Data-Driven Modeling and High-Performance Control of Multirotor Unmanned Aerial Vehicles in Challenging Environments

By

PENG WEI

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Mechanical and Aerospace Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Zhaodan Kong, Chair

Stavros G. Vougioukas

Iman Soltani

Committee in Charge

2023

CONTENTS

List of Figures v

List of Tables x

Abstract xi

Acknowledgments xiii

1 Introduction **1**

1.1 Motivation 1

1.2 Background 4

1.2.1 Multirotor UAV Modeling 4

1.2.2 Multirotor UAV Advanced Control 6

1.3 Contributions 8

1.4 Dissertation Outline 10

2 Mitigating Multirotor UAV Ground Effect using Adaptive Control **11**

2.1 Introduction 12

2.2 Related Work 13

2.3 Hardware Description 16

2.3.1 Test Quadcopter 16

2.3.2 Ground Effect Test Rig 16

2.4 Ground Effect Modeling 18

2.4.1 Experiment Design 19

2.4.2 Modeling Results 20

2.5 Controller Design 27

2.5.1 Equations of Motion 27

2.5.2 Attitude Controller 28

2.5.3 Position Controller 30

2.6 Fight Experiment Setup 33

2.7 Flight Experiment Results 34

2.7.1 Discussion 39
2.8 Summary 45

3 System Identification of Wind Effects on Multirotor UAV Dynamics 46
3.1 Introduction 46
3.2 Related Work 47
3.3 Flight Test Data Collection 49
3.3.1 Hardware Description 49
3.3.2 Wind Distribution Hypothesis 50
3.3.3 Flight Test Methodology 52
3.4 System Identification Methodology 56
3.4.1 Model Structure 57
3.4.2 Model Parameter Identification 59
3.5 System Identification Results 60
3.5.1 Calm Wind Results 60
3.5.2 Light and Strong Wind Results 62
3.5.3 Time Domain Verification 66
3.6 Eigenvalue Analysis 70
3.7 Discussion 71
3.8 Summary 72

4 Vision-based Control of UAV in Simulated Riverine Environments Using Imitation Learning 73
4.1 Introduction 73
4.2 Related Work 76
4.3 Methodology 79
4.3.1 Simulated Riverine Environments 79
4.3.2 Controller Design 79
4.3.3 Learn from Demonstrations 85
4.3.4 Experiment Design 86

4.4 Experimental Results 89
4.4.1 Training Results 89
4.4.2 Testing Results 90
4.5 Discussion 99
4.6 Summary 103

5 Validation of Vision and Learning-based UAV Control in Real Orchard Environments 104
5.1 Introduction 104
5.2 Related Work 106
5.3 Methodology 109
5.3.1 Intervention-based Learning 109
5.3.2 Network Architecture 110
5.3.3 Data Collection in Real World 112
5.4 Experimental Results 114
5.4.1 Experimental Platform 114
5.4.2 Qualitative Results 117
5.4.3 Quantitative Results 118
5.5 Discussion 123
5.6 Summary 126

6 Conclusions and Future Work 128
6.1 Conclusions 128
6.2 Future Work 130

A MRAC for MIMO System 148

B Equations of Motion for Multirotor UAV 151

List of Figures

1.1 Examples of UAV operating in challenging environments [1, 2]. 1

1.2 Example of multirotor UAVs taking off and landing in ground effect regions. . 2

1.3 Examples of multirotor UAV operating in windy conditions. 3

1.4 Examples of multirotor UAV operating in GPS-denied natural environments. . . 3

2.1 The modified Crazyflie 2.0 quadcopter with reflective markers and a laser-cut frame. 17

2.2 The test rig used to acquire the thrust generated by the quadcopter. 18

2.3 A sketch of the experiment setup used to emulate and measure the ground effect. 19

2.4 Fitting result with 2 rotors active. 20

2.5 Fitting result with 4 rotors active. 21

2.6 Comparison of three ground effect models. 22

2.7 The comparison of in-flight test result and on-ground test result, where $\times$ represents the flight test data and O represents the ground test data. 23

2.8 Increase the rotor distance by 10mm (a) Experimental result with 2 rotors active. (b) Experimental result with 4 rotors active. 25

2.9 Increase the rotor distance by 50mm (a) Experimental result with 2 rotors active. (b) Experimental result with 4 rotors active. 26

2.10 The control architecture of the system. 32

2.11 System diagram of the flight experiment. 33

2.12 The altitude response when the vehicle is completely outside the ground effect region. 34

2.13 (a) The altitude response and (b) absolute error when the vehicle is under the ground effect. 35

2.14 (a) The altitude response and (b) corresponding mean absolute error at different heights. 37

2.15 Average takeoff and landing trajectories with different control algorithms for $Z/R \approx 4$. 39

2.16 Average takeoff and landing trajectories with different control algorithms for $Z/R \approx 3$. 40
2.17 Average takeoff and landing trajectories with different control algorithms for $Z/R \approx 2$. 41

3.1 The octocopter platform used for the flight experiment. 49
3.2 Outdoor flight experiment setup. 51
3.3 Schematic illustration of the two wind sensor relative positions of which the sensors are positioned a) along the wind direction and b) perpendicular to the wind direction. 52
3.4 Example readings of the two wind sensors a) along the wind direction and b) perpendicular to the wind direction. 53
3.5 Schematic diagram of the bare-airframe model identification framework. 54
3.6 Example longitudinal axis flight data with (a) frequency sweep and (b) doublet signals under the calm wind condition. 55
3.7 Transfer function responses along the longitudinal axis under the calm wind condition. 61
3.8 Transfer function responses along the yaw and heave axes under the calm wind condition. 61
3.9 Transfer function responses in the longitudinal axis under the light wind condition. 64
3.10 Transfer function responses in the longitudinal axis under the strong wind condition. 65
3.11 Residuals of a_x and a_z under the light wind condition. 65
3.12 Residuals of a_x and a_z under the strong wind condition. 66
3.13 Time-domain validation under the calm wind condition. 67
3.14 Time-domain validation under the light wind condition with and without the wind correction. 67
3.15 Time-domain validation under the strong wind condition with and without the wind correction. 68

3.16 Plots of eigenvalues along the longitudinal and lateral axes under different wind conditions. 70

4.1 A drone flying in GPS-denied riverine environments requires a navigation policy. 74

4.2 A subset of the environments used for training with different difficulty levels: (a) easy; (b) medium; (c) hard. 79

4.3 System diagram of the linear regression-based controller. 80

4.4 The architecture of end-to-end neural network. 82

4.5 System diagram of the end-to-end neural network controller. 83

4.6 The architecture of variational autoencoder network. 84

4.7 The reconstructing performance of the VAE, which (a) are the sample images and (b) are the reconstructed images trained after 150 epochs. 84

4.8 System diagram of the VAE-based controller. 85

4.9 A conceptual illustration of the intervention-based learning method. 85

4.10 Example trajectories of the UAV in simulation environments (a) Map 1 (easy); (b) Map 3 (medium); (c) Map 5 (hard). 89

4.11 Percentage of interventions from human pilots in (a) easy maps; (b) medium maps; and (c) hard maps during the training. 91

4.12 Maximum distances traveled before a failure by three different controllers. . . . 93

4.13 Average distances traveled per failure by three different controllers. 93

4.14 Processing time of three different controller types. 94

4.15 Three different lighting conditions: (a) dark (b) normal and (c) bright from the same environment. 96

4.16 Average distances traveled per failure of three controller types under different lighting conditions. 97

4.17 Average distances traveled per failure of three controller types under different heights. 97

4.18 The proposed VAE-based controller versus a visual-teach-and-repeat controller. (a) Trajectories of both controllers in the training (demonstrated) map, (b) Trajectories of the VAE-based controller in the testing map, and (c) Maximum distance that the agent traveled before a failure in the training and testing map. . 98
4.19 Histogram of pilot demonstration data compared to the model prediction. The pilots are divided into three groups evenly, and each group contains five members. Each group is trained on a different controller type: (a) linear regression-based controller; (b) end-to-end neural network controller; and (c) VAE-based controller. 100
4.20 Visualization of the neural network decision by showing: (a) the input image (b) the VAE-based controller activation map, and (c) the end-to-end neural network controller activation map on the same input image. 102
4.21 The typical failure locations (a) thin branches and leaves (b) no river in the field of view, and (c) shadow from trees. 102

5.1 The drone is flying autonomously in this orchard without GPS and knows how to avoid the tree branches along the way. The view of onboard camera is provided. 105
5.2 The structure of the variational autoencoder network. 112
5.3 Sample image data collected in a variety of seasons, weathers, and times in a day from the field. 113
5.4 The custom-built quadcopter platform equipped with 1) Intel® Realsense D435i camera; 2) Intel® Realsense T265 camera; 3) Lightware SF11/C laser rangefinder; 4) Tattu 4s 5200mhA LiPo battery; 5) Pixracer R15 flight controller; 6) mRo Ski Telemetry radio; and 7) NVIDIA Jetson Xavier NX. 115
5.5 System architecture diagram. 116
5.6 A filtered point cloud generated from the experiment data. 116
5.7 Top-down view of the trajectory performed by the VAE-based controller after one iteration of training. 117

5.8 Top-down view of the trajectory performed by the VAE-based controller after three iterations of training. The views from the onboard camera at different locations are provided. 118
5.9 Top-down view of the trajectory performed by the VAE-based controller in a generalization environment. 118
5.10 The box plot of human intervention rate against training iterations for all three controllers. 119
5.11 Average distance traveled before a failure for all controllers in training and generalization environments. 122
5.12 Average distance traveled before a failure under different forward speeds. . . . 123
5.13 Histogram of human demonstration compared to the model prediction. The horizontal axis is the normalized control command and the vertical axis is the probability density. The model prediction comes from (a) baseline1 controller, (b) baseline2 controller, and (3) VAE-based controller. 125
5.14 Locations where the policy is more likely to fail (a) missing trees on the side; and (b) at the end of the row. 126

List of Tables

2.1 The modified Crazyflie 2.0 parameters. 17

2.2 The goodness of fit of each model with two rotors active. 23

2.3 The goodness of fit of each model with four rotors active. 24

2.4 The parameters used in the attitude controller. 29

2.5 The parameters used in the position controller. 31

2.6 Performance index for $Z/R = 4$. 42

2.7 Performance index for $Z/R = 3$. 43

2.8 Performance index for $Z/R = 2$. 44

3.1 The RMSEs of two sensor readings under different scenarios. 51

3.2 Trim states under different wind conditions. 60

3.3 Identified state and control derivative values. 63

3.4 Transfer function cost. 64

3.5 Identified parameters in G_w matrix. 66

3.6 RMSEs under different wind conditions and with different models. 68

3.7 Eigenvalues and modes under different wind conditions. 69

4.1 The expertise level of different human subjects. The human subjects are split into three groups and each group consists of both new and expert pilots. 87

4.2 The level of difficulty for different maps. 87

4.3 T-Test results with a significance level $\alpha = 0.10$. 92

4.4 Numerical values for the processing time of three different controllers. 95

5.1 T-Test results for human intervention rate with a significance level $\alpha = 10\%$. . 120

5.2 The onboard processing time comparison. 122

ABSTRACT

Data-Driven Modeling and High-Performance Control of Multirotor Unmanned Aerial Vehicles in Challenging Environments

Multirotor unmanned aerial vehicles (UAVs) have gained significant popularity in recent years due to their high maneuverability and vertical take-off and landing capability. The new roles require that the future multirotor UAVs will need to fly in a variety of challenging environments and the flight performance may significantly degrade due to the shift from nominal flight conditions. As their usage expands to increasingly challenging environments, the need for reliable and high-performance flight behavior becomes more pressing. The dissertation addresses these difficulties through a series of research efforts aimed at improving the overall flight performance of multirotor UAVs in challenging conditions.

First, an experimental study was conducted to identify a data-driven ground effect model for a small quadcopter, which takes into account the interference among the rotors and was validated through flight experiments. An adaptive control scheme was then developed to counter the model uncertainty resulting from the complex aerodynamics, leading to improved command tracking performance when the UAV is in the ground effect region. The effectiveness of the developed controller was demonstrated on a real quadcopter, with results showing superior performance compared to a traditional PID controller.

Second, the effect of wind on a hovering octocopter was investigated and modeled through field experiments. A data-driven approach was used to model the wind effects on the bare airframe by directly measuring the wind and including it as a control input. A state space model that explicitly considers the wind effect was identified from real flight data using a system identification approach. The validation results show that a significant error reduction can be achieved by considering wind effects and adding a correction term. The identified model can serve as a foundation for the future development of model-based controllers for outdoor multirotor aircraft, enhancing their flight performance in windy conditions.

Lastly, a vision-based control solution was developed in order to navigate the UAVs inside complex, unstructured, and GPS-denied environments. The proposed solution leverages imitation learning and a variational autoencoder neural network to enable the autonomous agent to learn reactive strategies from human experience effectively and efficiently. The learning framework and the developed controller were demonstrated in simulated riverine environments first and then validated in a real orchard on a custom-built quadcopter, with results outperforming existing baseline algorithms. The proposed vision-based control solution is expected to significantly enhance the performance of multirotor UAVs in complex and GPS-denied environments, where traditional navigation methods may not be applicable.

ACKNOWLEDGMENTS

My first and deepest thanks go to my advisor, Professor Zhaodan Kong, for his support and guidance throughout my graduate program. He gives his students the best resource and freedom to conduct their research. I would not have made these successes without his advising and help over the past six years. He is a great research lab leader and a nice mentor to work with.

I am deeply grateful to Professor Stavros G. Vougioukas and Professor Iman Soltani, my esteemed committee members, for their invaluable guidance and feedback throughout my research and dissertation.

I would like to thank my colleagues from the Cyber-Human-Physical Systems Lab. The experience of working at this lab is impressive and enjoyable. I luckily got a lot of support from the current and former lab members in pursuit of my academic goals. Many brilliant ideas came from the talks with these smart people, either in lab meetings or during weekly catch-ups. Their feedback always sparked new ideas and helped me strive to achieve and do more.

I also want to thank my friends, Gang Chen, Chen Peng and Zhenghao Fei. The academic discussions with them during our past daily coffee breaks were very constructive and guided me through many technical problems. They also provided me with sound career advice. I am grateful to have these friends with me and I wish them bright futures in their careers.

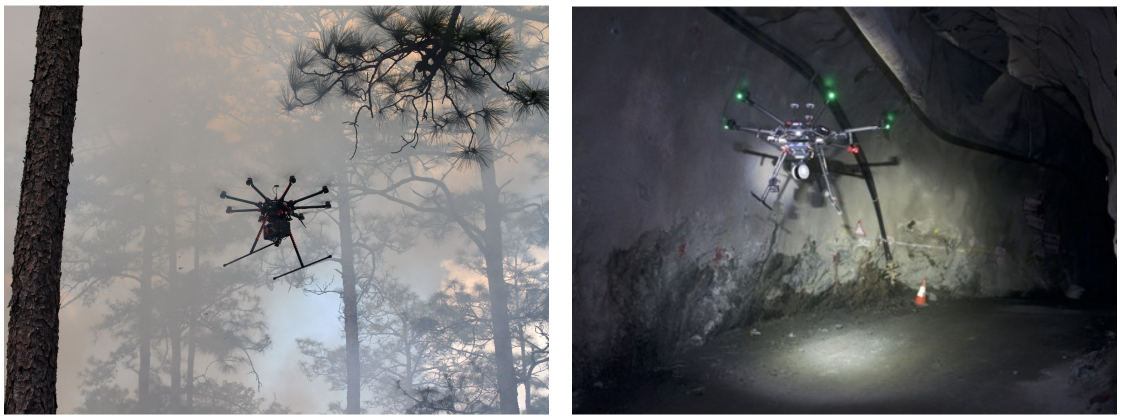
Lastly, I would like to thank my girlfriend, Xinge Zhang, for her accompaniment in my daily life and support during the most challenging time. I would also like to thank my parents for always being there and supportive in my long journey to completing my Ph.D. degree.

Chapter 1

Introduction

1.1 Motivation

Multirotor unmanned aerial vehicles are becoming increasingly popular over the past few years due to their high maneuverability and vertical take-off and landing capabilities. They have been used in a wide range of military missions, such as intelligence, surveillance, and reconnaissance [3, 4], and various civilian missions, for example, remote inspection [5], precision agriculture [6], search and rescue [7], aerial photography [8], site surveying [9], transportation and delivery [10, 11, 12]. However, these new roles require that the future UAVs will need to fly in many challenging environments (e.g., see examples in Fig. 1.1) which the flight performance may significantly degrade due to the shift from nominal flight conditions. Flying in



(a) Flying over a wildfire. (b) Flying in an underground mine.

Figure 1.1: Examples of UAV operating in challenging environments [1, 2].

those challenging environments or conditions may generate unexpected and unsafe behaviors and, sometimes, lead to crashes and property damage. As a result, how to provide high performance and reliable behaviors when the UAV is operating under challenging environments is gaining more and more attention. Three challenges are introduced in detail and explain why the proposed research is necessary.

The first challenge that most UAVs are facing is when the vehicle is flying near the ground (see Fig. 1.2). The operation of these aerial vehicles at low altitudes and near the ground raises safety concerns because an accident is likely to happen when a UAV leaves or enters the ground effect region. The ground effect can also introduce uncertainty into the thrust generated by the propulsion system, which can negatively affect flight behaviors. To improve safety and performance, it is crucial to understand the fundamental relationship between the UAV dynamics and the nearby surfaces and objects. Besides, developing a comprehensive controller that accounts for the ground effect is necessary to optimize the UAV's performance during missions that require it to operate in proximity to the ground.

