchez, Alma Y Alanis, JA Flórez, and Marco A Perez-Cisneros. Real-time integral backstepping with sliding mode control for a quadrotor uav. *IFAC-PapersOnLine*, 51(13):549–554, 2018.

[86] Hassan K Khalil and JW Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

[87] Abdellah Mokhtari, Nacer K M'Sirdi, Kamal Meghriche, and Abdelkader Belaidi. Feedback linearization and linear observer for a quadrotor unmanned aerial vehicle. *Advanced Robotics*, 20(1):71–91, 2006.

[88] V Mistler, A Benallegue, and NK M'sirdi. Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback. In *Proceedings of 10th IEEE International Workshop on Robot and Human Interactive Communication*, pages 586–593, 2001.

[89] Holger Voos. Nonlinear control of a quadrotor micro-UAV using feedback-linearization. In *Proceedings of 2009 IEEE International Conference on Mechatronics*, pages 1–6, 2009.

[90] Giulia Michieletto, Angelo Cenedese, Luca Zaccarian, and Antonio Franchi. Nonlinear control of multi-rotor aerial vehicles based on the zero-moment direction. *IFAC-PapersOnLine*, 50(1):13144–13149, 2017.

[91] Angela P Schoellig, Fabian L Mueller, and Raffaello D'Andrea. Optimization-based iterative learning for precise quadrocopter trajectory tracking. *Autonomous Robots*, 33(1-2):103–127, 2012.

[92] Ian D Cowling, James F Whidborne, and Alastair K Cooke. Optimal trajectory planning and lqr control for a quadrotor UAV. In *Proceedings of the International Conference on Control*, 2006.

[93] Alessandro Gasparetto, Paolo Boscariol, Albano Lanzutti, and Renato Vidoni. Path planning and trajectory planning algorithms: A general overview. *Motion and operation planning of robotic systems*, pages 3–27, 2015.

[94] Youmin Zhang and Jin Jiang. Bibliographical review on reconfigurable fault-tolerant control systems. *Annual reviews in control*, 32(2):229–252, 2008.

[95] Alessandro Freddi, Alexander Lanzon, and Sauro Longhi. A feedback linearization approach to fault tolerance in quadrotor vehicles. *IFAC proceedings volumes*, 44(1):5413–5418, 2011.

[96] Jihad Ghandour, Samir Aberkane, and Jean-Christophe Ponsart. Feedback linearization approach for standard and fault tolerant control: Application to a quadrotor UAV testbed. In *Proceedings of Journal of Physics: Conference Series*, volume 570, page 082003. IOP Publishing, 2014.

[97] Chaofang Hu, Xianpeng Zhou, Binghan Sun, Wenjing Liu, and Qun Zong. Nussbaum-based fuzzy adaptive nonlinear fault-tolerant control for hypersonic vehicles with diverse actuator faults. *Aerospace Science and Technology*, 71: 432–440, 2017.

[98] Farid Sharifi, Mostafa Mirzaei, Brandon W Gordon, and Youmin Zhang. Fault tolerant control of a quadrotor UAV using sliding mode control. In *2010 Conference on Control and Fault-Tolerant Systems*, pages 239–244, 2010.

[99] Abbas Chamseddine, Didier Theilliol, YM Zhang, Cédric Join, and Camille-Alain Rabbath. Active fault-tolerant control system design with trajectory re-planning against actuator faults and saturation: Application to a quadrotor unmanned aerial vehicle. *International Journal of Adaptive Control and Signal Processing*, 29(1):1–23, 2015.

[100] Yongduan Song, Liu He, Dong Zhang, Jiye Qian, and Jin Fu. Neuroadaptive fault-tolerant control of quadrotor UAVs: A more affordable solution. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1975–1983, 2018.

[101] Mien Van. An enhanced robust fault tolerant control based on an adaptive fuzzy pid-nonsingular fast terminal sliding mode control for uncertain nonlinear systems. *IEEE/ASME Transactions on Mechatronics*, 23(3):1362–1371, 2018.

[102] Sihao Sun, Leon Sijbers, Xuerui Wang, and Coen de Visser. High-speed flight of quadrotor despite loss of single rotor. *IEEE Robotics and Automation Letters*, 3(4):3201–3207, 2018.

[103] Oliver Purwin and Raffaello D'Andrea. Performing and extending aggressive maneuvers using iterative learning control. *Robotics and Autonomous Systems*, 59(1):1–11, 2011.

[104] Robin Ritz, Markus Hehn, Sergei Lupashin, and Raffaello D'Andrea. Quadrocopter performance benchmarking using optimal control. In *Proceedings of 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5179–5186. IEEE, 2011.

[105] Christopher G Mayhew, Ricardo G Sanfelice, and Andrew R Teel. On quaternion-based attitude control and the unwinding phenomenon. In *Proceedings of the 2011 American Control Conference*, pages 299–304. IEEE, 2011.

[106] Taeyoung Lee. Robust adaptive attitude tracking on SO (3) with an application to a quadrotor UAV. *IEEE Transactions on Control Systems Technology*, 21 (5):1924–1930, 2012.

[107] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Geometric tracking control of a quadrotor UAV on SE (3). In *Proceedings of 49th IEEE conference on decision and control (CDC)*, pages 5420–5425. IEEE, 2010.

[108] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Nonlinear robust tracking control of a quadrotor UAV on SE(3). *Asian Journal of Control*, 15 (2):391–408, 2013.

[109] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles. *IEEE Robotics and Automation magazine*, 20(32), 2012.

[110] Gabriel Hoffmann, Steven Waslander, and Claire Tomlin. Quadrotor helicopter trajectory tracking control. In *Proceedings of the AIAA guidance, navigation and control conference and exhibit*, page 7410, 2008.

[111] Douglas A Bristow, Marina Tharayil, and Andrew G Alleyne. A survey of iterative learning control. *IEEE Control Systems*, 26(3):96–114, 2006.

[112] Mohammad Shaqura and Jeff S Shamma. An iterative learning approach for motion control and performance enhancement of quadcopter UAVs. In *Proceedings of 2018 18th International Conference on Control, Automation and Systems (ICCAS)*, pages 285–290. IEEE, 2018.

[113] Kevin L Moore. *Iterative learning control for deterministic systems.* Springer Science & Business Media, 2012.

[114] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in neural information processing systems*, pages 1–8, 2007.

[115] Nicola Jones. Computer science: The learning machines. *Nature News*, 505 (7482):146, 2014.

[116] M Moravčík, M Schmid, N Burch, V Lis, D Morrill, N Bard, T Davis, K Waugh, M Johanson, and M Bowling. Deepstack: expert-level artificial intelligence in no-limit poker. *arXiv preprint arXiv:1701.01724*, 2017.

[117] Tesla. Tesla Autopilot, 2021. URL https://www.tesla.com/autopilotAI. Accessed: 2021-08-18.

[118] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *Proceedings of 2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[119] Somil Bansal, Anayo K Akametalu, Frank J Jiang, Forrest Laine, and Claire J Tomlin. Learning quadrotor dynamics using neural network for flight control. In *Proceedings of 2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4653–4660, 2016.

[120] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

[121] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, 6(5):679–684, 1957.

[122] JAEE Van Nunen. A set of successive approximation methods for discounted markovian decision problems. *Zeitschrift fuer operations research*, 20(5):203–208, 1976.

[123] Richard Bellman. Dynamic programming and stochastic control processes. *Information and control*, 1(3):228–239, 1958.