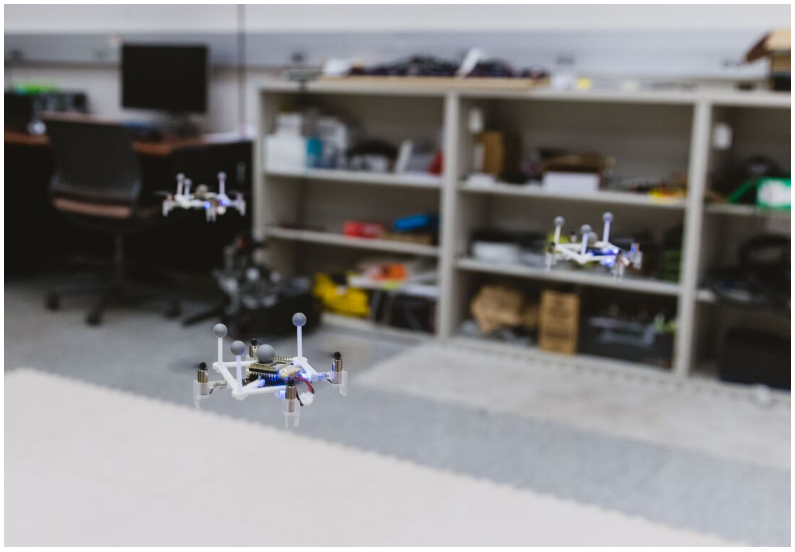


Figure 1.2: Example of multirotor UAVs taking off and landing in ground effect regions.

The second major challenge in the field of multirotor UAVs is to ensure reliable flight performance in outdoor environments with unpredictable environmental disturbances, such as strong winds (see Fig. 1.3). The forces and moments acting on a multirotor UAV can significantly change due to the complex aerodynamics, which can result in reduced overall flight performance. Thus, it is crucial to investigate and model the effects of wind on multirotor UAVs to improve the control system's effectiveness. By thoroughly understanding the wind effects, we can better equip the control system to handle environmental disturbances and ensure reliable and efficient flight performance.

Figure 1.3: Examples of multirotor UAV operating in windy conditions.

Figure 1.4: Examples of multirotor UAV operating in GPS-denied natural environments.

The third challenge is how to enable a UAV to fly in complex and GPS-denied natural environments (e.g., under the canopy in heavy foliage or riverine) effectively and efficiently (see Fig. 1.4). It is critical that the UAV should be able to navigate in these environments and avoid obstacles with less human assistance. This is especially useful when, for example, the UAV is serving as a wingman for the Marine Corps, providing reconnaissance information about terrain, friendly troops, and potential threats. Currently, flying in those complex and unknown environments remains a big challenge and requires significant input from human pilots. One possible solution is to rely on exteroceptive sensors, such as cameras, and develop advanced control strategies that allow the UAV to navigate autonomously with minimal human interventions. The use of artificial intelligence and machine learning algorithms can be particularly valuable in addressing this challenge. By incorporating these technologies, we can enable the effective and efficient use of multirotor UAVs in those complex and GPS-denied environments.

1.2 Background

The following provides a summary of the literature related to the aforementioned topics and challenges. Detailed discussions and developments of the methodology to address each challenge are presented in the corresponding chapters.

1.2.1 Multirotor UAV Modeling

Compared to traditional fixed-wing aircraft, the modeling of multirotor UAVs is not as well-developed and researched. Physical-based first-principle approaches, such as the blade element theory and momentum theory [13], have been used to determine the forces and moments generated at the rotor plane of a multirotor aircraft [14, 15, 16]. However, these analytical equations were developed based on full-scale aircraft and have not been fully validated for small multirotor UAVs. Modeling the complex interactions between multiple rotors and between rotors and the aircraft body, particularly under environmental disturbances, is challenging and often inaccurate. Another physical-based strategy for modeling the complex flow between rotors and bodies is through the use of computational fluid dynamics (CFD) [17]. CFD models provide valuable insights into the aeromechanics of multirotor UAVs, but they are complex and computationally expensive, making them more suitable for simulation and analysis rather than flight control design.

Data-driven approaches have become increasingly popular for modeling multirotor UAVs due to their greater flexibility and wider range of capabilities when compared to traditional physical-based approaches. Although wind tunnel experiments have been a long-standing method for measuring the aerodynamic properties of fixed-wing and rotary-wing aircraft, their use in modeling multirotor UAVs has been limited. Recent research by NASA Ames has sought to quantify the lift and drag changes of a few commercial multirotor UAVs at different wind speeds and attitudes [18], demonstrating the potential for wind tunnel experiments to contribute to the modeling of multirotor UAVs. However, wind tunnels can be expensive to use, and the data obtained has limitations. One significant limitation is that wind tunnel experiments alone cannot accurately determine the dynamic characteristics of the aircraft, and free flight tests are necessary to obtain a more accurate model of the aircraft.

System identification is another appealing data-driven approach that has been successfully used in the modeling of full-scale fixed-wing and rotary-wing airplanes [19]. In this approach, the aircraft to be identified must execute specific maneuvers to enable effective excitation of its dynamics. The model parameters of the aircraft can then be estimated by analyzing the flight data collected in either the time domain or frequency domain [20, 21, 22, 23]. This approach has also been applied to the modeling of small UAVs [24, 25, 26]. Small UAVs usually have unique configurations, and the aerodynamic interactions between their various components can be quite complicated. The system identification approach is a useful tool for modeling these complex interactions and can provide valuable insights into the behavior of multirotor UAVs during flight operations.

Considering those complex aerodynamic effects, the ground effect is a phenomenon that has potential benefits and dangers for most aerial vehicles, including multirotor UAVs. When operating near the ground or other flat surfaces, ground effect increases the efficiency of the rotor system by decreasing the downward velocity of air and reducing induced drag. This allows vehicles to hover with less power. However, extra caution must be taken when entering or exiting the ground effect region to avoid sudden changes in thrust that can result in accidents. Mathematical models for ground effect on fixed-wing aircraft and helicopters have been developed in literature [27, 28, 29], but ground effect on multirotor UAVs is not as well-studied. While an approximated ground effect model for a single rotor has been proposed [30] and is commonly used [31, 32, 33, 34], it is not suitable for multirotor UAVs due to the unique airflow around multiple lifting rotors [35, 36, 37]. This brings the motivation of developing a ground effect model specifically for multirotor UAVs.

Wind and turbulence can greatly impact the stability, speed, and maneuverability of multirotor UAVs, especially when flying outdoors. Gusts of wind can cause the UAV to become unstable, making it challenging for multirotor UAVs to maintain stability since they require precise control of their propellers. Wind can also reduce the UAV's speed and maneuverability, making it less effective for certain applications. The Dryden spectral model, commonly used for fixed-wing airplanes [38], assumes a "frozen-field" and may not be suitable for low-speed rotorcraft since the mean wind speed becomes dominant [39]. Existing research has used blade element theory and momentum theory to derive the forces and the moments generated at the rotor plane of a multirotor aircraft [16]. The blade flapping effect has also been studied and adapted for multirotor aircraft [40, 41]. However, these methods may not be suitable since they are derived from a single-rotor model. Recently, researchers proposed a method called control equivalent turbulence inputs (CETI) to model the turbulence on a low-speed helicopter and treated the turbulence as an equivalent control input [42]. This approach has proven successful in capturing turbulence effects on a UH-60 helicopter [43] and a 3DR Iris+ quadrotor [44]. An approach of directly modeling wind as a control input and explicitly deriving its effect on multirotor UAV dynamics can be a useful approach for developing control systems that are robust to varying wind conditions.

1.2.2 Multirotor UAV Advanced Control

The field of multirotor UAV control has made significant progress in recent years. For basic control tasks like hovering and slow movement, proportional–integral–derivative (PID) [45] and linear quadratic regulator (LQR) [46] controllers have been successfully implemented. For more complex maneuvers, advanced controllers that consider the aircraft's non-linearity have been developed [47, 48, 49]. Model predictive control (MPC) schemes have been utilized to achieve better flight performance while satisfying system constraints [50]. Despite their success in controlled conditions, these controllers struggle in the presence of model uncertainties, spurring renewed interest in adaptive control techniques. Adaptive control handles uncertainties by varying gains in-flight according to an adaptive law. The model reference adaptive control [51, 52] and L1 adaptive control [53] approaches have been commonly studied. For example, the authors in [54] proposed an adaptive tracking controller for a quadcopter based on output feedback linearization to solve the problem of center of gravity shifting in flight. Similarly, researchers focused on altitude control for quadcopters have implemented model reference adaptive control to retain performance when the mass of quadcopter changes in-flight [55]. In certain instances, such as aircraft damage, adaptive control has shown promising results towards recovering stability and control [56].

For autonomous navigation tasks in UAVs, a common approach is to break the system into different modules, including perception, planning, and control [57]. Each module can be tackled separately. For example, a global or local map of the environment can be built from perception sensors and the robot's state is estimated simultaneously. A feasible path or trajectory can then be planned sequentially within the map to approach the target point and avoid obstacles. A tracking algorithm will control the robot to follow the planned path. While this modular design scheme has been explored extensively in the literature, it has some drawbacks, such as a high computation load and discrepancies between the planning and deployment due to the hierarchical task decomposition [58]. The question arises whether this modular design structure is necessary. As we know, humans can fly a drone purely relying on visual inputs (e.g., first-person-view camera). They map the visual input directly to control commands and are able to provide good reactive behaviors in unknown environments [59]. A similar reactive strategy can be developed without a need for global planning, and such a visuomotor control policy can be trained using modern learning-based approaches.

Among these approaches, reinforcement learning (RL) finds the optimal policy by maximizing a reward function. RL has been successfully applied in a wide range of applications. For example, [60] used deep Q-network to train an agent playing Atari games based on pixel inputs. [61] trained a deep neural network with proximal policy optimization (PPO) to fly drones across race tracks at fast speed. Recently, [62] solved the autonomous car racing problem using a soft actor-critic (SAC) algorithm and the policy achieved super-human performance. [63] successfully trained quadruped locomotion in real world with deep reinforcement learning. Nevertheless, RL is notorious for its bad sample complexity. Generally, it requires a substantial amount of data to obtain good results and the training may converge slowly. The trial-and-error learning process also causes severe problems for safety-critical systems.

Imitation learning (IL), on the other hand, learns sequential decision-making policies from expert demonstrations [64, 65]. The author in [66] successfully trained a drone to fly inside low-altitude forests and perform tree avoidance with imitation learning. Although the early stage of IL was limited to relatively low-dimensional problems, the rise of deep learning techniques in recent years has provided powerful solutions through the appealing properties of deep neural networks and enable the scaling of IL to more complex and high-dimensional problems [67, 68, 69]. Imitation learning is attractive for applications where interaction with the real environment could be dangerous (e.g., safety-critical systems) and the training data are available from experts. It can overcome many of the limitations of reinforcement learning for the learning agent. For example, it can significantly reduce the sample complexity and eliminate the need for hand-crafted rewards, enabling the agent to learn a good policy directly from expert demonstrated dataset.

1.3 Contributions

The contribution of this dissertation are as follows:

- A linear ground effect model for a mini quadcopter was obtained through ground experiments and validated in flight. The influence of separation distances between rotors on the ground effect was studied through experiments and a transition from linear to quadratic model was observed when the rotor distance becomes large enough.

- A control architecture that utilizes MRAC was developed and implemented on a real quadcopter. The MRAC was added to the altitude control loop to overcome the ground effect and its performance was evaluated through a set of flight experiments. The controller dramatically outperforms a traditional position controller in tracking the altitude command when the vehicle is in the ground effect region.

- A data-driven approach was developed to model the effects of wind on multirotor UAVs. The wind can be included as a control input to the state-space model of the bare-airframe dynamics, which is obtained by a system identification approach in the frequency domain.

- The effects of wind on UAV dynamics, particularly with respect to different wind speeds, have been studied and validated from outdoor experiments on an octocopter with measured wind data. Additional discussions on how wind speed affects the damping ratio and the natural frequency of the pitch mode of the bare-airframe model are also provided. These insights can help improve the development of model-based controllers to optimize the performance of outdoor multirotor UAVs.

- An imitation learning framework was proposed and used to train vison-based navigation policies for UAVs operating in complex and GPS-denied environments. The system relies solely on a forward-facing camera to perform reactive maneuvers and guide the UAV through the environment.

- A variational autoencoder (VAE)-based was developed and the performance was compared against different vision-based controllers trained from 15 human subjects in the simulated riverine environments. Based on the results, it has been found that the proposed VAE-based controller outperforms the other controllers with a lower intervention rate from the pilot and a longer traveling distance.

- The performance of the above learning framework and vision-based control solution were verified in real orchard environments on a custom-built quadcopter platform. The VAE-based controller demonstrated superior performance to the existing baseline algorithms and the learned policy achieved a longer flying distance with less human assistance.

- The trained control policy was further evaluated in novel environments that the agent had never encountered during the training to test its generalization ability. The results demonstrate that the proposed vision-based algorithm is able to generalize well to these novel environments, with strong performance observed in both simulation and real-world experiments.

1.4 Dissertation Outline

The rest of the dissertation is structured as follows: Chapter 2 describes the effort to model the ground effect on a small quadcopter platform. The ground effect model is compared with existing models and further explored with respect to different rotor distance configurations. The design of an adaptive controller is also introduced in Chapter 2 followed by flight experiment results which evaluate its performance. Chapter 3 presents the effort to model the wind effects on a real multirotor platform outdoors. This chapter consists of the hardware description, system identification methodology, parameter identification results, and procedures to model the wind effect. The identified model and verification results are provided. In the next two chapters, a vision-based controller trained with imitation learning method is introduced and detailed explained. The results in simulated riverine environments are provided in Chapter 4. The proposed method is verified in real-world orchards with comparisons to existing baselines and these results are presented in Chapter 5. Lastly, Chapter 6 summarizes the conclusions and provides recommendations for future research.

Chapter 2

Mitigating Multirotor UAV Ground Effect using Adaptive Control

Mitigating ground effect becomes a big challenge for autonomous aerial vehicles when they are flying in close proximity to the ground. This chapter aims to develop a precise model of ground effect on mini quadcopters, provide an advanced control algorithm to counter the model uncertainty and, as a result, improve the command tracking performance when the vehicle is in the ground effect region. The mathematical model of ground effect has been established through a series of experiments and validated by a flight test. The experiments show that the total thrust generated by rotors increases linearly as the vehicle gets closer to the ground, which is different from the commonly-used ground effect model for a single rotor vehicle. In addition, the model switches from a piecewise linear to a quadratic function when the rotor to rotor distance is increased. A control architecture that utilizes the model reference adaptive controller (MRAC) has also been designed, where MRAC is added to the altitude loop. The performance of the proposed control algorithm has been evaluated through a set of flight tests on a mini quadcopter platform and compared with a traditional proportional-integral-derivative (PID) controller. The results demonstrate that MRAC dramatically improves the tracking performance of altitude command and can reduce the rise time by 80% under the ground effect.

2.1 Introduction

Multirotor aerial vehicles become popular over the past few years due to their high maneuverability and vertical takeoff and landing capability. There are growing interests in using this hardware platform for transportation and delivery services [70, 10, 11]. Small multirotor unmanned aerial vehicles, e.g. quadcopters, have also been widely used in aerial photography and site surveying in which these vehicles are flying in proximity to humans. On the other hand, the high popularity of these aerial vehicles raise concerns about the safety, especially when they are flying at very low altitude and near the ground. For a rotorcraft, an accident is likely to happen during takeoff and landing phases when the vehicle enters or leaves the ground effect region [30], which may cause property damages as well as vehicle crashes. In this work, the ground effect on a mini quadcopter is studied and a solution is introduced to guarantee the safe operation and good performance when the vehicle is flying near the ground.

Ground effect on aerial vehicles is a widely studied topic due to its potential benefits and dangers, and the effect varies significantly in flights due to several factors, including the distance from ground, type of ground, and vehicle's speed. It is at close proximity to the ground where the efficiency of the rotor system increases [71]. The ground effect can be viewed as the momentum change in a control volume that is bounded with streamlines. The streamlines near the ground are turning around and changing the vertical direction to the rotor disk to the parallel direction with the ground. These streamlines near the ground are similar to a jet flow in which the flow velocity is increased with the reduced exit area. The increased velocity at the exit of the control volume translates to the increased thrust on the rotor disk in the context of the control volume theory. The vehicles that take the advantage of ground effect have been designed in [72, 73]. Meanwhile, there are a lot of accidents due to ground effect, from incidents to fatal accidents [74, 75, 76, 77, 78]. When analyzing the ground effect, an empirical model is preferred than a computational fluid dynamics (CFD) modeling, which can be very complex for a multirotor due to the flow interaction between the rotors. Besides, a CFD simulation is not suitable for real-time control because the computation has to be repeated at different operating conditions which can be extremely time-consuming. Thus, a data-driven model is adopted in this work. In this chapter, a test platform is first built and a mathematical model of ground effect is obtained on a mini quadcopter through experiments. Considering the small size of a mini quadcopter, the ground effects are studied under different separation distances between the rotors. The results are compared to the single rotor model described in Eqn. (2.1). The obtained model is validated by a flight test and used for designing the control algorithm to mitigate the ground effect.

A model reference adaptive controller (MRAC) is proposed in this work which helps to handle the thrust uncertainty caused by the ground effect. The MRAC is added to the z direction along with the original PID position controller in the outer loop. Two different reference models have been selected to match the system uncertainty. One is a polynomial function and the other one is a set of radial basis functions. The performances are evaluated through a series of flight tests and the results are compared to a pure PID controller. The major contribution of this work is that first a linear ground effect model for a mini quadcopter is obtained through experiments. The influence of separation distances between rotors on the ground effect is studied and a transition from linear to quadratic model is observed when the rotor distance is long enough. Then, a control architecture which utilizes MRAC is developed and implemented on a real platform. The MRAC is added to the altitude loop to overcome the ground effect and its performance is evaluated through a set of flight tests. It is demonstrated that the proposed adaptive controller dramatically outperforms a traditional position controller in tracking the altitude command when the vehicle is in the ground effect region.

2.2 Related Work

The mathematical model of ground effect has been well developed for fixed-wing aircrafts [79] and helicopters [29, 80]. The fundamental ground effect model for a single rotor has been proposed by Cheeseman [30] such that

$$\frac{F}{F_0} = \frac{1}{1 - (\frac{R}{4Z})^2} \tag{2.1}$$

where F is the actual thrust in the ground effect region, F_0 is the nominal thrust outside the ground effect region, R is the radius of the rotor disk, and Z is the distance of the rotor plane above the ground. The equation holds true at a constant rotor speed. However, it was developed for a single-rotor helicopter and may not be suitable for multirotor aircraft. There have been many researches showing that the ground effect on quadcopters is stronger than what Cheeseman's model predicts [35, 81]. When two rotors are close to each other, the interaction between airflows will induce a stronger ground effect which is called the fountain effect [36, 82, 83]. A simulation of this effect on quadrotors was presented by Sanchez-Cuevas et al. through computational fluid dynamics [84]. Danjun in [85] proposed a coefficient ρ in Cheeseman's model to correct the difference between a quadcopter and a helicopter, considering the unpredictable airflow influences among the rotors

$$\frac{F}{F_0} = \frac{1}{1-\rho(\frac{R}{4Z})^2} \tag{2.2}$$

They found out the ground effect is stronger than that in Eqn. (2.1) and still measurable when $Z/R = 4$, compared to the result from Cheeseman's model which states that the ground effect is significant up to $Z/R = 1.5$. The author in [86] presented another model to capture a wider range of the height

$$\frac{F}{F_0} = ae^{-Z/b} + 1 \tag{2.3}$$

where a and b are coefficients that depend on the geometry of the blade as well as the separation distance between two rotors. They claimed that the maximum ground effect ratio has a finite value 1.7 when Z/R approaches zero. In [84], a complex ground effect model on quadrotors has been characterized and validated through experimental tests. They also identified the partial ground effect which the ground effect only occurs on part of the rotors. Besides, visual feedback was used in [87] to model the external disturbance during the experiment and a support vector regression was utilized to predict the ground effect force. Bernard in [88] characterized the ground effect in a dynamic sense for quadrotors. They accessed the effect of distance from ground on the dynamics of individual rotor as well as the overall attitude motion of the platform. However, none of these papers study the ground effect on a mini quadcopter, which has a much smaller separation distance between the rotors and may induce more interference of airflow in proximity to the ground.

The simple tasks on quadcopters, such as stabilizing at a stationary point or moving at low speed, have been successfully implemented by linearizing the system and applying a PID [45] or a LQR controller [89]. The PID controller has also been successfully tested on a multirotor aerial platform while its overhanging robotic arm is moving [90]. By using a multi-layer control architecture and nonlinear controls considering its nonlinear nature, the trajectory following and aggressive maneuvering problems have also been addressed recently [47, 48]. Although these controllers perform really well in designed conditions, they suffer performance losses in the presence of model uncertainties. It is because these controllers have constant parameters designed for determined models and conditions, thus any changes to the model will cause a performance degradation.

The concerns about uncertainty bring renewed interests in adaptive control techniques [91, 92], of which the gain is able to vary in flight according to adaptive laws. In [93], the optimal path tracking problem has been solved by using a linear quadratic tracking (LQT) algorithm with a varying gain in time. The proposed algorithm is robust to model uncertainty and external disturbance. Raffo in [94] used a nonlinear H_∞ controller with the assistance of a model predictive controller to solve path-following problems, under structural and parametric uncertainties. In [54], an adaptive tracking controller was applied on a quadcopter and they proved its great tracking performance with the center of gravity (CoG) shifted in flight. The adaptive controller has already been used to address the model uncertainty caused by the ground effect. In [95], a nonlinear adaptive backstepping controller was used to control the altitude of a small-scale helicopter during takeoff and landing while a horizontal wind gust is present. In [86], the authors designed an adaptive nonlinear disturbance observer (ANDO) to enhance the PID controller. The settling time with ANDO implemented was largely reduced in the simulation, however, the experimental validation is lacking. Also, the ANDO was added into the inner loop which may cause an issue for the platform with limited computing power. Researchers in [55] implemented an MRAC to assist an existing PI heave-velocity stabilizer, where changes in mass greatly reduced the system performance. Their result shows that the proposed controller is able to retain its performance when the mass of quadcopter changes in flight.

2.3 Hardware Description

2.3.1 Test Quadcopter

The aerial vehicle platform is the Crazyflie 2.0 designed by Bitcraze equipped with customized components as displayed in Fig. 2.1. The laser-cut frame is made of a $1/16''$ thickness acetal resin sheet which provides enough strength. The reflective markers, together with the Optitrack, a motion capture system, are used to track the vehicle's position and orientation. The reflective markers are attached through plastic slotted screws which are glued to the designed holes on the base frame. The entire structure is tightened with rubber bands at four arms. The Crazyflie 2.0 communicates with the computer over a Crazyradio PA radio. The control over the vehicle can be extended through a modified firmware onboard.

A 3D model was created in Solidworks after measuring the dimension and weight of each piece. The total mass m and the length of arm L were read directly, while the moment of inertia I_{xx}, I_{yy} and I_{zz}, were calculated using the Solidworks Toolbox. The values of these parameters are provided in Table. 2.1. The motor force constant k_f has an important role in studying the ground effect and thus was determined from an experiment. In the experiment, the throttle was increased manually from 0% to 100% in an increment of 10%. The corresponding motor signal (pulse-width modulation signal, PWM) and the thrust generated by one rotor were recorded. A linear model was fitted to the data and related the PWM signal to the thrust. The slope of the linear model was determined as k_f. The motor moment constant k_m was obtained from [96] and related to the PWM signal as well. The motor constant ratio γ was calculated by k_m/k_f.

2.3.2 Ground Effect Test Rig

In a ground test, the total thrust is measured by a digital scale, on which the quadcopter is placed upside down with a 3D-printed support. The support elevates the quadcopter from the scale by three times the diameter of its rotor, such that the intake air flows freely without interference. The scale is placed on a flat platform and an ultrasonic sensor is attached to its upper surface. The ultrasonic sensor is used to measure the distance from the platform to the reference ground. The data collection and filtering are processed through a microcontroller (Arduino UNO) connected to the sensor. The ultrasonic sensor is calibrated before each experiment. The ground

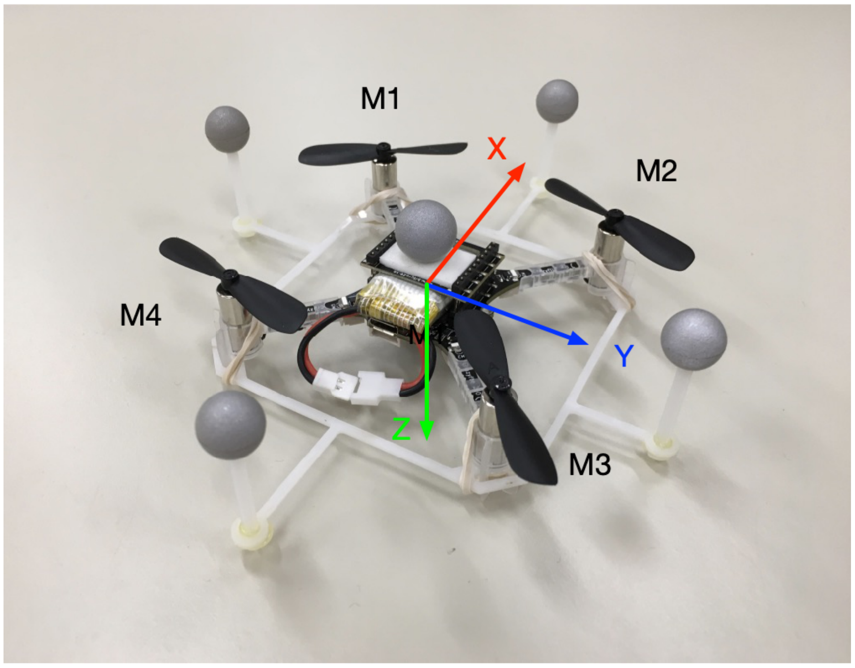


Figure 2.1: The modified Crazyflie 2.0 quadcopter with reflective markers and a laser-cut frame.

Table 2.1: The modified Crazyflie 2.0 parameters.

Parameter	Description	Value
m	Mass with markers and frame	$0.3812\ kg$
L	Distance from rotor to center line	$0.035\ m$
I_{xx}	Moment of inertia along x-axis	$2.661 \times 10^{-5}\ kg\text{-}m^2$
I_{yy}	Moment of inertia along y-axis	$2.858 \times 10^{-5}\ kg\text{-}m^2$
I_{zz}	Moment of inertia along z-axis	$5.799 \times 10^{-5}\ kg\text{-}m^2$
k_f	Motor force constant	$2.083 \times 10^{-6}\ N$
k_m	Motor moment constant	$7.707 \times 10^{-9}\ N\text{-}m$
γ	Motor constant ratio	$3.700 \times 10^{-3}\ N\text{-}m/N$

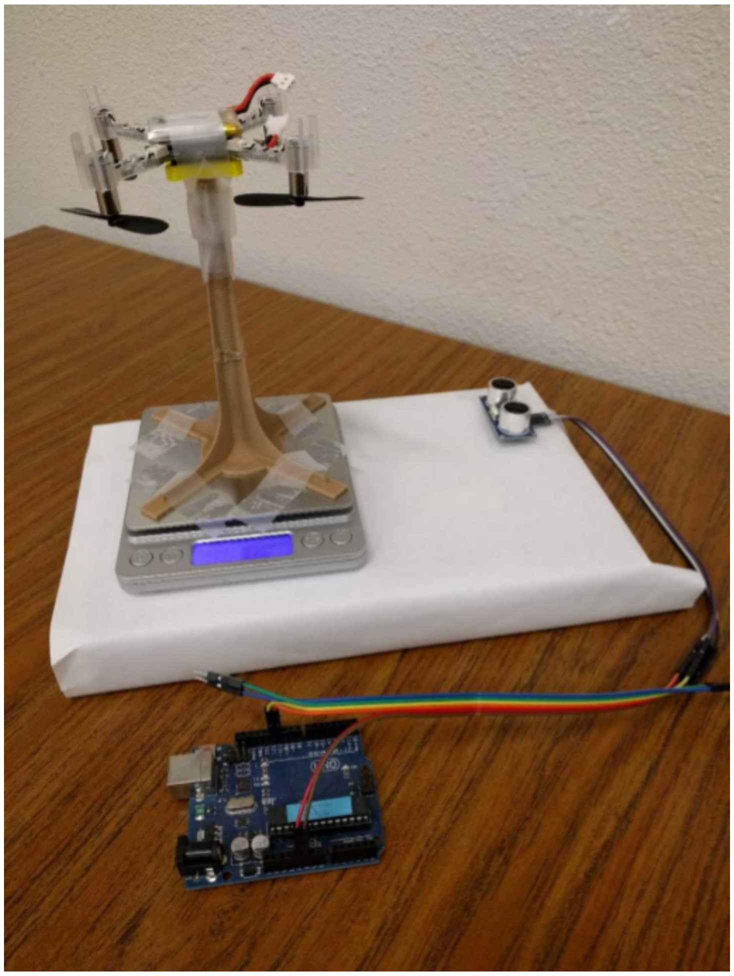


Figure 2.2: The test rig used to acquire the thrust generated by the quadcopter.

effect test rig is shown in Fig. 2.2. To emulate a ground effect in-flight and measure the corresponding thrust, the test rig is placed under a long flat table and atop a 3D-printed platform jack. The height of the platform jack can be adjusted by turning the knob on it. Since the quadcopter is placed inversely and the thrust is acting downwards, the long table emulates the reference ground and the ground effect, as a result, is able to be measured at different heights by adjusting the platform jack. A sketch of the above platform is shown in Fig. 2.3.

2.4 Ground Effect Modeling

This section presents a modeling result of the ground effect on the modified Crazyflie 2.0. Most of the existing models on quadcopters are still based on the single rotor relation described in Eqn. (2.1). This section verifies its correctness on a real quadcopter and introduces a ground

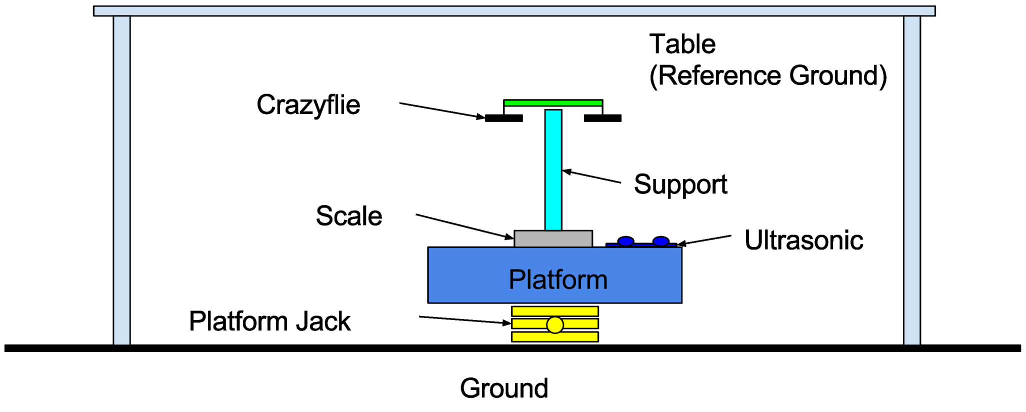


Figure 2.3: A sketch of the experiment setup used to emulate and measure the ground effect.

effect model generated from a set of ground tests and flight test validations. Note that the material of ground can impact the ground effect. For the purpose of this study, we have focused on the ground effect on hard surfaces. Future work will explore the ground effect on other materials, such as grass or soft surfaces.

2.4.1 Experiment Design

Before the ground experiment, two working assumptions have been made to let the emulation of the in-flight condition work in a ground test:

Hypothesis 1 (Working hypothesis) *Small objects placed at the center of the quadcopter have minimal effect on the airflow since it is away from the rotors.*

Hypothesis 2 (Working hypothesis) *Ground effect is affected mainly by the distance from the ground, not the vehicle's vertical velocity. Therefore, tests conducted statically on the ground have the same result as those done dynamically in flight.*

With the above two working hypotheses, the platform described in Section 2.3.2 has been successfully used to measure the thrust F at different distances above the ground. The reading from ultrasonic sensor Z_{sensor} is converted to $Z_{actually}$ based on the calibration result. $Z_{actually}$ is then subtracted by the distance between the ultrasonic sensor and the rotor plane to obtain the distance from the rotors to the table, Z. The PWM value was set to a fixed value (i.e., hovering) during the experiment. The distance was adjusted at an increment of 1 cm and the data was

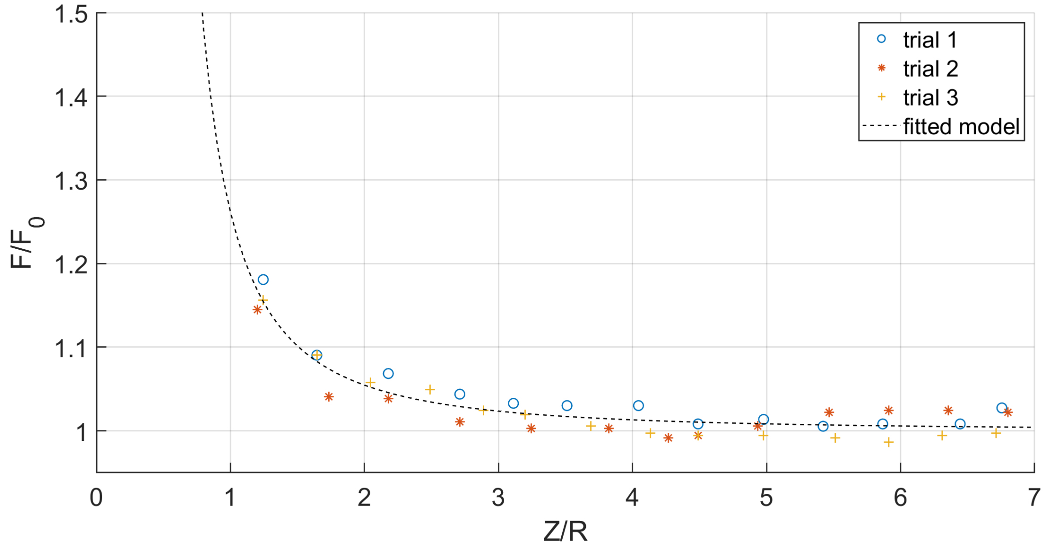


Figure 2.4: Fitting result with 2 rotors active.

recorded until the distance reaches 20 cm. After that, the test platform was relocated and the thrust F_0 outside the ground effect was measured. The radius of the rotor R can be measured directly. The experiment was conducted in two different situations, one is when only two motors (motor 1 and 3, rotate counterclockwise) are active and the other one is when all four motors are active. Due to the size of the landing gear, the minimum value of Z/R is slightly higher than 1.