[124] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*, 2020.

[125] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[126] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[127] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[128] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards.* PhD thesis, King's College, Cambridge, 1989.

[129] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8 (3-4):279–292, 1992.

[130] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, England, 1994.

[131] Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pages 363–372. Springer, 2006.

[132] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[133] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[134] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[135] Fei-Yue Wang, Jun Jason Zhang, Xinhu Zheng, Xiao Wang, Yong Yuan, Xiaoxiao Dai, Jie Zhang, and Liuqing Yang. Where does alphago go: From church-turing thesis to alphago thesis and beyond. *IEEE/CAA Journal of Automatica Sinica*, 3(2):113–120, 2016.

[136] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[137] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the Advances in neural information processing systems*, pages 1057–1063, 2000.

[138] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Proceedings of the Advances in neural information processing systems*, pages 1008–1014, 2000.

[139] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International conference on machine learning*, pages 1928–1937, 2016.

[140] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*, 2015.

[141] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of International Conference on Machine Learning (ICML)*, 2014.

[142] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[143] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning*, pages 1889–1897, 2015.

[144] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[145] Nicolas Heess, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[146] Alex Irpan. Deep reinforcement learning doesn't work yet. https://www.alexirpan.com/2018/02/14/rl-hard.html, 2018.

[147] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[148] Anežka Chovancová, Tomáš Fico, L'uboš Chovanec, and Peter Hubinsk. Mathematical modelling and parameter identification of quadrotor (a survey). *Procedia Engineering*, 96:172–181, 2014.

[149] Teppo Luukkonen. Modelling and control of quadcopter. *Independent research project in applied mathematics, Espoo*, 22, 2011.

[150] Blake Moffitt, Thomas Bradley, David Parekh, and Dimitri Mavris. Validation of vortex propeller theory for UAV design with uncertainty analysis. In *Proceedings of 46th AIAA Aerospace Sciences Meeting and Exhibit*, page 406, 2008.

[151] Monal Merchant and L Scott Miller. Propeller performance measurement for low reynolds number uav applications. In *Proceedings of 44th AIAA Aerospace Sciences Meeting and Exhibit*, pages 1127–1182, 2006.

[152] Egbert Torenbeek and H Wittenberg. *Flight physics: essentials of aeronautical disciplines and technology, with historical notes.* Springer Science & Business Media, 2009.

[153] P. Bristeau, P. Martin, E. Salaün, and N. Petit. The role of propeller aerodynamics in the model of a quadrotor UAV. In *Proceedings of 2009 European Control Conference*, pages 683–688, Aug 2009.

[154] Mark Drela. XFOIL, subsonic airfoil development system, 2013. URL https://web.mit.edu/drela/Public/web/xfoil/. Accessed: 2021-08-18.

[155] Mark Drela. QPROP, propeller analysis and design, 2007. URL http://web.mit.edu/drela/Public/web/qprop. Accessed: 2021-08-18.

[156] CJ Youngblood Ent. Stingray 500, 2016. URL http://www.curtisyoungblood.com/legacy-product-support-curtis-youngblood/attachment/stingray-500/. Accessed: 2019-09-1.

[157] Gregory L Baker, James A Blackburn, et al. *The pendulum: a case study in physics.* Oxford University Press, 2005.

[158] Zhixiang Liu, Chi Yuan, Youmin Zhang, and Jun Luo. A learning-based fault tolerant tracking control of an unmanned quadrotor helicopter. *Journal of Intelligent & Robotic Systems*, 84(1-4):145–162, 2016.

[159] Fuyang Chen, Rongqiang Jiang, Kangkang Zhang, Bin Jiang, and Gang Tao. Robust backstepping sliding-mode control and observer-based fault estimation for a quadrotor UAV. *IEEE Transactions on Industrial Electronics*, 63(8): 5044–5056, 2016.

[160] Yujiang Zhong, Zhixiang Liu, Youmin Zhang, Wei Zhang, and Junyi Zuo. Active fault-tolerant tracking control of a quadrotor with model uncertainties and actuator faults. *Frontiers of Information Technology & Electronic Engineering*, 20(1):95–106, 2019.

[161] Xin Qi, Juntong Qi, Didier Theilliol, Dalei Song, Youmin Zhang, and Jianda Han. Self-healing control design under actuator fault occurrence on single-rotor unmanned helicopters. *Journal of Intelligent & Robotic Systems*, 84(1-4):21–35, 2016.

[162] Mark W Mueller and Raffaello D'Andrea. Relaxed hover solutions for multicopters: Application to algorithmic redundancy and novel vehicles. *The International Journal of Robotics Research*, 35(8):873–889, 2016.

[163] Abdel-Razzak Merheb, Hassan Noura, and François Bateman. Emergency control of ar drone quadrotor UAV suffering a total loss of one rotor. *IEEE/ASME Transactions on Mechatronics*, 22(2):961–971, 2017.

[164] Maryamsadat Tahavori and Agus Hasan. Fault recoverability for nonlinear systems with application to fault tolerant control of UAVs. *Aerospace Science and Technology*, 107:106282, 2020.

[165] Remus C Avram, Xiaodong Zhang, and Jonathan Muse. Nonlinear adaptive fault-tolerant quadrotor altitude and attitude tracking with multiple actuator faults. *IEEE Transactions on Control Systems Technology*, 26(2):701–707, 2018.

[166] Zhiwei Hou, Peng Lu, and Zhangjie Tu. Nonsingular terminal sliding mode control for a quadrotor UAV with a total rotor failure. *Aerospace Science and Technology*, 98:105716, 2020.

[167] Daniel Boley. Computing the Kalman decomposition: An optimal method. *IEEE Transactions on Automatic Control*, 29(1):51–53, 1984.

[168] Tao Pang, Kemao Peng, Feng Lin, and Ben M Chen. Towards long-endurance flight: Design and implementation of a variable-pitch gasoline-engine quadrotor. In *Proceedings of 2016 12th IEEE International Conference on Control and Automation (ICCA)*, pages 767–772. IEEE, 2016.

[169] Mihály Petreczky, Roland Tóth, and Guillaume Mercère. Realization theory for LPV state-space representations with affine dependence. *IEEE Transactions on Automatic Control*, 62(9):4667–4674, 2016.

[170] Zhikun Wang, Roderich Groß, and Shiyu Zhao. Controllability analysis and controller design for variable-pitch propeller quadcopters with one propeller failure. *Advanced Control for Applications: Engineering and Industrial Systems*, page e29, 2020. (Invited special issue).

[171] A. Baldini, R. Felicetti, A. Freddi, S. Longhi and A. Monteriu. Actuator fault tolerant control of variable pitch quadrotor vehicles. In *Proceedings of 21st IFAC World Congress*. IFAC, 2020.

[172] F3CN. Helicopter Manoeuvre Descriptions, 2020. URL https://www.f3cn.org/index.php/system/files/archive/Annex%205F.1%20F3N%20Manoeuvre%20Descriptions.pdf.

[173] Teodor Tomić, Moritz Maier, and Sami Haddadin. Learning quadrotor maneuvers from optimal control and generalizing in real-time. In *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1747–1754. IEEE, 2014.

[174] Guofan Wu and Koushil Sreenath. Safety-critical control of a planar quadrotor. In *Proceedings of 2016 American Control Conference (ACC)*, pages 2252–2258. IEEE, 2016.

[175] Xingyi Zhang, Ye Tian, Ran Cheng, and Yaochu Jin. A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 22(1):97–112, 2016.