2.4.2 Modeling Results

The normalized thrust and distance, F/F_0 and Z/R, are computed based on experimental results and the results are plotted in Fig. 2.4 and 2.5. Fig. 2.4 shows the result for 2-propeller case (M2 and M4 spinning in the same direction), while Fig. 2.5 is the result for 4-propeller case. For both configurations, as the vehicle approaches the ground, F/F_0 increases from 1 to 1.15 and higher. That means more thrust is generated at close proximity to the ground and is an indication of the ground effect. It is observed that there is an exponential trend for the 2-propeller case, where the vehicle experiences a 5% increase in thrust while it is 2 times its rotor's radius away from the ground. A similar model to Eqn. (2.2) is fitted to the dataset, which is

$$\frac{F}{F_0} = \frac{1}{1 - 3.315(\frac{R}{4Z})^2} \tag{2.4}$$

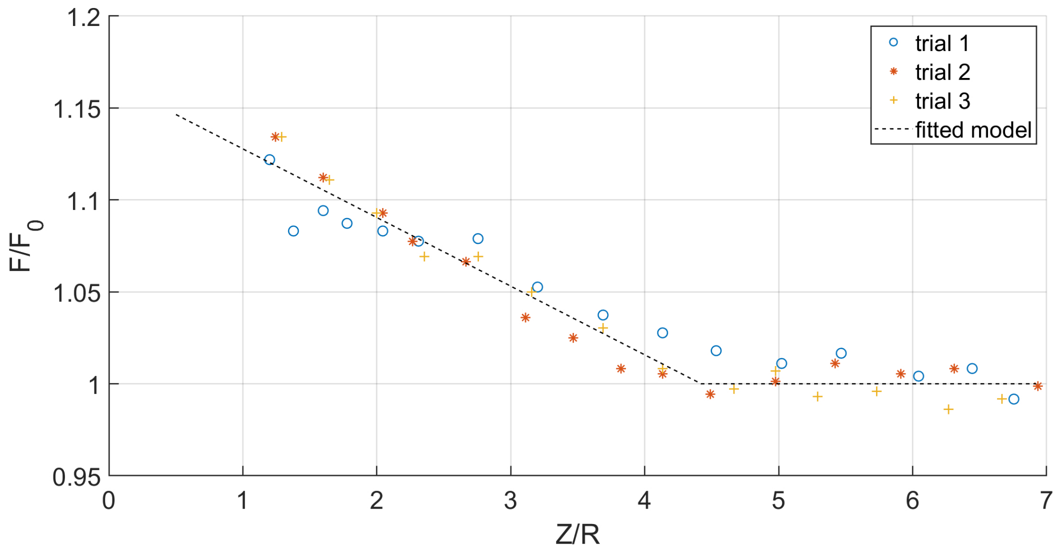


Figure 2.5: Fitting result with 4 rotors active.

On the other hand, the 4-propeller case shows a linear trend, and the vehicle only needs to reach 3 times of its rotor radius to experience the same magnitude of increase in thrust. The fitted linear model is

$$\frac{F}{F_0} = \begin{cases} -0.0373\frac{Z}{R} + 1.1651, & \text{if } \frac{Z}{R} < 4.42 \\ 1.0000, & \text{Otherwise} \end{cases} \tag{2.5}$$

The two fitted models as well as the single rotor model are put together in Fig. 2.6 for a comparison. Though the 2-rotor case shows a similar exponential pattern as the single rotor relation, the quadcopter has been found to experience more ground effect (higher value of F/F_0 indicates a more ground effect experienced), while the 4-rotor case shows a totally different pattern from the other two. The possible explanation, from an aerodynamic perspective, is that when the rotors are in close proximity, the jet flow from a rotor near the ground is blocked by the opposite jet flow from the other rotor. The blockage creates a re-circulation between multiple rotors so that the exit velocity in the control volume is reduced. This re-circulation and disturbance of flows are considered as the interactional aerodynamics between multiple rotor wakes and it becomes much stronger for four rotors compared with two rotors. The significant reduction in the lift near the ground for four rotors is revealed as the linear behavior in the ground effect. In order to confirm that the discrepancy is caused by the actual aerodynamics,

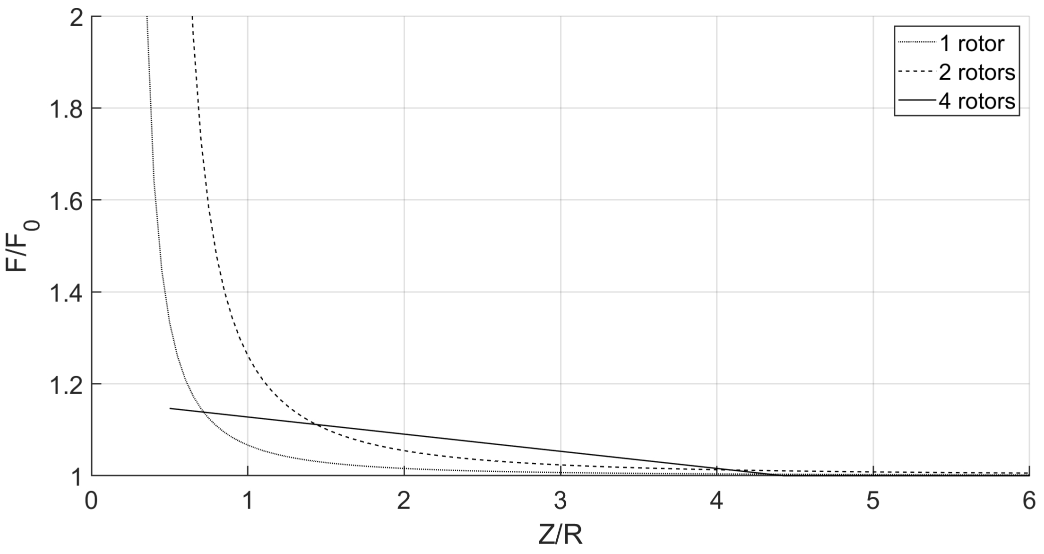


Figure 2.6: Comparison of three ground effect models.

not the experiment error, flight tests have been conducted to verify the data collected in the ground experiment.

The flight test is designed to let a quadcopter start from a height of $Z/R \approx 10$, which is outside of the ground effect region and then descend to the ground slowly and through multiple steps. During each step, the vehicle hovers for 10 seconds at a certain altitude. The Motor's PWM signal, altitude, and weight of the vehicle are recorded during the test. To ensure that the vehicle is hovering stably without any vertical motion, only the data from the second half of the 10-second duration is being used, where the net force is approximately zero. The total thrust with ground effect, F, in this case, is the weight of the vehicle. The total thrust without ground effect, F_0, is calculated by using the recorded PWM and the theoretical thrust equation as follows:

$$F_0 = \sum_{i=1}^{4} (k_f \times PWM_i), \tag{2.6}$$

where k_f is the motor force constant in Table. 2.1. The in-flight relation of F/F_0 and Z/R is plotted in Fig. 2.7 and compared with the ground test result. It has been found that the flight test result agrees well with the ground test result in which they both show a linear pattern when Z/R is smaller than 4.4, and the thrust both increase when the vehicle gets closer to the ground.

Considering the dimension of the quadcopter used in the experiment, the geometry of it

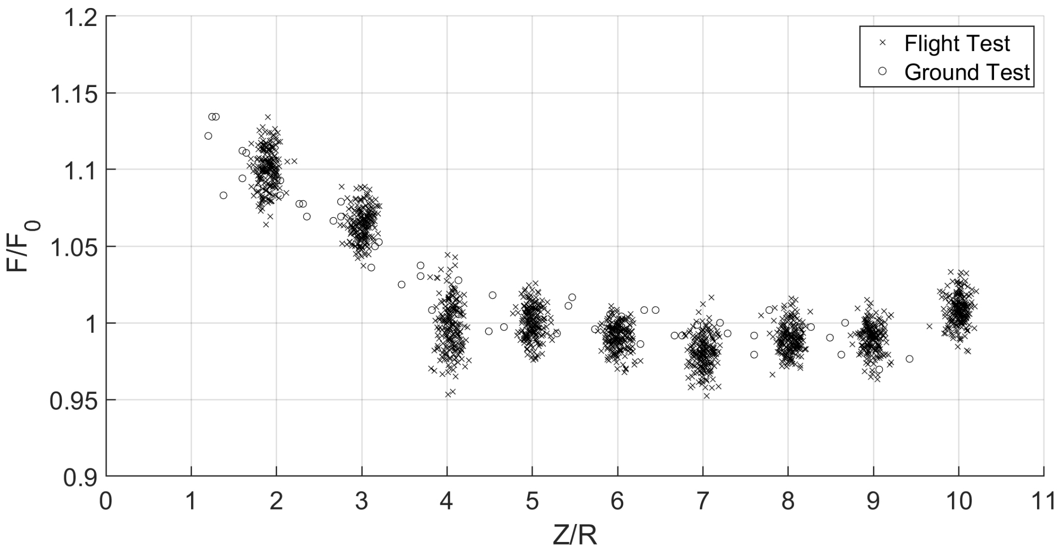


Figure 2.7: The comparison of in-flight test result and on-ground test result, where × represents the flight test data and O represents the ground test data.

Table 2.2: The goodness of fit of each model with two rotors active.

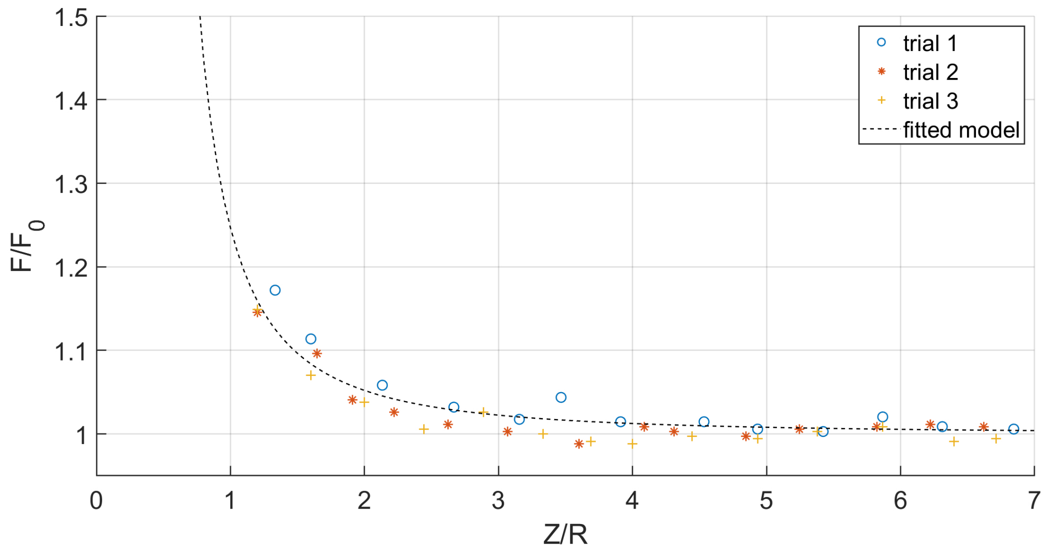
	Quadratic		
Rotor distance increment	RMSE	R square	Correction coefficient
0mm	0.0151	0.8893	3.315
10mm	0.0161	0.8740	3.166
20mm	0.0115	0.9143	2.977
30mm	0.0130	0.9221	3.097
40mm	0.0116	0.9204	2.829
50mm	0.0136	0.8889	2.788

Table 2.3: The goodness of fit of each model with four rotors active.

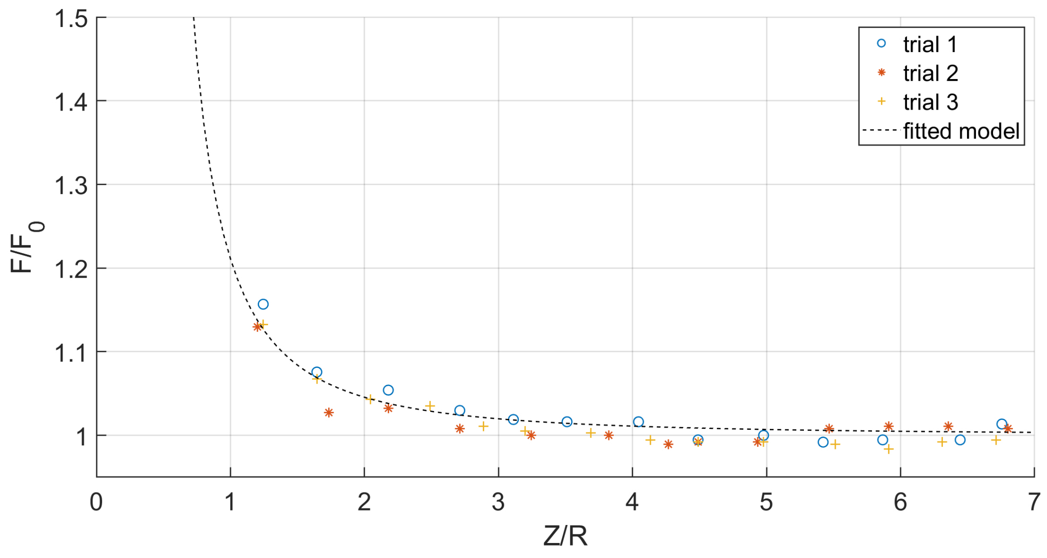
	Quadratic		Piecewise Linear	
Rotor distance increment	RMSE	R square	RMSE	R square
0mm (Piecewise linear)	0.0233	0.7246	0.0103	0.9451
10mm (Piecewise linear)	0.0139	0.8364	0.0103	0.9084
20mm (Piecewise linear)	0.0133	0.8210	0.0109	0.8779
30mm (Quadratic)	0.0099	0.9083	0.0118	0.8654
40mm (Quadratic)	0.0114	0.8962	0.0129	0.8634
50mm (Quadratic)	0.0071	0.9475	0.0086	0.9215

may have a large effect on the total generated thrust. Therefore, another set of experiments has been conducted to discover the effect of different rotor distances on a small-size quadcopter. Additional 3D printed parts are used to extend the motor arm length while keeping the geometry under the rotor plane the same, so it will only change the separation distance between rotors. The arm length is increased by 5mm, 10mm, 15mm, 20mm, and 25mm, in another word, the rotor distance is increased by 10mm, 20mm, 30mm, 40mm, and 50mm, respectively. Then the thrust and distance to the ground are measured in a similar way as what has been done in previous sections and the results are compared with the original platform (0mm represents the original platform). The equation and correction coefficient in Eqn. (2.2) are used to fit the two rotor data. The goodness of fit of each model is given in Table. 2.2, where a larger correction coefficient indicates a stronger ground effect. As can be seen, when the arm length increases, the ground effect becomes less significant (except the 15mm data point). The fitted ρ decreases from 3.3 to 2.7 when the distance between the rotors becomes larger and the results are presented in the Table. The results for 10mm and 50 mm are plotted in Fig. 2.8 in which only two rotors are active.