[176] Jon Hull. Master the Tic Toc, 2021. URL http://www.helipilotonline.com/master-the-tic-toc/.

[177] Hassan K Khalil. *Noninear systems*. New York: Macmillan, 1992.

[178] SS Antman JE Marsden, L Sirovich S Wiggins, L Glass, RV Kohn, and SS Sastry. *Interdisciplinary Applied Mathematics*, volume 3. Springer, 1993.

[179] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proceedings of 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[180] Haobin Shi, Lin Shi, Meng Xu, and Kao-Shing Hwang. End-to-end navigation strategy with deep reinforcement learning for mobile robots. *IEEE Transactions on Industrial Informatics*, 16(4):2393–2402, 2019.

[181] Yang Li, Shijie Guo, Lishuang Zhu, Toshiharu Mukai, and Zhongxue Gan. A recurrent reinforcement learning approach applicable to highly uncertain environments. *International Journal of Advanced Robotic Systems*, 17(2): 1729881420916258, 2020.

[182] Tom Le Paine, Caglar Gulcehre, Bobak Shahriari, Misha Denil, Matt Hoffman, Hubert Soyer, Richard Tanburn, Steven Kapturowski, Neil Rabinowitz, Duncan Williams, et al. Making efficient use of demonstrations to solve hard exploration problems. *arXiv preprint arXiv:1909.01387*, 2019.

[183] Davide Bicego, Jacopo Mazzetto, Ruggero Carli, Marcello Farina, and Antonio Franchi. Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs. *Journal of Intelligent & Robotic Systems*, 100(3):1213–1247, 2020.

[184] RCbenchmark. RCbenchmark Series 1850, 2021. URL http://www.tytororobotics.com/.

Appendix A

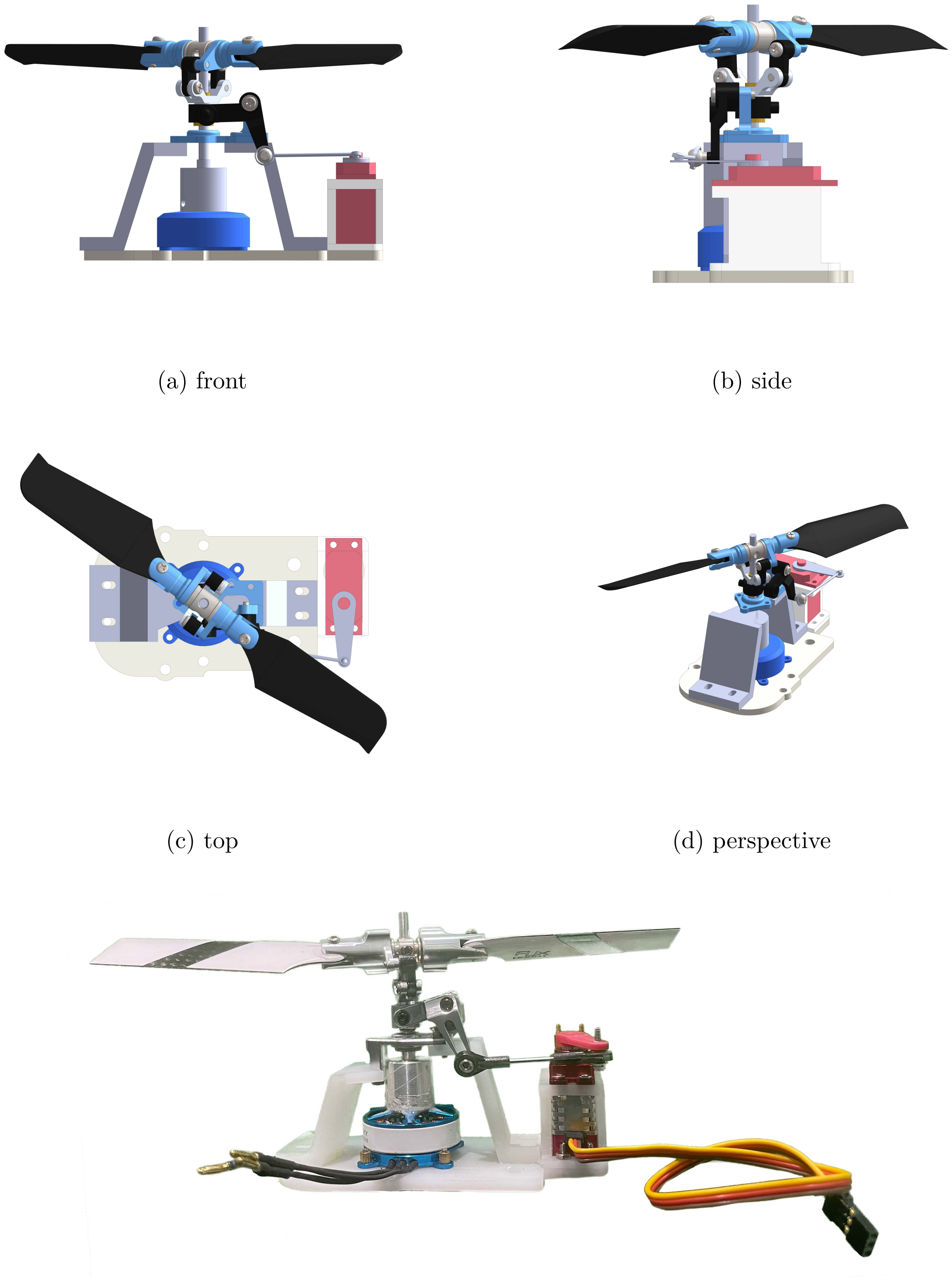
Variable Pitch Propeller Test Stand Details

A.1 Variable Pitch Propeller Actuator

A standard VPP actuator contains two control inputs, a propeller pitch angle and a propeller rotation speed. These two control inputs are determined by a servo and a brushless motor, respectively. In this project, we choose to use a GARTT 450 helicopter tail rotor as the designed structure basis. We only retained the propeller connection and propeller pitch adjustment mechanism of this helicopter tail rotor. The original belt driven connection is replaced by a motor shaft. This design aims to directly power the VPP propeller through a brushless motor. In order to balance the VPP safety and performance, a pair of 65 mm NACA 0012 propellers are used to replace the original propellers. We use the standard symmetrical rectangular NACA0012 series propellers to maintain consistency with the simulation experiment. The servo motor is designed to be placed parallel to the motor shaft and close to the motor. Such a mechanical design aims to reduce space occupation and balance mass distribution.

Other connection designs are designed based on the previous structure basis to achieve a firm structure and convenient installation. Most of these connections are built by 3D printing where the overall structure design is shown in Fig. A.1.

Generally, the coaxiality error between the motor and the shaft will cause vibration. Therefore, we use a motor shaft produced by high-precision CNC machining to reduce this coaxiality error.


(a) front

(b) side

(c) top

(d) perspective

(e) construction

Figure A.1. VPP actuator design. (a)-(d) show the mechanical design from different perspectives and (e) shows the actual construction.

Due to the certain deviation between the actual model purchased and the 3D printed accessories, the design sketch is somewhat different from the actual one and the actual construction is shown in Fig. A.1 (e). However, this does not affect the overall VPP actuator performance.

A.2 Test Stand

The actuator test stand we used is a standard general mechanical structure designed by RCbenchmark [1] [184], shown in Fig. A.2 (a). The test stand is mainly constructed by three force sensors, one vertical and two horizontal aligned, as shown in Fig. A.2 (b). In addition, an RPM probe is used to measure the rotation speed of the motor and a Futaba R6208SB receiver is used for servo control. A connection board is designed to connect VPP actuators and test stand as well.

This actuator test stand also provides testing software to control the dynamic VPP actuator system through test scripts. It can also record and save the sensor feedbacks autonomously. This makes the platform an excellent candidate for this work.

A.3 Implementation Design

This section extends Chapter 3 and provides how the thrust and torque testing experiments are implemented. Since the software only provides the function of controlling the motor, the servo is designed to be controlled through a wireless remote controller.

The experiment begins with zero thrust pitch angle calibration to reduce assembly errors. We test the thrust produced by the VPP, which runs at 60% full speed, and the pitch angle sweep from -5 to 5 degrees. The angle that generates the minimum thrust is considered as the zero thrust pitch angle.

For practical reasons, we use an ESC to control the VPP rotation speed. Instead of directly setting the target motor rotation speed, we used an evenly distributed 50 Hz PWM signal as the reference motor control input. However, due to the dead zone of the ESC, our VPP actuator can not run with low rotation speed. In our case, the VPP actuator starts to rotate when the PWM signal provided to its ESC is higher than 1455 ms.

[1]For more details visit: http://www.tytororobotics.com

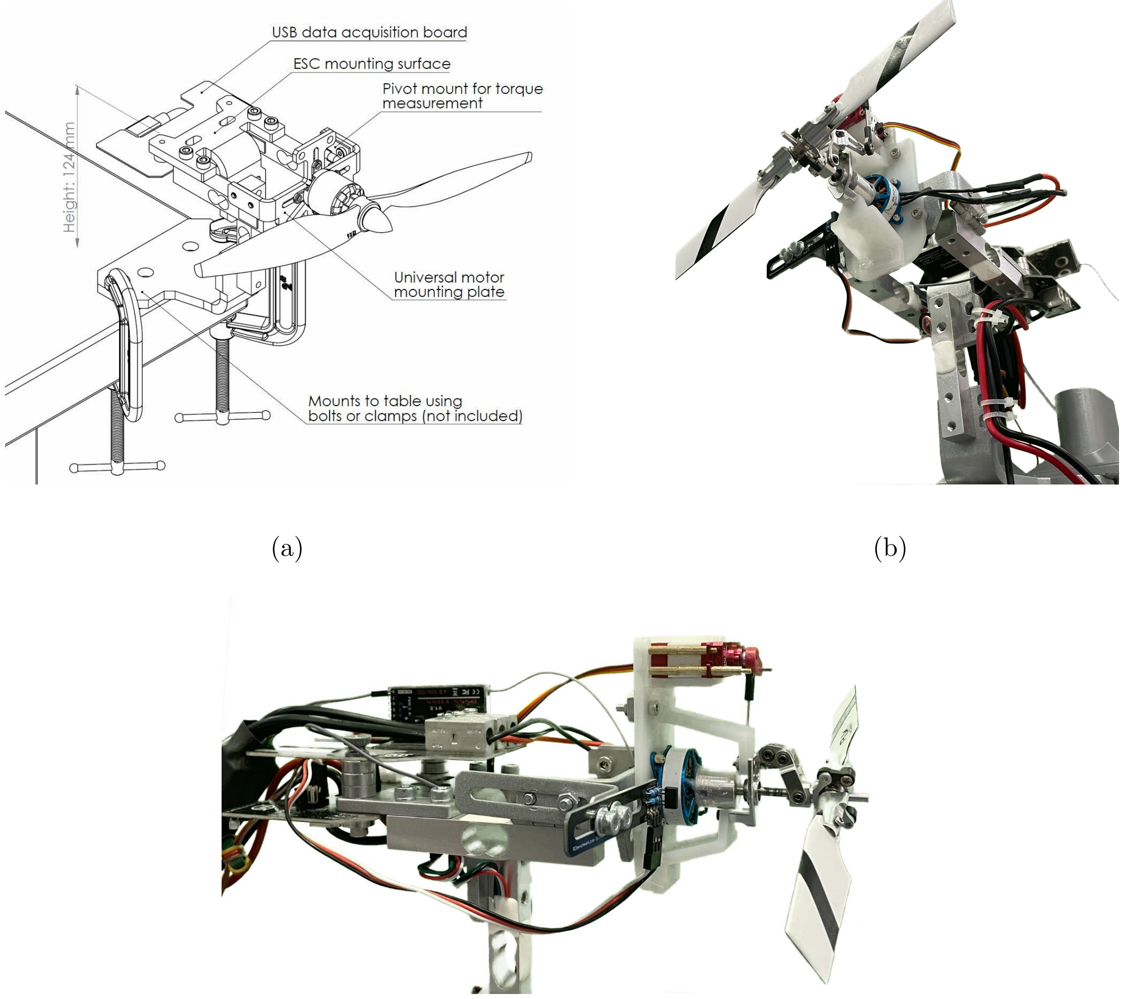

USB data acquisition board
ESC mounting surface
Pivot mount for torque measurement
Height: 124 mm
Universal motor mounting plate
Mounts to table using bolts or clamps (not included)
(a)
(b)
(c)

Figure A.2. The actuator test sand, where (a) is the sketch diagram reprinted from [184] and (b)-(c) are the actual constructions.

A testing script is designed to test the thrust and torque generated by the VPP under different control combinations (include motor speeds and propeller pitch angles). The script uses step function signals to control the rotation speed of the motor with the same propeller pitch angle to test the thrust and torque. In order to test different control combinations, we need to run the script multiple times. We call each full run of the script a trial.

In each trial, we set 21 data collection points from 1450 ms (about to rotate) to 2000 ms (maximum speed) according to PWM frequency. After reaching the target signal, the VPP actuator sets a settling time of 5 seconds in order to avoid dynamic interference. Then the platform collects sensor data. To reduce random noise, each sampling data we recorded is the average value of the sensor feedbacks within 10 seconds.

In addition, the propeller angle is fixed during each trial, and there will be slight changes between the two trials. The VPP pitch angle ranges from -0.35 rad to 0.35 rad, with an increase of 0.0436 rad for each trial. Instrument calibration is performed at the beginning of each trial.

The testing code is written in JavaScript and the pseudocode is presented in Algorithm 2

Algorithm 2: VPP Actuator Step Testing Program

```
Input: minimum input Value $v_{min} = 1450$, maximum input Value
       $v_{max} = 2000$, step numbers $Q = 21$, settling time $t_s = 5$, average
       sampling $t_a = 10$
begin
    Creat a sequence $N$ of $Q$ numbers that are equally distributed from $v_{min}$
      to $v_{max}$
    for $v \leftarrow N$ do
        set motor PWM to v
        running for $t_s$ seconds
        collect the sensor data for $t_a$ seconds
        calculate the average of the feedback data during $t_a$ seconds
        if Vibration too large then
            exit for
    store the data into a record file
```

Appendix B

Reinforcement Learning Methods Comparison

In addition to the DDPG algorithm we presented in Chapter 6, we also compared it with three other RL algorithms. This section outlines the comparison of the principles and performances of these three methods. To compare the training performance, all these algorithms run in the same pendulum environment for 6 trials, each trial with a different random seed. Random seeds will determine the initial network parameter settings and have a great impact on the learning performance. In order to test the algorithm with the same initializations, a variable-controlling approach is used. All these algorithms use the same random seed sequence. In addition, in order to speed up the training process, when the network reaches the reward threshold (-740) for 10 consecutive episodes, the program will stop training in advance.