For the four-rotor case, the errors fitted by two models were compared: 1) a quadratic function similar to Eqn. (2.2) and 2) a piecewise linear function similar to Eqn. (2.5), and the one with a larger R^2 was chosen as the fitted model for different rotor distances. The goodness of fit of each model is provided in Table. 2.3. Based on the result, when the rotor distance



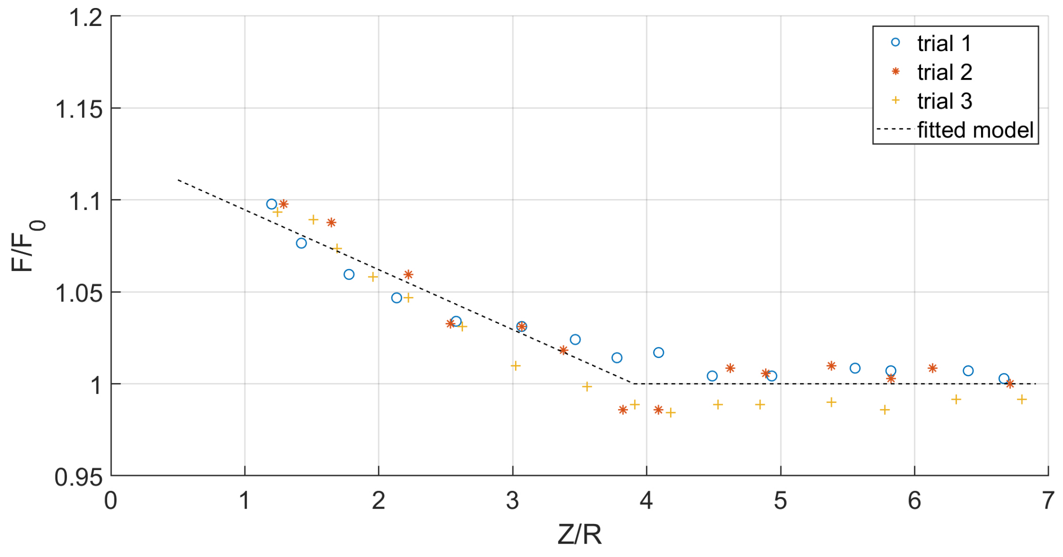
(a)



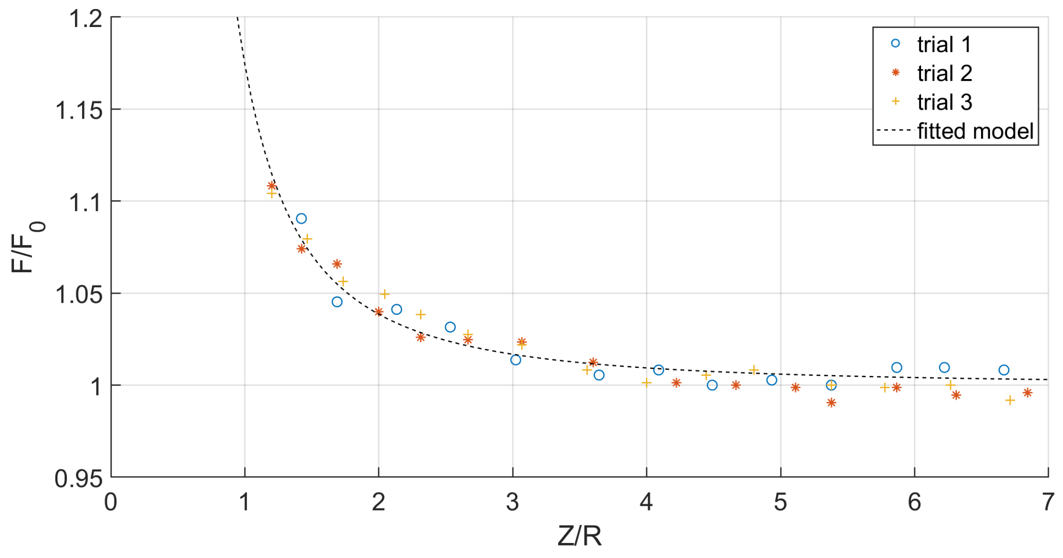
(b)

Figure 2.8: Increase the rotor distance by 10mm (a) Experimental result with 2 rotors active. (b) Experimental result with 4 rotors active.

increment is smaller than 30mm, a piecewise linear model fits better than the quadratic one. However, the difference between the two models becomes smaller and smaller when the rotor distance increment gets closer to 30mm. When the length is equal and greater than 30mm, the ground effect translates from piecewise linear to a quadratic model. The results for 10mm and



(a)



(b)

Figure 2.9: Increase the rotor distance by 50mm (a) Experimental result with 2 rotors active. (b) Experimental result with 4 rotors active.

50 mm are plotted in Fig. 2.9 when four rotors are active.

By increasing the separation distance between the rotors, it's been found out that the relation between ground effect and distance above ground becomes quadratic when the rotor separation distance is large enough and the interactional aerodynamic effect is significantly reduced. In order to understand the detailed physics of the ground effect for multiple rotors and interactional aerodynamics, high-fidelity computational fluid dynamics (CFD) simulations are suggested to be performed for multiple rotors in close proximity to each other.

2.5 Controller Design

In this section, a set of nonlinear equations obtained from Newton-Euler equations are linearized about the hovering state and used to decouple the system. A multi-layer control architecture is proposed to stabilize the vehicle and satisfy the waypoint tracking requirement. In the inner loop, an LQR full-state feedback controller plus a feedforward controller are used to stabilize the quadcopter as well as track the attitude command. In the outer loop, a PID controller is designed for position control in x, y, and z, while an MRAC is added to the z loop to mitigate the ground effect when the vehicle is in close proximity to the ground.

2.5.1 Equations of Motion

The nonlinear dynamics equations of the quadcopter are obtained from Newton-Euler equations:

$$\mathscr{F} = m \cdot \dot{\mathbf{v}} \tag{2.7}$$

$$\mathscr{M} = \mathbf{I} \cdot \dot{\omega} + \omega \times (\mathbf{I} \cdot \omega) \tag{2.8}$$

where $\mathscr{F}$ is the external force acting on the vehicle, $\Updownarrow$ is the mass of the body, $\mathbf{v}$ is the velocity vector, $\mathbf{M}$ is the external moments, $\mathbf{I}$ is the 3×3 moment of inertia matrix and ω is the angular velocity vector. The cross product terms in moment of inertia matrix $\mathbf{I}$ here are assumed to be all zeros (only use I_{xx}, I_{yy} and I_{zz}). The nonlinear differential equations are linearized about the hovering equilibrium state, and thus the position dynamics and attitude dynamics are decoupled successfully. The attitude dynamics is provided in Eqn. (2.9) and the position dynamics is given in Eqn. (2.10).

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0_{3\times 3} & 0_{3\times 3} \\ I_{3\times 3} & 0_{3\times 3} \end{bmatrix} \cdot \begin{bmatrix} p \\ q \\ r \\ \phi \\ \theta \\ \psi \end{bmatrix} + \begin{bmatrix} -\frac{Lk_f}{4I_{xx}} & -\frac{Lk_f}{4I_{xx}} & \frac{Lk_f}{4I_{xx}} & \frac{Lk_f}{4I_{xx}} \\ \frac{Lk_f}{4I_{yy}} & -\frac{Lk_f}{4I_{yy}} & -\frac{Lk_f}{4I_{yy}} & \frac{Lk_f}{4I_{yy}} \\ \frac{\gamma k_f}{4I_{zz}} & -\frac{\gamma k_f}{4I_{zz}} & \frac{\gamma k_f}{4I_{zz}} & -\frac{\gamma k_f}{4I_{zz}} \end{bmatrix} \cdot \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} \tag{2.9}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} 0_{3\times 3} & I_{3\times 3} \\ 0_{3\times 3} & 0_{3\times 3} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -g\sin(\psi_0) & -g\cos(\psi_0) & 0 \\ g\cos(\psi_0) & -g\sin(\psi_0) & 0 \\ 0 & 0 & \frac{1}{m} \end{bmatrix} \cdot \begin{bmatrix} \phi \\ \theta \\ T \end{bmatrix} \tag{2.10}$$

where $p,\ q,\ r$ are body angular rate; $\phi,\ \theta,\ \psi$ are Tait-Bryan angles, which represent for roll, pitch and yaw, respectively; $x,\ y,\ z$ are 3D positions in vehicle frame; $v_x,\ v_y,\ v_z$ are three velocities in vehicle frame; T is the total thrust in body frame; $m_1,\ m_2,\ m_3,\ m_4$ are the PWM signals for four motors, $I_{3\times 3}$ is a 3×3 identity matrix.

2.5.2 Attitude Controller

A full-state LQR feedback controller is used to stabilize the vehicle and reject the disturbances. The LQR feedback controller takes the vehicle's current Tait-Bryan angles ϕ, θ, ψ and body angular rate p, q, r as inputs and outputs the motors control signals. The LQR algorithm minimized the quadratic cost function shown in Eqn. (2.11), where $\mathbf{x} = [p, q, r, \phi, \theta, \psi]$ is the state vector, $\mathbf{u_{lqr}} = [PWM_1, PWM_2, PWM_3, PWM_4]$ is the input vector, $\mathbf{Q}$ and $\mathbf{R}$ are two weighting matrices.

$$J = \int_0^\infty \left(\mathbf{x}^T \cdot \mathbf{Q} \cdot \mathbf{x} + \mathbf{u_{lqr}}^T \cdot \mathbf{R} \cdot \mathbf{u_{lqr}}\right) dt \tag{2.11}$$

The feedback control law $\mathbf{u_{lqr}} = -K_{lqr}\mathbf{x}$, is calculated from $K_{lqr} = R^{-1}B^T P$, where P is achieved by solving a Riccati equation. However, with only the feedback control, the vehicle is not able to track the attitude command, thus a simple feedforward controller is added to the

Table 2.4: The parameters used in the attitude controller.

Parameter	Description	Value
Q	Weighting matrix on states	$100 \cdot \begin{bmatrix} I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 10 \cdot I_{3\times3} \end{bmatrix}$
R	Weighting matrix on control inputs	$10^{-6} \cdot I_{4\times4}$
k_ϕ	Feedforward gain for roll	15811.3883
k_θ	Feedforward gain for pitch	15811.3883
k_ψ	Feedforward gain for yaw	15811.3883
k_T	Feedforward gain for thrust	1.000

loop. The feedforward controller receives command vector, $\mathbf{y_{cmd}} = [\phi_c, \theta_c, \psi_c, T_c]$ and outputs the desired PWM signals, $\mathbf{u_{fwd}} = M \cdot \mathbf{y_{cmd}}$, where the mapping matrix M has the form of:

$$M = \begin{bmatrix} -k_\phi & k_\theta & k_\psi & k_T \\ -k_\phi & -k_\theta & -k_\psi & k_T \\ k_\phi & -k_\theta & k_\psi & k_T \\ k_\phi & k_\theta & -k_\psi & k_T \end{bmatrix} \tag{2.12}$$

where $k_\phi, k_\theta, k_\psi, k_T$ are the attitude loop feedforward gains for roll, pitch, yaw angle and throttle, respectively. The combination of the feedback and feedforward controllers leads to a total control output in Eqn. (2.13).

$$\mathbf{u_{att}} = \mathbf{u_{lqr}} + \mathbf{u_{fwd}} \tag{2.13}$$

The feedforward gains can be calculated by enforcing the closed-loop DC gains equal to one in order to ensure the command following. The feedforward gains, k_ϕ, k_θ, k_ψ, should be equal to the inverse of corresponding DC gains of diagonal entries for the closed-loop transfer function matrix. Note that the attitude dynamic equations do not include thrust, so k_T is set to one. Using the attitude dynamics model in Eqn. (2.9), an attitude controller is created in Matlab. The system is simulated with a step input and different weighting matrices **Q** and **R** have been tried and the one with the best performance was selected based on the system crossover frequency, steady state error, and the overshoot. The final values of the parameters are provided in Table. 2.4.

2.5.3 Position Controller

The position controller consists of a traditional PID controller plus an MRAC. The PID controller takes position errors and commanded yaw angle $[x_e,\ y_e,\ z_e,\ \psi_c]$ as inputs, and outputs the desired Tait-Bryan angles and thrust $[\phi_c, \theta_c, \psi_c, T_c]$. The relation is given as follows:

$$\mathbf{u_{pid}} = K_p \cdot \mathbf{e}(t) + K_i \cdot \int_0^t \mathbf{e}(\tau)\, d\tau + K_d \cdot \frac{d\mathbf{e}(t)}{dt} \tag{2.14}$$

where $\mathbf{e} = [x_e, y_e, z_e]^T$ and $\mathbf{u_{pid}} = [\phi_c, \theta_c, \psi_c, T_c]^T$. Since the heading angle has nothing to do with the position, no controller is put on controlling the yaw angle. The values of K_p, K_i, and K_d are tuned using the Matlab PID tuner and the values of them are given in Table. 2.5. The model created in Matlab includes the position dynamics coming from Eqn. (2.10), a second-order actuator model, which has a bandwidth of 10Hz and a damping ratio of 1, and the desired PID controller. In order to accelerate the tuning process, the inner loop is neglected here since the DC gain of the inner loop has a much faster frequency than the outer loop. The best coefficients are selected to achieve zero steady-state errors as well as overshoots.

Although a pure PID controller usually performs well under nominal flight conditions, it suffers when the quadcopter is in proximity to the ground due to the ground effect. Thus, a model reference adaptive controller is added to the position control loop. The MRAC has the advantage of adjustable parameters and a mechanism to update those parameters online, which ensures that the system states follow the model states even with the presence of uncertainties and external disturbances. In MRAC, an unknown matched system uncertainty, $F(\mathbf{x})$ is selected and added to the original linearized system (Eqn. (2.15)), which can be written as a linear combination of N provided locally Lipschitz-continuous basis functions, $\Phi(\mathbf{x})$, with unknown constant coefficients, Θ in Eqn. (2.16).

$$\dot{\mathbf{x}} = A \cdot \mathbf{x} + B \cdot (\mathbf{u} + F(\mathbf{x})) \tag{2.15}$$

$$F(\mathbf{x}) = \Theta^T \cdot \Phi(\mathbf{x}) \tag{2.16}$$

A reference model is defined as a system ($\mathbf{x_{ref}}$, $\mathbf{u_{cmd}}$) with desired performance, and the state tracking error, is the difference between the real model states and reference model states. The control input is calculated to mitigate the system uncertainty as follows,

$$\mathbf{u_{adp}} = -\hat{\Theta}^T \cdot \Phi(\mathbf{x}) \tag{2.17}$$

Table 2.5: The parameters used in the position controller.

	x direction	y direction	z direction
K_p	0.388869225	0.388869225	42692.2
K_i	-0.074870430	-0.074870430	8219.7
K_d	-0.504936570	-0.504936570	55434.8

where $\hat{\Theta}$ is the estimated unknown constant coefficients for the matched uncertainty. And a radially unbounded quadratic Lyapunov function candidate is selected to be:

$$V(\mathbf{e}, \Delta\Theta) = \mathbf{e}^T \cdot P \cdot \mathbf{e} + trace\left(\Delta\Theta^T \cdot \Gamma_\Theta^{-1} \cdot \Delta\Theta\right) \tag{2.18}$$

where $\Delta\Theta = \hat{\Theta} - \Theta$, P is the solution to algebraic Lyapunov equation for the reference model ($P \cdot A_{ref} + A_{ref}^T \cdot P = -Q$), and Γ_Θ is the rate of adaption. By differentiating Eqn. (2.18) on both sides, and keep $\dot{V} < 0$ satisfied to ensure Lyapunov stability, the adaptive law is derived in Eqn. (2.19) and updated on-line at each time step. Then the adaptive control input is calculated by Eqn. (2.17) (a detailed derivation can be found in Appendix. A).

$$\dot{\hat{\Theta}} = \Gamma_\Theta \cdot \Phi(x) \cdot \mathbf{e}^T \cdot P \cdot B \tag{2.19}$$

Notice that the ground effect has the most effect in the z (vertical) direction, thus the MRAC is only considered in the altitude loop and the closed-loop model is expected to perform like a PID altitude controller, thus $\mathbf{x} = [z, v_z]$ and $\mathbf{u_{cmd}} = [z_c]$. The reference model is found through a system identification approach by assuming that the system behaves consistently with or without the presence of ground effect. The states are recorded when the quadcopter is hovering at different heights (except the ground effect region), and a second-order state space model is used to fit the data by using the Matlab System Identification Toolbox. The identified model is provided in Eqn. (2.20) and used as the reference model for a desired performance.

$$\begin{bmatrix} \dot{z} \\ \dot{v_z} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -5.416 & -7.027 \end{bmatrix} \cdot \begin{bmatrix} z \\ v_z \end{bmatrix} + \begin{bmatrix} -0.1145 \\ 6.273 \end{bmatrix} \cdot z_c \tag{2.20}$$

With the reference model determined, the system states dependent functions, $\Phi(\mathbf{x})$ also needs to be decided to be used in Eqn. (2.16), which $\mathscr{F}(\mathbf{x})$ represents for the ground effect. Two potential $\mathscr{F}(\mathbf{x})$ functions are considered here:

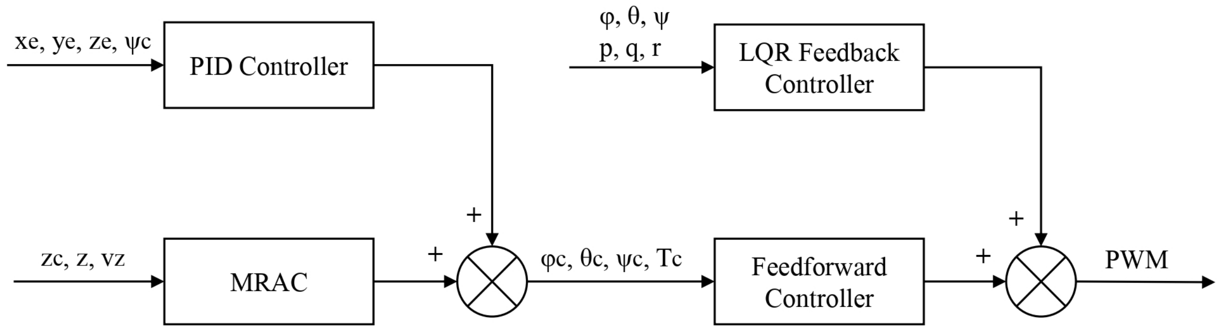


Figure 2.10: The control architecture of the system.

1. Polynomial Function: Since a linear relation is obtained between the thrust in the ground effect region and the distance from the ground in Section 2.4, which can be written in the form of $F = \theta_1 Z + \theta_0$. One simple way to select the regressor and coefficient is that

$$\Phi(\mathbf{x}) = [Z,\ 1]^T \tag{2.21}$$

$$\Theta = [\theta_1,\ \theta_0]^T \tag{2.22}$$

The Θ will be estimated online by adaptive law using Eqn. (2.19). $\Gamma_\Theta = 10^{-3} \cdot I_{2\times2}$, and $Q = [100,\ 0;\ 0,\ 1]$ are selected to ensure that the system states follow the reference states without oscillation.