All of these algorithms are tested in the Matlab R2021a version, on the same computer equipped with Intel Core i9-9900K, 64 GB RAM, and without using GPU.

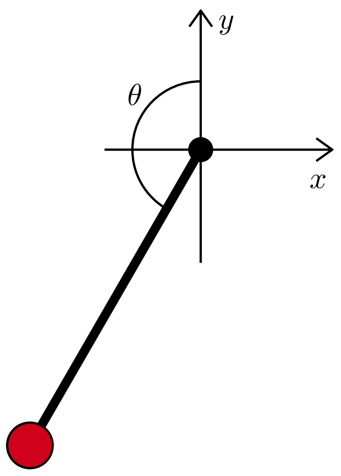


Figure B.1. RL training environment.

B.1 Environment Setting

The training environment is shown in Fig. B.1, where the pendulum starts from downward. The aim of this environment is to control and stabilize the pendulum to the upward direction align with y axis. The angle between the stick and y axis is given as $\theta \in [-\pi, \pi)$, where θ is positive when the pendulum is in the area of the negative x-axis. Assume the mass of the stick for connection is ignored, the mass of the red ball is 1 kg, and the distance between the ball and the centre of rotation is 1 m. The torque τ is acted on the end and is limited from -2 to 2 N·m. To reduce nonlinear parameters, the observations of this environment is set as $\left[\cos\theta, \sin\theta, \dot{\theta}\right]$. Therefore, the reward function is set as

$$R = -(\theta^2 + 0.1\dot{\theta}^2 + 0.001\tau^2).$$

B.2 Deep Deterministic Policy Gradient

In addition to the principles presented in Chapter 6, the detailed experiment results are given in this section.

B.2.1 Evaluation

The results of 6 trials are shown in Fig. B.2. Training results show that DDPG algorithm can pre-stop the training section within 350 episodes. DDPG shows great training performance compare with the other three algorithms both in training time (average 4.5 hours) and final results (≤ -740).

In addition, DDPG can provide the same action when the states are the same. This is more similar to our expected controller output.

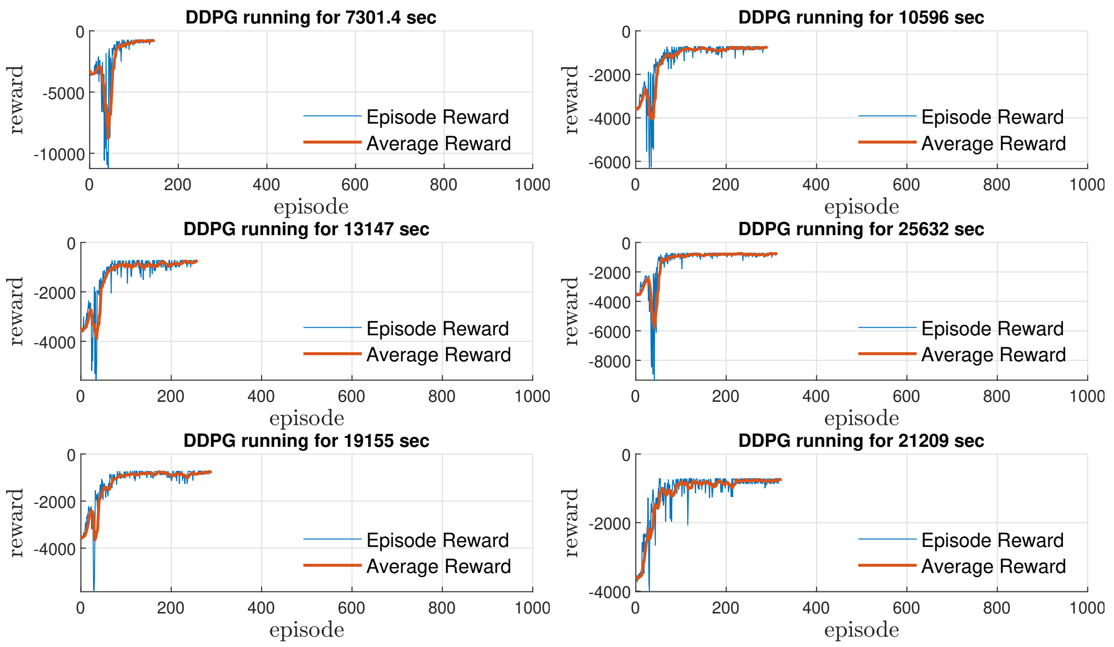


Figure B.2. DDPG training results, where each subplot is a trial result. The largest learnt reward are -729, -725, -739, -733 and -728, corresponding to the subplots from left to right and up to down.

B.3 Twin Delayed Deep Deterministic Policy Gradient

B.3.1 Principles

Twin delayed deep deterministic policy gradient (TD3) is a successor of DDPG algorithm. It aims to solve the problem that DDPG will overestimate the Q-values [142]. It uses three tricks to achieve such a target:

1. Double Q NN. TD3 uses two Q-networks to criticize the agent performance and choose the lower one to compute the target network. This method can update the policy network to prevent overestimation. Therefore, the overall algorithm has six networks in total.

2. Delayed Policy Update. Once a poor policy network is overestimated, it will lead to agent diverges. A less frequently updated policy network can have more stable training.

3. Target Policy Smoothing. TD3 uses a regularization strategy for the value function update. It adds clip noise to the action in order to achieve the target that similar actions should have similar estimate values. This makes the estimation of the target network becomes robust and accurate.

The pseudocode of TD3 is shown in Algorithm 3.

B.3.2 Evaluation

The training results are shown in Fig. B.3. Except for the unfinished learning of one episode, the average learning time of TD3 is 8.9 hours. It is notable that, compared with other algorithms, TD3 is more susceptible to the influence of randomized seeds. Results show that TD3 training performance is only competitive in well-initialized trials. However, when the initialization is not good, there will be a huge gap in the performance of TD3 training.

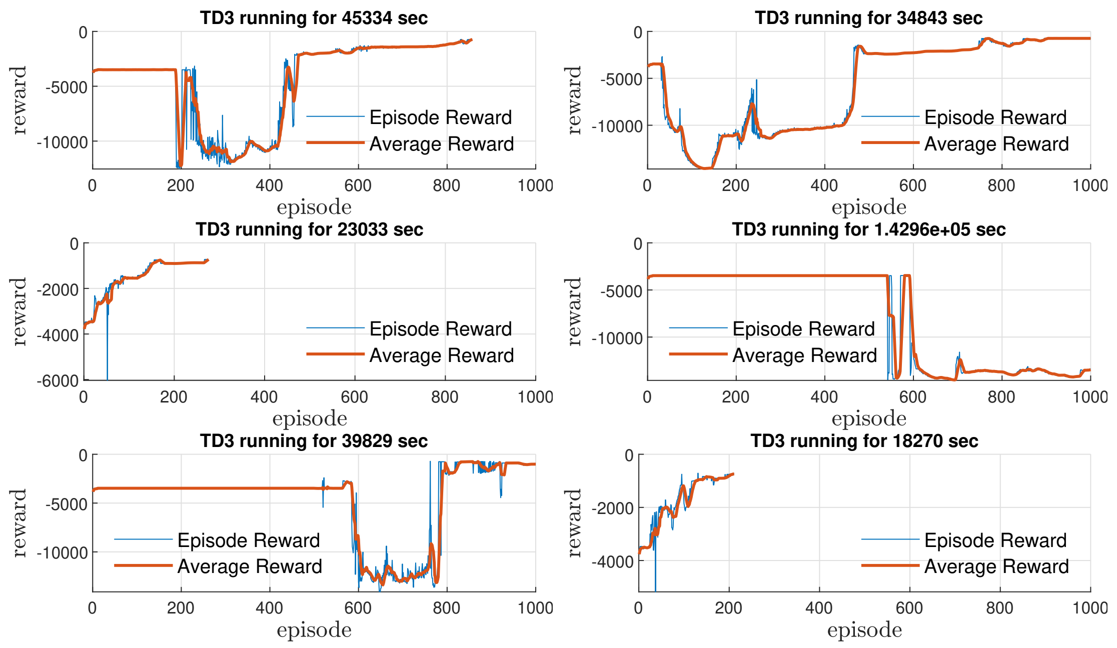


Figure B.3. TD3 training results, where each subplot is an average trial result. The largest learnt reward are -734, -740, -727, -3469, -745 and -739, corresponding to the subplots from left to right and up to down.

Algorithm 3: Twin Delayed Deep Deterministic Policy Gradient

```
Input: episodes $K$, steps per episode $T$, batch size $M$
Init: actor policy NN $\mu(a|s,\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$, critic value NNs
      $Q_1(s,a,\boldsymbol{w}_1)$ with parameters $\boldsymbol{w}_1$ and $Q_2(s,a,\boldsymbol{w}_2)$ with parameters $\boldsymbol{w}_2$,
      replay experience $R$
Init: target policy NN parameters $\mu'(a|s,\boldsymbol{\theta}') \leftarrow \mu(a|s,\boldsymbol{\theta})$, target value NN
      parameters $Q'_1(s,a,\boldsymbol{w}'_1) \leftarrow Q_1(s,a,\boldsymbol{w}_1)$ and
      $Q'_2(s,a,\boldsymbol{w}'_2) \leftarrow Q_2(s,a,\boldsymbol{w}_2)$
begin
    for $k \in K$ do
        Reset the environment and the noise model $N$
        for $t \in T$ do
            Take action $a_t = \mu(s,\boldsymbol{\theta}) + \epsilon, \epsilon \in N$ by observe state $s_t$
            Observe next time step states $s_{t+1}$, done signal $d_t$ and reward $r_t$
            Store experience set $< s_t, a_t, r_t, s_{t+1}, d_t >$ in $R$
            if it is time to update then
                Sample $M$ random experience sets $< s_i, a_i, r_i, s_{i+1}, d_i >$ from $R$
                Compute targets
                    if $d_t == done$ then $y_i = r_i$
                    else $y_i = r_i + \gamma \min_{j=1,2} Q'_j(s_{i+1}, \text{clip}(\mu'(s_{i+1}, \boldsymbol{\theta}' + \epsilon)), \boldsymbol{w}'_j)$
                Update the critic value NN paramters $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ to minimize
                  the loss (gradient descent) by
                    $L_j = \frac{1}{M}\sum_{i=1}^{M}(y_i - Q_j(s_i, a_i, \boldsymbol{w}_j))^2 \quad \text{for } j = 1,2$
                if policy update delay then
                    Update the actor policy NN paramters $\boldsymbol{\theta}$ to maxmize the
                      expected reward (gradient ascent) by
                        $\nabla_{\boldsymbol{\theta}} \approx \frac{1}{M}\sum_{i=1}^{M}\Big(\nabla_a \min_{j=1,2} Q_j(s, \mu(s,\boldsymbol{\theta}), \boldsymbol{w}_j)\nabla_{\boldsymbol{\theta}}\mu(s_i,\boldsymbol{\theta})\Big)$
                    Soft target update:
                        $\boldsymbol{\theta}' \leftarrow \tau\boldsymbol{\theta}' + (1-\tau)\boldsymbol{\theta}$
                        $\boldsymbol{w}'_1 \leftarrow \tau\boldsymbol{w}'_1 + (1-\tau)\boldsymbol{w}_1$
                        $\boldsymbol{w}'_2 \leftarrow \tau\boldsymbol{w}'_2 + (1-\tau)\boldsymbol{w}_2$
            if $d_t == done$ then quit episode
```

B.4 Proximal Policy Optimization

B.4.1 Principles

Proximal policy optimization (PPO) is an on-policy algorithm that the agent takes action according to policy probability and update the policy by taking the great possible improvement step without stepping so far. There are two variants of PPO, namely PPO-penalty and PPO-clip, where the difference is the policy update rule. Because PPO-Clip uses a clipping function to keep the policy improvement step, it is simple to implement. Here, we use the PPO-clip according to [144].

In order to speed up the training process and keep the new policy not quite different from the old policy, PPO only uses the policy from the previous learning epoch as the old-policy to collect experience sets. Therefore, PPO is still considered as an on-policy algorithm. In order to update the policy by using the experience sets collected from the old policy, PPO first defines the probability ratio by $d = \frac{p(a|s,\boldsymbol{\theta})}{p(a|s,\boldsymbol{\theta}_{old})}$, where $\boldsymbol{\theta}_{old}$ is the old policy and $\boldsymbol{\theta}$ is the new policy. Then, according to the importance sampling algorithm, PPO can update the new policy via dA_n, where A_n is the advantage estimation of the old policy. Moreover, a clip function given by $\text{clip}(d, 1+\epsilon, 1-\epsilon)$ is built to limit the update step value d between $1+\epsilon$ and $1-\epsilon$. If the probability ratio d is far from 1, the gradient becomes 0 and the policy will not be updated. The clip factor ϵ is normally defined by 0.2.

The pseudocode of PPO is shown in Algorithm 4.

```
Algorithm 4: Proximal Policy Optimization
  Input: episodes $K$, steps per episode $T$, batch size $M$, learning epoch $E$
  Init: actor policy NN $p(a|s,\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$, critic value NNs
        $Q(s,a,\boldsymbol{w})$ with parameters $\boldsymbol{w}$
  begin
      for $k \in K$ do
          Running the current agent policy NN $p(a|s,\boldsymbol{\theta}_k)$ and collect sequence
            sets of $N$ experience started from $t$ as
            $< s_t, a_t, r_{t+1}, s_{t+1}, \dots, r_{t+N}, s_{t+N} >$
          Compute the generalized advantage estimation $A_n$ of N step
            $n \in (1, \dots, N)$ by generalized advantage estimator, based on the
            current value policy $Q(s,a,\boldsymbol{w}_k)$
          $\boldsymbol{\theta} \rightarrow \boldsymbol{\theta}_{old}$
          for $e \in E$ do
              Sample M experience sets
              Update Critic Network by $L = \frac{1}{M}\sum_{i=1}^{M}(r_i + \gamma Q_{i+1} - Q_i)^2$
              Calculate the clip action difference by $d = \frac{p(a|s,\boldsymbol{\theta})}{p(a|s,\boldsymbol{\theta}_{old})}$
              Update Actor Network by
                  $\nabla_{\boldsymbol{\theta}} = -\frac{1}{M}\sum_{i=1}^{M}\min(dA_i, \text{clip}(d, 1+\epsilon, 1-\epsilon)A_i)$
```

B.4.2 Evaluation

The training results are shown in Fig. B.4, where all trials have not finished the training target within 1000 episodes. The average training time is 2.5 hours and the average best reward is -1225. The results indicate that, within our training environment, PPO algorithm has no outstanding performance compared with other algorithms.