2. Radial Basis Function (RBF): A RBF $\phi(x,c)$ is a real-valued function symmetric about its center c. And a linear combination of these functions can be used to approximate the given function in different fields [97, 98]. A series of *Gaussian* RBFs along with a bias term is used here to approximate the ground effect, where

$$\phi\,(x,c) = e^{-\varepsilon^2 (x-c)^2} \tag{2.23}$$

The center c of RBF is selected to be [0 m, 0.2 m, 0.4 m, 0.6 m, 0.8 m, 1.0 m], which is the height above ground and covers the entire flight envelope. ε is chosen to be 6.67. The regressors and coefficients are shown as follows,

$$\Phi(\vec{x}) = [\phi_{(z,\ 0)},\ \phi_{(z,\ 0.2)},\ \phi_{(z,\ 0.4)},\ \phi_{(z,\ 0.6)},\ \phi_{(z,\ 0.8)},\ \phi_{(z,\ 1.0)},\ 1]^T \tag{2.24}$$

$$\Theta = [\theta_6,\ \theta_5,\ \theta_4,\ \theta_3,\ \theta_2,\ \theta_1,\ \theta_0]^T \tag{2.25}$$

$\Gamma_\Theta = 10^{-3} \cdot I_{7\times7}$, and $Q = [100,\ 0;\ 0,\ 1]$ are selected.

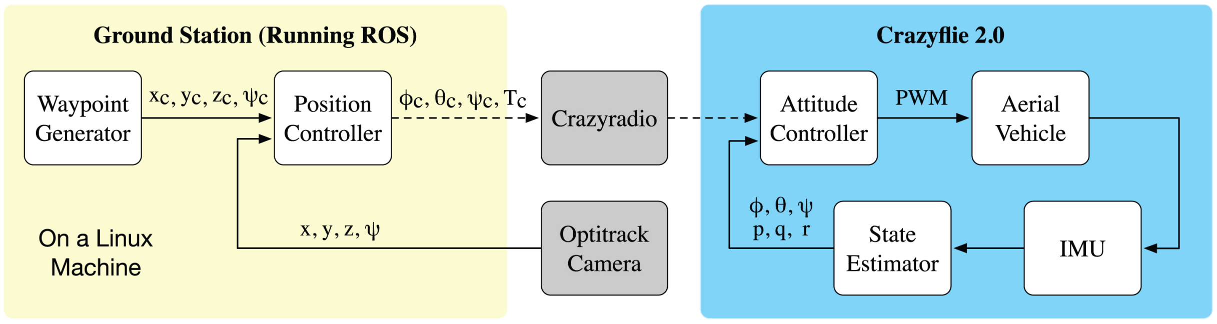


Figure 2.11: System diagram of the flight experiment.

The control output from the position controller will be the sum of PID output and MRAC output in Eqn. (2.26). As a result, the combination of the proposed position controller and attitude controller forms the complete control architecture. The complete control architecture is shown in Fig. 2.10 and its performance is tested and described in the next section.

$$\mathbf{u}_{\mathbf{pos}} = \mathbf{u}_{\mathbf{pid}} + \mathbf{u}_{\mathbf{adp}} \tag{2.26}$$

2.6 Fight Experiment Setup

The proposed control architecture has been implemented on the test quadcopter and the system diagram is provided in Fig. 2.11. The system composes of two major parts, a ground station and a Crazyflie 2.0, while they are connected through a CrazyRadio PA running at 2.4GHz. The attitude controller is written in a C program and has been flashed to the microcomputer on Crazyflie. The attitude control algorithm is running onboard at 500Hz and sends PWM commands to four motors. The embedded system also includes an IMU data processing unit, and a state estimator (a complementary filter) provided by the stock firmware running at 1000Hz, which form the inner loop altogether. In the outer loop, a waypoint generator and the position controller are implemented in a Robot Operating System (ROS) running on a Linux computer. ROS is able to handle the interface between the components and communication with the vehicle. The position controller is implemented in C and runs at 200Hz. The commanded state is created by a waypoint generator implemented in Python and running at 50Hz. The estimated positions and yaw angle are returned by the OptiTrack motion capture system, which is running at 240Hz.

A series of experiments have been conducted to evaluate the proposed control architecture.

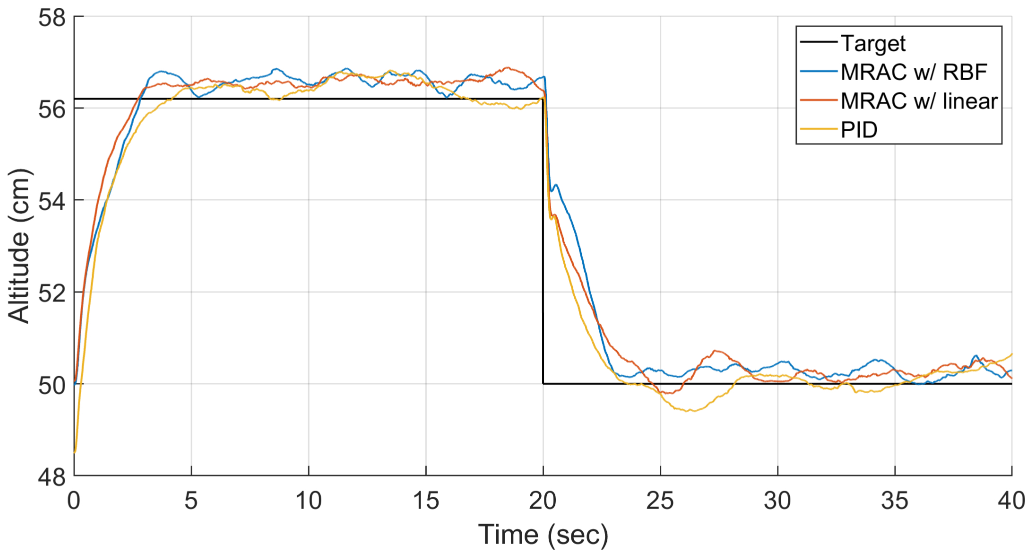


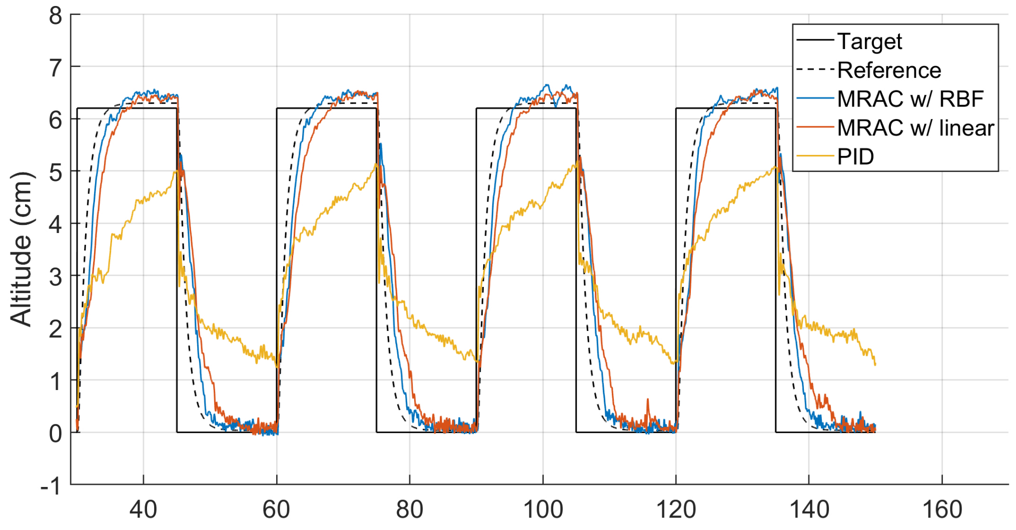
Figure 2.12: The altitude response when the vehicle is completely outside the ground effect region.

Two different uncertainty models (a linear model and an RBF model) have been implemented in our experiments and the performances are compared to a pure PID controller. The results are presented as follows.

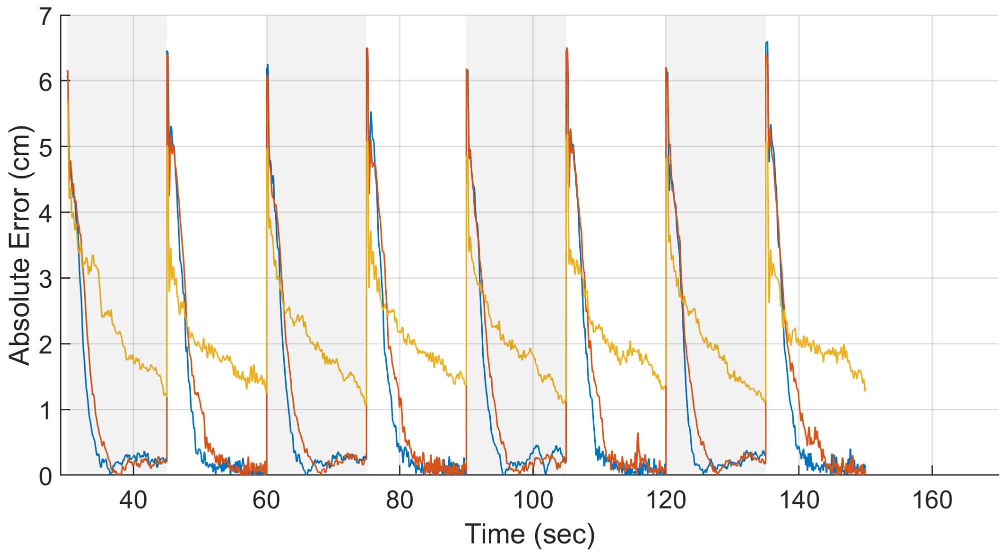
2.7 Flight Experiment Results

To verify that the MRAC has the same performance as a pure PID controller when the vehicle is outside the ground effect region, an experiment was first conducted to see the altitude response when it's high enough above the ground. The vehicle was commanded to take off at 0.5m ($Z/R \approx 22$) above the ground and hover for a certain time to ensure it has reached a steady state. A target position of 6.2cm higher than the current position was sent to simulate the takeoff pattern for 20 seconds and then the vehicle was commanded to descend back to 0.5m to simulate the landing pattern for the next 20 seconds. The corresponding altitude response was recorded. The response is plotted in Fig. 2.12. As can be seen, the three controllers have similar altitude responses when the vehicle is completely outside the ground effect region.

To see the influence of ground effect on the altitude tracking performance, the same experiment was repeated. But this time, the vehicle was commanded to take off to 6.2cm ($Z/R \approx 4$) above the ground and then land multiple times so that the entire flight is under the ground ef-



(a)



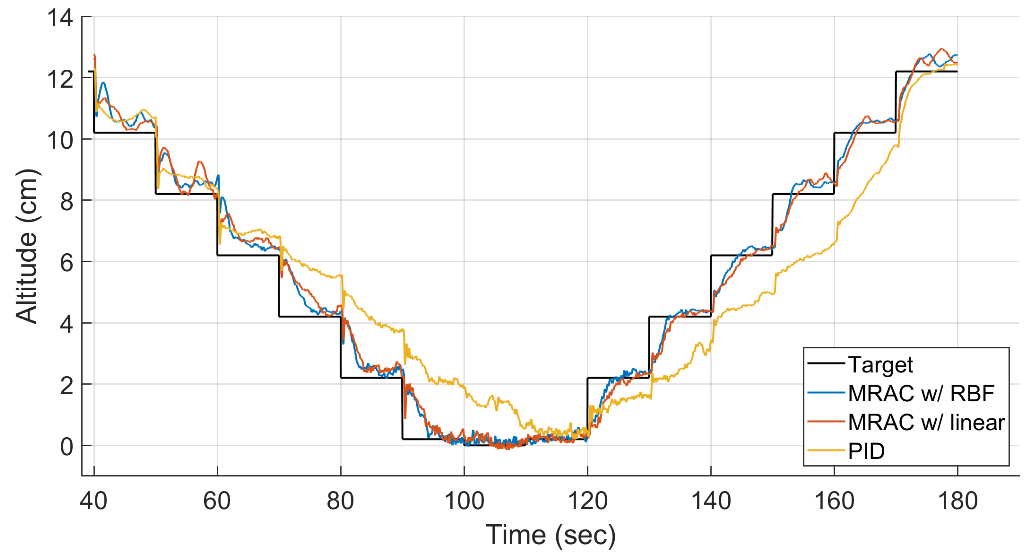
(b)

Figure 2.13: (a) The altitude response and (b) absolute error when the vehicle is under the ground effect.

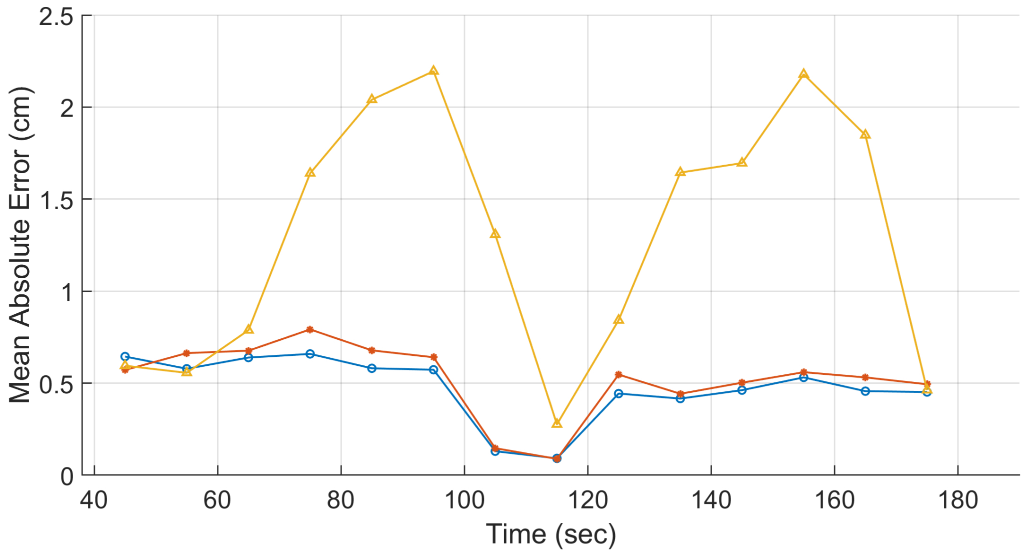
fect. The altitude tracking performance is plotted in Fig. 2.13 (a), where the vehicle's altitude responses for all three controllers are compared against the target altitude and reference altitude used by MRAC (The initial takeoff is removed). A figure of the absolute error is also provided in Fig. 2.13 (b). The shaded region represents for a takeoff phase and the non-shaded region represents for a landing phase. Based on the two figures, it can be found out that a pure PID controller is not able to track the command well in the given time period due to the ground effect. The drone is more than 1cm away from the target altitude at the end of each takeoff and landing phase with a pure PID controller and, as a result, a manual intervention is necessary to bring the vehicle down at the very end of the experiment. As a comparison, both MRAC controllers help to mitigate the ground effect significantly and allow the vehicle to reach the target altitude with an absolute error smaller than 0.25cm. It can also land the vehicle stably and safely without any human intervention. A video demonstrating different controller performances can be found here: `https://www.youtube.com/watch?v=y926-YTZKis`.

The performance at different altitudes is displayed in Fig. 2.14, in which the vehicle was commanded to hover at different distances above the ground. The time length between two sequential commands was selected to be 10 seconds and the tracking performances are compared among three controllers. The altitude response is plotted in Fig. 2.14 (a) and the calculated mean absolute error at different heights is plotted in Fig. 2.14 (b). As noticed, the MRAC and PID controllers both follow the command well when the vehicle is outside the ground effect region. The PID controller then starts losing its tracking performance when the altitude is lower than 6cm and, as a result, the vehicle can not reach the target position in the given time. The PID controller keeps suffering from the ground effect until it climbs above 10cm and regains its normal performance eventually. As a comparison, two MRAC controllers track the targets well for the entire trajectory and the MRAC with RBFs has a smaller mean absolute error than the MRAC with a linear model. The result also validates the previous conclusion that the ground effect on the test platform is significant when $Z/R \leq 4.4$.