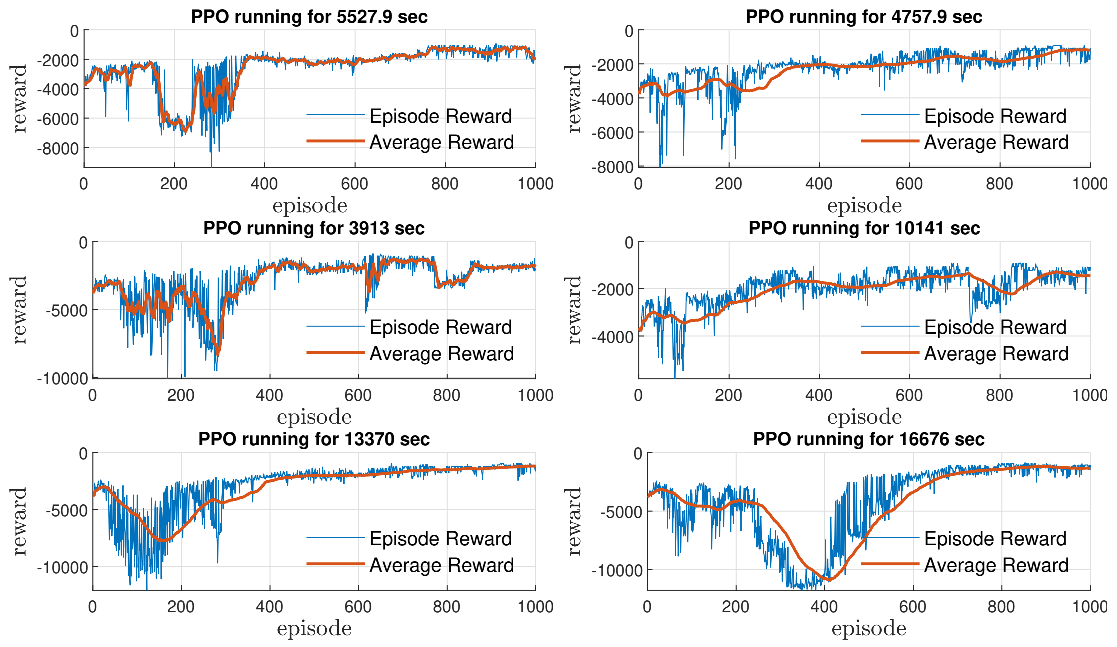


Figure B.4. PPO training results, where each subplot is an average trial result. The largest learnt reward are -1152, -1160, -1316, -1288, -1176 and -1184, corresponding to the subplots from left to right and up to down.

In addition, we find that although reward value did not meet expectations, the trained PPO network may stabilize the pendulum. This may occur because the PPO is a scholastic RL algorithm and it use a Gaussian distribution to select the action. Because of this randomness, the trained network can not always take the same action even under the same states. That is the main reason why the PPO algorithm can not reach the reward threshold and pre-stop the training.

B.5 Soft Actor-Critic

B.5.1 Principles

Soft Actor-Critic (SAC) is an off-policy algorithm that optimize the stochastic policy in a TD3-style approaches. It uses double Q NN and target NN structure from TD3 and Gaussian probability distribution action policy similar to PPO.

Moreover, SAC uses entropy regularization to balance exploration and exploitation for the policy training. Entropy is a quantity that represents the randomness of the distribution of the action policy. Increasing entropy leads to closer possibilities for all actions and encourages actors to explore more. Entropy of action a who obeys the distribution of $p(a)$ is given by $H(p) = \alpha E_{a \sim p}[-\log p(a)]$, where α is the entropy weight.

Moreover, SAC can prevent the policy from prematurely converging to a local optimum.

The pseudocode of SAC is shown in Algorithm 5.

B.5.2 Evaluation

The training results are shown in Fig. B.4, where the average finish time is 12.8 h. Compare with PPO, which has same stochastic policy, SAC shows better training performance. Compare with DDPG, which has similar training performance, SAC requires twice the training time.

B.6 Summary

Comparing the results from Fig. B.2 to B.5, it can be found that PPO and TD3 can not achieve good training results as we expected in the simulated pendulum environment. SAC has almost the same good training performance compare with DDPG, whether it is from the final convergence speed (episodes) or the final performance. However, SAC requires more than twice the training time of DDPG.

We also notice that the training performance of these algorithms differs between our research and the literature. This difference may be caused by the coding environment, hyperparameters, or other advanced tricks.

In addition, to further analyse the controller performance, we test four RL algorithm NNs in the same environment. The chosen NN of each RL algorithm is

Algorithm 5: Soft Actor-Critic

```
Input: episodes $K$, steps per episode $T$, batch size $M$, reply buffer $R$, target
       entropy weight $H_t$
Init: actor policy NN $p(a|s, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$, critic value NNs
      $Q_1(s, a, \boldsymbol{w}_1)$ with parameters $\boldsymbol{w}_1$ and $Q_2(s, a, \boldsymbol{w}_2)$ with parameters $\boldsymbol{w}_2$
Init: target policy NN parameters $p'(a|s, \boldsymbol{\theta}') \leftarrow p(a|s, \boldsymbol{\theta})$, target value NN
      parameters $Q'_1(s, a, \boldsymbol{w}'_1) \leftarrow Q_1(s, a, \boldsymbol{w}_1)$ and
      $Q'_2(s, a, \boldsymbol{w}'_2) \leftarrow Q_2(s, a, \boldsymbol{w}_2)$
begin
    for $k \in K$ do
        Take action $a_t = p(s_t)$ by observe state $s_t$
        Observe next time step states $s_{t+1}$, done signal $d_t$ and reward $r_t$
        Store experience set $< s_t, a_t, r_t, s_{t+1}, d_t >$ in $R$
        if *it is time to update* then
            Sample $M$ random experience sets $< s_i, a_i, r_i, s_{i+1}, d_i >$ from $R$
            Compute targets
                if $d_t == done$ then $y_i = r_i$
                else $y_i = r_i + \gamma(\min_{j=1,2} Q'_j(s_{i+1}, a_{i+1}, \boldsymbol{w}'_j) - H(p(s_{i+1}, \boldsymbol{\theta})))$
            Update the critic value NN paramters $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ to minimize the
              loss (gradient descent) by
                $L_j = \frac{1}{M}\sum_{i=1}^{M}(y_i - Q_j(s_i, a_i, \boldsymbol{w}_j))^2 \quad \text{for } j = 1, 2$
                if *policy update delay* then
                    Update the actor policy NN paramters $\boldsymbol{\theta}$ to maxmize the
                      expected reward (gradient ascent) by
                      $\nabla_{\boldsymbol{\theta}} = \frac{1}{M}\sum_{i=1}^{M}\left(\min_{j=1,2} Q'_j(s_i, a_i, \boldsymbol{w}'_j) - H(p(s_i, \boldsymbol{\theta})))\right)$
                    Update the entropy weight :
                      $L_\alpha = \frac{1}{M}\sum_{i=1}^{M}(\alpha H_t - H(p(s_i, \boldsymbol{\theta}))^2)$
                    Soft target update:
                      $\boldsymbol{\theta}' \leftarrow \tau\boldsymbol{\theta}' + (1-\tau)\boldsymbol{\theta}$
                      $\boldsymbol{w}'_1 \leftarrow \tau\boldsymbol{w}'_1 + (1-\tau)\boldsymbol{w}_1$
                      $\boldsymbol{w}'_2 \leftarrow \tau\boldsymbol{w}'_2 + (1-\tau)\boldsymbol{w}_2$
        if $d_t == done$ then quit episode
```

the one who has the highest reward value within all 6 trials. Results are shown in Fig. B.6.

Results indicate that all these four RL methods can successfully achieve the upward control goal in the pendulum balancing environment. Figures B.6(a) and B.6(b) clearly show that DDPG and TD3 NNs can take deterministic action for the same state and generate torque value smoothly. Since the random algorithms SAC

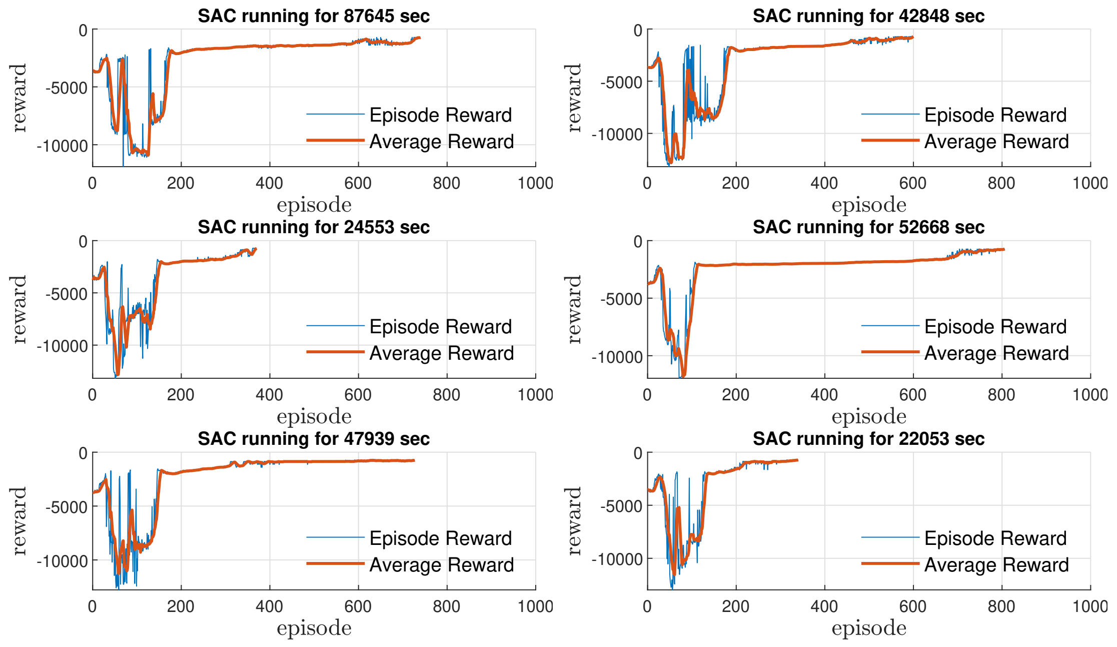
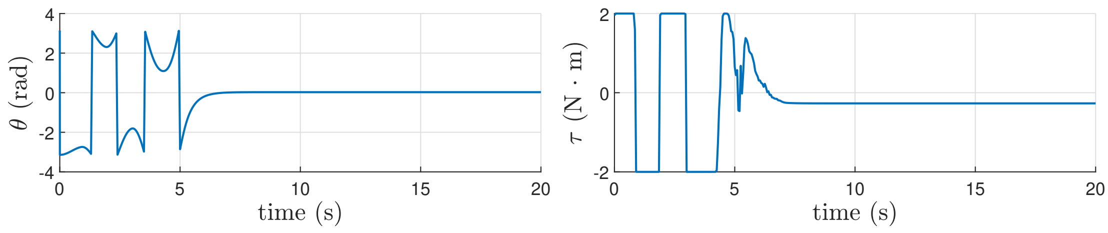


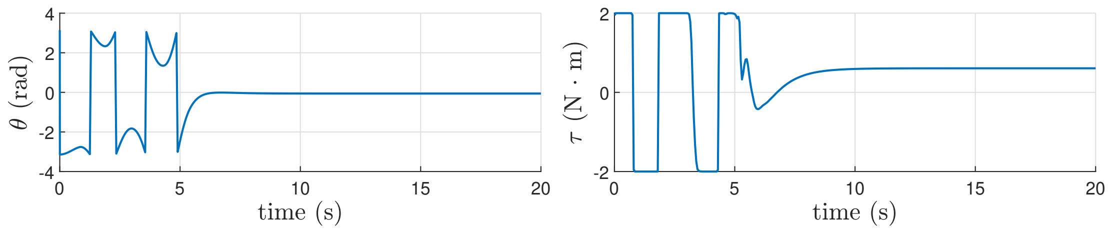
Figure B.5. SAC training results, where each subplot is an average trial result. The largest learnt reward are -735, -728, -730, -735, -737 and -729, corresponding to the subplots from left to right and up to down.

and PPO use probability distribution to select actions, their action outputs are an fluctuating curves, shown in Figs B.6(c) and B.6(d).

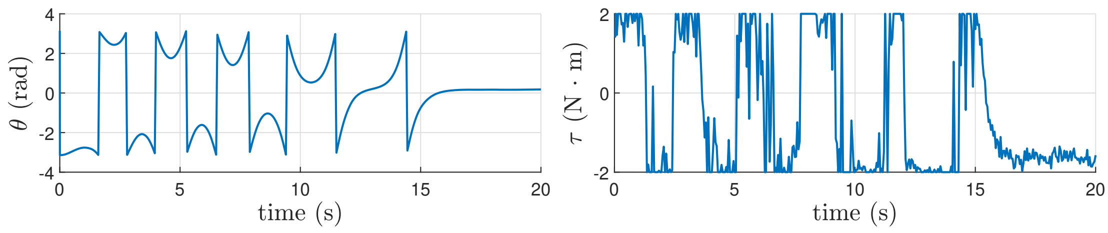
In this study, we aim to control a VPP quadcopter to realize tic-toc manoeuvrer through RL algorithms. Therefore, RL policies that can generate smooth action output are closer to the real situation and are worthy of recommendation. In contrast, stochastic reinforcement learning algorithms cannot meet our expectations. Overall, we choose DDPG as our baseline training algorithm for tic-toc manoeuvrer training.



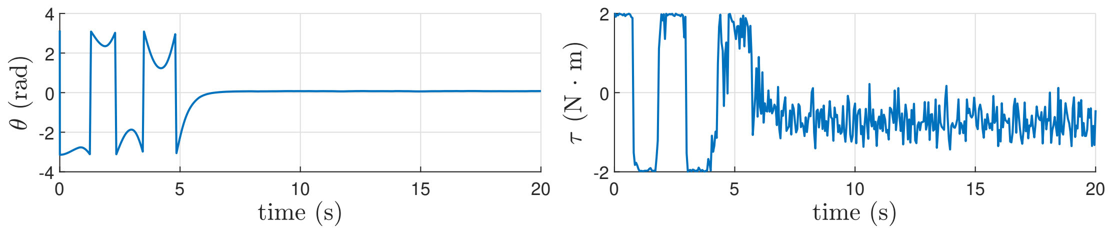
(a)



(b)



(c)



(d)

Figure B.6. State feedback of four RL algorithms in a pendulum simulation environment. From (a) to (d) are simulation results of DDPG, TD3, PPO and SAC correspondingly. The left sub-figures are the pendulum angles and the right ones are the torques (actions).