To quantify the performance of each controller, another set of experiments has been conducted in which the vehicle was commanded to fly at different altitudes and the vehicle response was recorded during the takeoff and landing phases. The heights selected are 6.2cm ($Z/R \approx 4$),



(a)

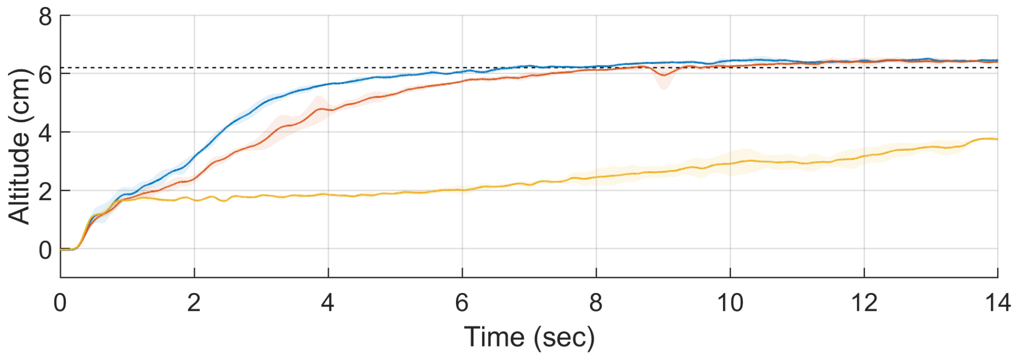


(b)

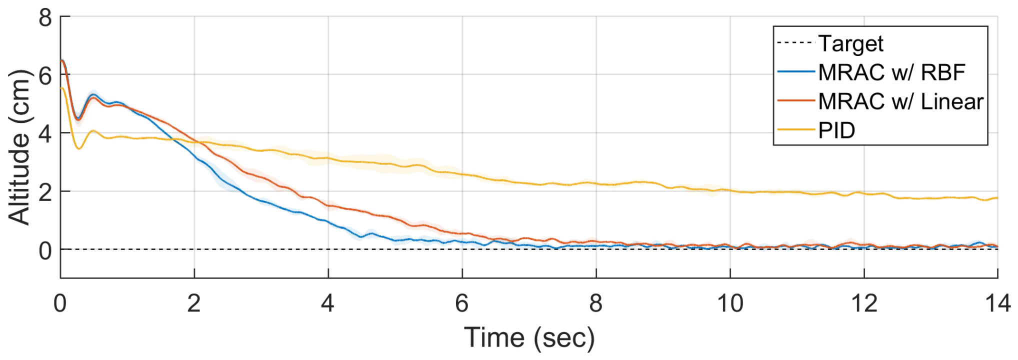
Figure 2.14: (a) The altitude response and (b) corresponding mean absolute error at different heights.

4.2cm ($Z/R \approx 3$) and 2.2cm ($Z/R \approx 2$) above the ground so that all of them are in the ground effect region. The maximum recording time for two MRACs was 20 seconds while the PID controller was selected to be 40 seconds in order to capture its full transient response. For each commanded altitude, three experiments were conducted. The average and standard deviation at each time step are calculated and an average trajectory for each pattern is generated. The average trajectories are plotted in Fig. 2.15 to 2.17, where the solid lines represent for the average trajectories and the shaded regions represent for the standard deviation. The performance indices for each altitude are extracted and provided in Table. 2.6 to 2.8, which include the steady state error with respect to the step command (SSE), the mean squared error with respect to the reference model (MSE) for two MRAC controllers, rise time, overshoot and the settling time. Note that since the altitude cannot be lower than the ground, no overshoot will occur during the landing phase. Also, since the settling time for the PID controller exceeds the maximum recording time, the value for it is not calculated.

Comparing the performance indices in three tables, the MRAC controllers have less steady-state error than the PID controller for both takeoff and landing phases. The rise time and settling time for a PID controller is quite long in the ground effect region compared to the response in Fig. 2.12 and, as a result, it is unable to track the altitude command well. The situation becomes even worse when the commanded altitude is closer to the ground. Meanwhile, the MRAC reduces the rise time by at least 80% compared to a PID controller and thus allows the vehicle to take off and land the vehicle in time. What's more, the MRAC w/ RBF has at least a 45% smaller MSE and 15% shorter rise time than the MRAC w/ linear, thus standing out among all three controllers. Although the MRAC w/ RBF shows the best performance, it needs to compute seven parameters online, compared to MRAC w/ linear which only needs to update two during the flight. The additional computational load puts some constraints on its use, for example, on an embedded system that has limited power. In that case, a linear model MRAC can be utilized since it still improves the system performance significantly while it has fewer requirements for computing power.



(a)



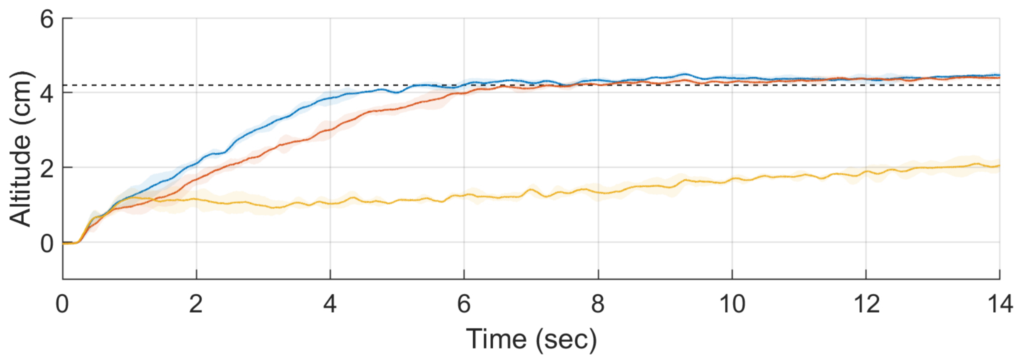
(b)

Figure 2.15: Average takeoff and landing trajectories with different control algorithms for $Z/R \approx 4$.

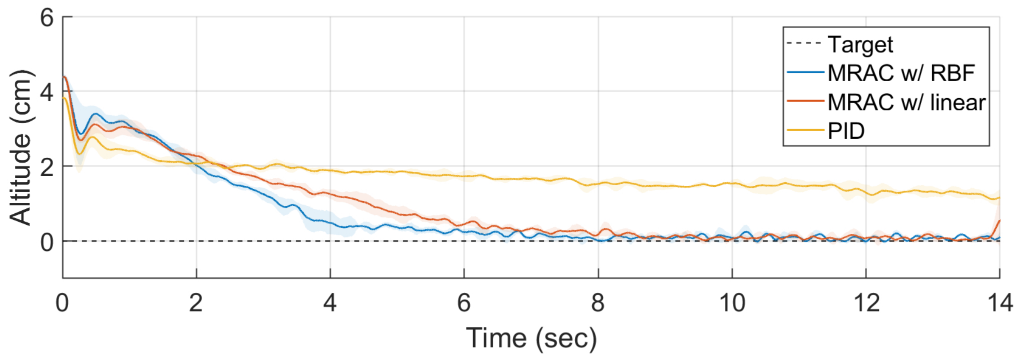
2.7.1 Discussion

In the experiment setup, the wind is not considered because the drone was flying indoors. However, the wind can cause additional aerodynamic forces and needs to be considered if flying outside and becomes more important when the wind speed is high. Modeling the wind effect will be discussed in the next chapter. In addition, a motion capture system is used to retrieve the real-time position of the vehicle. The motion capture system can achieve a centimeter-level accuracy. But for a standard drone which only has the GPS and barometer, the poorer altitude accuracy may degrade the performance of the controller. By incorporating an ultrasonic or 1D laser rangefinder, accurate height measurements above ground level can be obtained without interference from rotor airflow.

In addition, the MRAC is only added to the z direction because the tracking performance



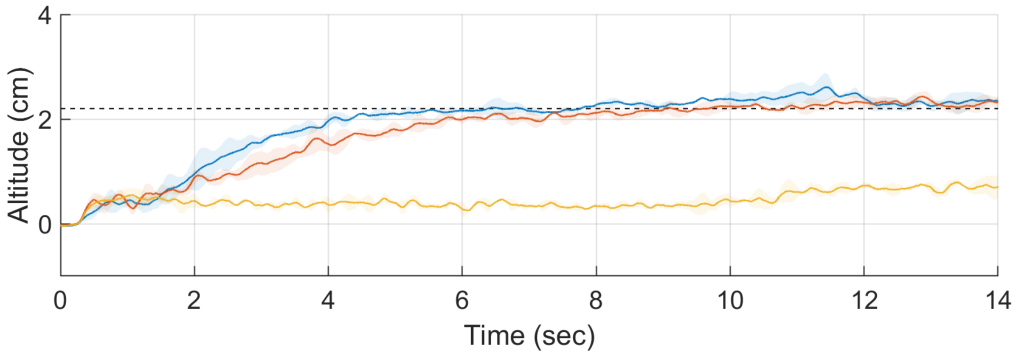
(a)



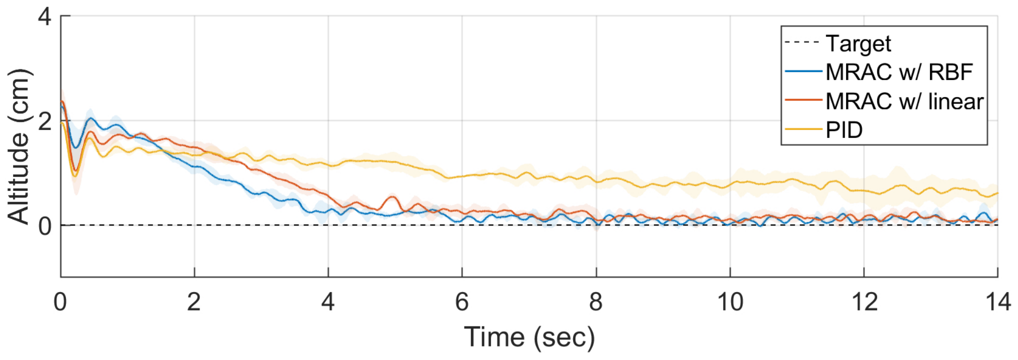
(b)

Figure 2.16: Average takeoff and landing trajectories with different control algorithms for $Z/R \approx 3$.

of altitude command is the main focus of this work. However, MRAC can also be added to the other axes if the vehicle is doing some close-ground maneuvers and with a non-zero attitude angle. Considering that a non-zero attitude angle will completely change the ground effect model and the controller design, this will be left as future work.



(a)



(b)

Figure 2.17: Average takeoff and landing trajectories with different control algorithms for $Z/R \approx 2$.

Table 2.6: Performance index for $Z/R = 4$.

	PID	MRAC w/ Linear	MRAC w/ RBF
Taking off:			
SSE (cm)	0.65 ± 0.13	0.23 ± 0.04	0.27 ± 0.05
MSE (cm)	I	0.79 ± 0.21	0.32 ± 0.25
Rise time (sec)	22.40 ± 0.96	4.53 ± 0.85	2.99 ± 0.68
Overshoot (%)	I	4.55 ± 0.92	5.02 ± 0.73
Settling time (sec)	I	6.95 ± 1.23	5.00 ± 1.08
Landing:			
SSE (cm)	1.03 ± 0.19	0.10 ± 0.02	0.12 ± 0.03
MSE (cm)	I	1.10 ± 0.44	0.60 ± 0.35
Rise time (sec)	25.72 ± 1.03	4.99 ± 0.59	3.86 ± 0.62
Overshoot (%)	I	I	I
Settling time (sec)	I	7.40 ± 1.43	5.40 ± 1.12

Table 2.7: Performance index for $Z/R = 3$.

	PID	MRAC w/ Linear	MRAC w/ RBF
Taking off:			
SSE (cm)	0.40 ± 0.11	0.19 ± 0.05	0.18 ± 0.04
MSE (cm)	|	0.43 ± 0.17	0.18 ± 0.10
Rise time (sec)	25.86 ± 1.79	4.58 ± 0.97	3.15 ± 0.88
Overshoot (%)	|	7.58 ± 0.98	7.02 ± 1.01
Settling time (sec)	|	19.91 ± 1.95	19.18 ± 1.34
Landing:			
SSE (cm)	0.63 ± 0.11	0.10 ± 0.02	0.09 ± 0.02
MSE (cm)	|	0.44 ± 0.10	0.23 ± 0.05
Rise time (sec)	27.62 ± 1.44	5.23 ± 0.86	3.53 ± 0.90
Overshoot (%)	|	|	|
Settling time (sec)	|	11.09 ± 1.83	10.25 ± 1.19

Table 2.8: Performance index for $Z/R = 2$.

	PID	MRAC w/ Linear	MRAC w/ RBF
Taking off:			
SSE (cm)	0.26 ± 0.04	0.15 ± 0.01	0.23 ± 0.03
MSE (cm)	I	0.16 ± 0.03	0.09 ± 0.02
Rise time (sec)	31.07 ± 1.99	4.90 ± 1.07	3.50 ± 0.93
Overshoot (%)	I	11.40 ± 1.11	18.62 ± 1.56
Settling time (sec)	I	19.23 ± 0.99	19.94 ± 1.72
Landing:			
SSE (cm)	0.57 ± 0.08	0.09 ± 0.01	0.12 ± 0.02
MSE (cm)	I	0.18 ± 0.05	0.08 ± 0.02
Rise time (sec)	I	4.11 ± 0.76	3.51 ± 0.97
Overshoot (%)	I	I	I
Settling time (sec)	I	13.17 ± 1.30	13.88 ± 1.24

2.8 Summary

In this chapter, the model of ground effect on mini quadcopters is studied first. Different from the single rotor model which indicates a quadratic relation, the experimental result shows that the thrust generated by rotors to hover in the sky has a linear relation with the distance from the ground surface. The thrust will increase as the vehicle gets closer to the ground. Also, the model will switch from the linear relation to a quadratic function when the separation distance between rotors is large enough and the transition was observed from the experiment. A model reference adaptive controller is then developed and used to mitigate the ground effect in the control architecture. The control architecture consists of an LQR feedback + feedforward attitude controller, and a PID + MRAC position controller. The MRAC is added to the altitude loop to improve the altitude tracking performance. When designing the MRAC, two different uncertainty models are considered to approximate the ground effect function. One is a linear model and the other one is a set of RBFs. The performance of the controller is evaluated through a series of flight tests on a mini quadcopter and compared to a pure PID position controller. Based on the experiment result, the adaptive controller outperforms a traditional PID controller which is not able to either reach the target altitude during take-off or touchdown during landing in time. The vehicle with MRAC implemented can track the altitude command well and reduce the rise time by at least 80% of a pure PID controller. Furthermore, the result shows that the MRAC with RBFs has a better performance than the MRAC with a linear model as it has a 45% smaller mean square error and a 15% shorter rise time.

Chapter 3

System Identification of Wind Effects on Multirotor UAV Dynamics

Multirotor airplanes are widely used in many outdoor applications, e.g., agriculture, transportation, and public safety, where winds might be strong and prevalent. However, the effects of wind on multirotor aircraft are still not fully understood yet. The objective of this chapter is to investigate and model wind effects on a real hovering octocopter. The wind is directly measured and considered as one of the inputs to the bare-airframe model. Then a series of models, each corresponding to a different wind condition, are identified from real flight data through a system identification approach. The time-domain validation results show that an average of 15% error reduction can be achieved by considering wind effects, captured by a wind correction term. The identified models will play an important role in the future development of model-based controllers for outdoor multirotor aircraft.

3.1 Introduction

Multirotor UAVs are becoming increasingly popular over the past few years due to their high maneuverability and vertical takeoff and landing capability. They have been widely used in many outdoor applications, such as aerial photography, infrastructure inspection, agriculture monitoring, transportation, public safety, and delivery services [9, 6]. When flying in outdoor environments, such multirotor airplanes may be subject to the influences of various environmental disturbances, particularly winds. A strong wind may cause dramatic changes in the forces and moments acting on a multirotor UAV, which, if not taken care of properly, can significantly degrade the overall UAV flight performance. Therefore, the effects of wind on multirotor UAVs need to be formally investigated and modeled, which is the objective of this work.

In this work, a system identification approach is used to investigate and model the wind effects on an octocopter platform based on data collected from real flight experiments. Wind speeds and orientations are directly measured from a high-precision wind sensor and treated as an additional input entering the bare-airframe model. A series of linearized state-space models, each corresponding to a different wind condition, are identified in the frequency domain with Comprehensive Identification from Frequency Response (CIFER®) software. In addition, the eigenvalues and modes of the identified models are calculated and compared to evaluate the wind effects on aircraft dynamics. Finally, the identified models are validated with real flight doublet tests. The major contributions of this work are as follows: 1) this work is the first instance where the system identification approach has been used to model the wind effects on multirotor airplanes. The wind is directly measured and considered as one of the inputs to the model and 2) similarly, it is believed that this work is also the first instance where the effects of wind on multirotor airplanes, particularly with respect to different wind speeds, have been studied and validated from outdoor experiments with real wind data. Analysis and discussion on how the wind speeds affect the damping ratio and the natural frequency of the pitch mode of the bare-airframe model are provided. Such kind of insights will play an important role in the future development of model-based controllers for outdoor multirotor aircraft.

3.2 Related Work

Existing approaches to analyze and model aerodynamic effects, including wind effects, on multirotor aircraft can be roughly classified into four categories. The first category is through principle-based methods. The blade element theory and the momentum theory are widely used to derive the forces and the moments generated at the rotor plane of a multirotor airplane [14, 15, 16]. The blade flapping effect, which is significant in helicopters, has also be studied and adapted for multirotor aircraft [40, 99, 41]. These first-principle approaches all make assumptions that are mostly derived from single-rotor aircraft. Moreover, these approaches regularly neglect the inferences between different rotors, as well as between the rotors and the vehicle airframe. Flight experiments are required to validate such assumptions and subsequently the derived models. Recent work in [100] presents a way to model the downwash effect during proximity flight of two quadrotors by using the velocity field model and the result is validated in flight experiments.

Computational fluid dynamics (CFD) simulations is another option to study this complex aerodynamic effect. High-fidelity CFD simulations have been carried out to investigate aerodynamic interactions on a small quadrotor in [17]. However, such simulations are computationally expensive and cannot directly produce an analytic model, necessary for the purpose of either dynamic characteristics analysis or controller design.

The third option is by performing wind tunnel experiments. NASA Ames has conducted several wind tunnel experiments to determine the forces and the moments of a range of commercial multirotor airplanes under different wind speeds and attitudes [18]. However, the cost of such wind tunnel experiments is very high. Furthermore, at least with the wind tunnel used in [18], only static forces and moments can be measured at fixed conditions. Dynamic characteristics still remain indeterminable from such wind tunnel experiments and free flight tests are required to obtain the actual dynamic aircraft model.

Lastly, many researchers are using the system identification approach to solve the problem. System identification has been used successfully in the modeling of full-scale fixed-wing and rotary-wing airplanes [19]. The to-be-identified airplane needs to execute designated maneuvers to enable effective excitation on its dynamics. Then the airplane's model parameters can be estimated by analyzing the gathered flight data in either a time domain or a frequency domain [20, 21, 22, 23]. This approach has also been applied to the modeling of small UAVs [24, 25, 26], which usually have unique configurations and the aerodynamic interactions between their various components can be very complicated. When evaluating the turbulence effects on fixed-wing airplanes, the Dryden spectral model, which assumes a "frozen-field", is normally used [38]. However, the Dryden model is not suitable for a low-speed rotorcraft as the mean wind speed becomes dominant [39]. The authors in [42] proposed a method called control equivalent turbulence inputs (CETI) to model the turbulence on a low-speed helicopter

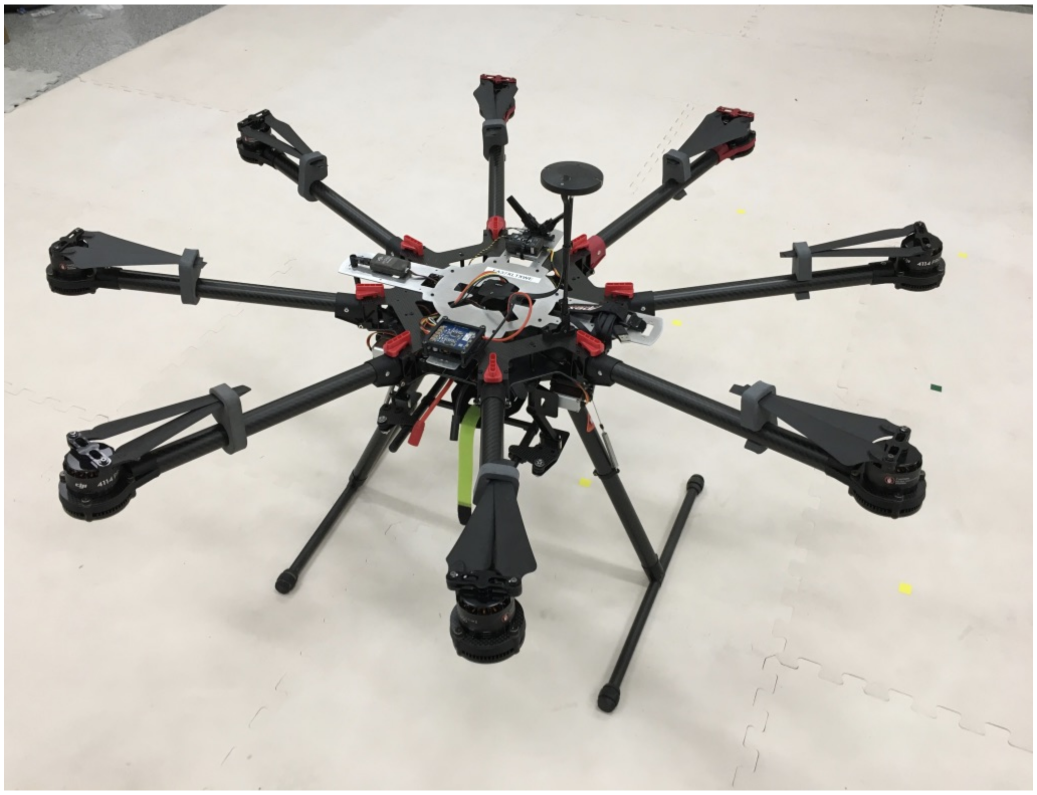


Figure 3.1: The octocopter platform used for the flight experiment.

and treated the turbulence as an equivalent control input. This approach has been proven to capture the effects caused by the turbulence on a UH-60 helicopter in [43] and a 3DR Iris+ quadrotor in [44]. Nevertheless, the wind speeds in these cases were not measured directly but inferred. More importantly, they assumed the mean wind is constant temporally and therefore didn't consider the effects of wind changes on the rotorcraft dynamics.

3.3 Flight Test Data Collection

This chapter introduces the UAV platform as well as the wind sensor used in flight experiments. A working hypothesis regarding the wind distribution is proposed which provides the necessary rationale for the experiment design. It also describes the flight experiment design and how the data on both the UAV and the wind are collected.

3.3.1 Hardware Description

A photo of the UAV platform is provided in Fig. 3.1. It is a retrofitted octocopter, of which the main body is built from a DJI Spreading Wings S1000+. The Pixhawk Cube is selected as the flight controller due to its fully programmable capability. A FrySky X8R receiver and a Taranis X9D Plus transmitter are used as the RC pair. A Here+ RTK GPS is used to provide precise position and velocity estimation of the UAV and a telemetry radio is installed to monitor the real-time flight status. Each propeller of the octocopter has a diameter of 38.1 cm and a maximum thrust of 2.8 kg. The octocopter has a motor-to-motor distance of 104.5 cm and a takeoff weight of 7.0 kg. A 16,000 mAh 6S lithium polymer battery is used, which can provide a flight duration of around 20 minutes.

Model 91000 ResponseONE ultrasonic anemometer is selected as the wind sensor to measure horizontal winds in the field. The anemometer has an accuracy of ±0.3 m/s in speed measurement and ±2 degrees in direction measurement. The sensor is mounted on top of a telescopic antenna mast and has a distance of 4.6 m above the ground. Since winds in the vertical direction are less significant than those in the horizontal direction, the vertical winds are ignored in the flight experiments. Fig. 3.2 shows the setup of the outdoor flight experiments.

3.3.2 Wind Distribution Hypothesis

The winds ideally should be measured at the exact location of the UAV to precisely study the its effect on the UAV dynamic. However, this can not be achieved easily in practice due to the limited payload and space on the aircraft. Therefore, one hypothesis is made as follows:

Hypothesis 3 (Working hypothesis) *After considering transport delay, the wind at the location of the UAV is roughly the same as the wind at the location of the wind sensor, as long as the UAV and the wind sensor are at the same height and not far away from each other.*

Such a hypothesis is in accordance with Taylor's hypothesis, which states that an eddy, satisfying $\sigma_M < 0.5M$ with M being the wind speed and σ_M being the standard deviation of the wind speed, has negligible change as it advects past a sensor [101]. In order to test the Hypothesis 3, two identical wind sensors were used to compare their wind measurements under two different scenarios, as shown in Fig. 3.3: a) the sensors are positioned along the wind direction and b) the sensors are positioned perpendicular to the wind direction. They are placed at the same height (4.6 m) above the ground. The wind speed and direction measurements from the two sensors are plotted in Fig. 3.4. A transport delay has been considered in scenario a) to

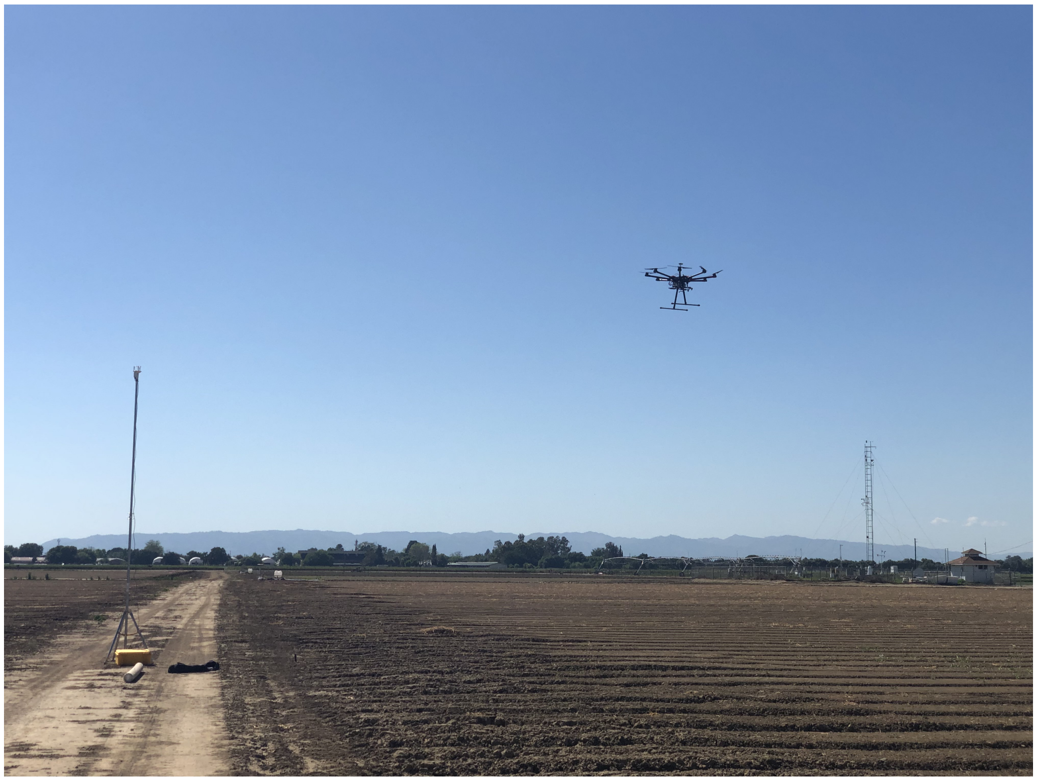


Figure 3.2: Outdoor flight experiment setup.

Table 3.1: The RMSEs of two sensor readings under different scenarios.

	Along wind direction	Perpendicular to wind direction
RMSE of wind speed	0.1558 m/s	0.2223 m/s
RMSE of wind direction	1.7190 deg	2.2723 deg

account for the wind traveling time. The root-mean-square errors (RMSEs) between the two wind sensor readings are calculated for both scenarios and the results are provided in Table 3.1. It can be observed that the RMSEs are very small compared to the mean wind speed and the mean direction. It also reflects the fact that winds change more rapidly temporally than spatially [101]. Thus the Hypothesis 3 is tested. Besides, it's observed that the difference along the wind direction is smaller than the one perpendicular to the wind direction.

Based on the hypothesis and the observations, the UAV is commended to hover at a position a distance of 4.6 m away from a single wind sensor and along the wind direction during the flight experiments. The measured wind speed and direction by the wind sensor will be regarded

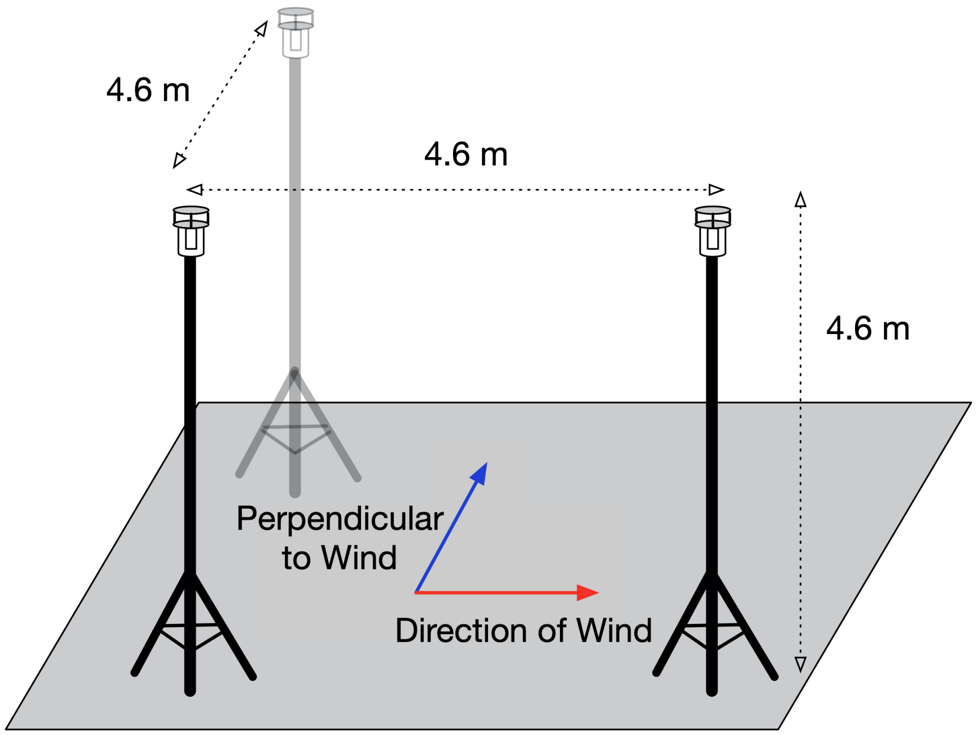
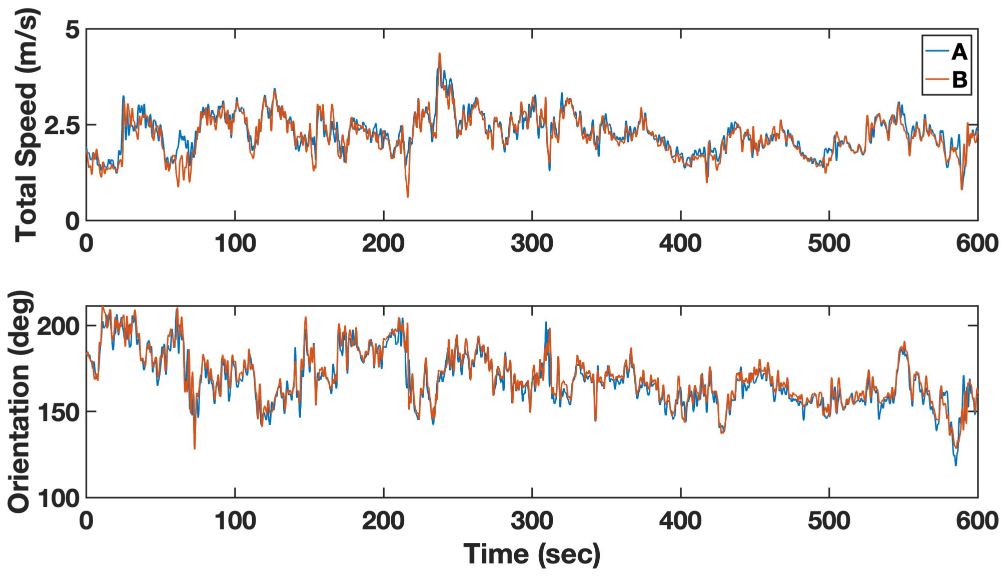


Figure 3.3: Schematic illustration of the two wind sensor relative positions of which the sensors are positioned a) along the wind direction and b) perpendicular to the wind direction.

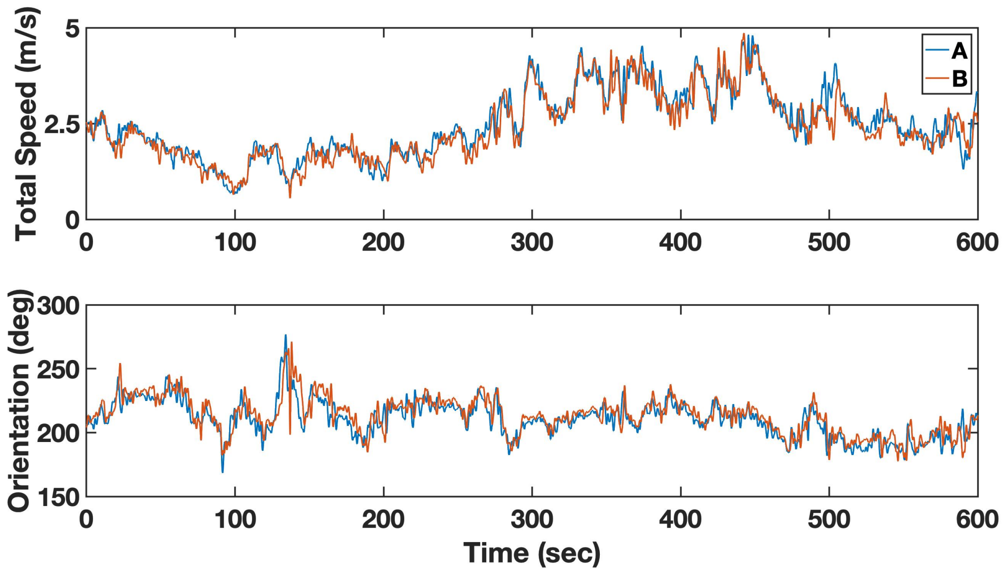
as wind data at the location of the UAV after taking a transport delay into consideration. The wind sensor is placed upstream of the UAV so that the downwash from UAV propellers does not affect the wind measurements. Moreover, the UAV is commanded to fly at the same height as the wind sensor therefore the vertical distribution of the wind can be ignored.

3.3.3 Flight Test Methodology

The goal of the modeling effort is to identify the bare-airframe model under the effects of wind. The bare-airframe model is a model consisting of the mixer, actuators, and the vehicle, as shown in Fig. 3.5. The mixer generates the command for each motor based on the controller output. The commands are then executed by actuators which include ESCs, motors, and rotors, resulting in different forces and torques and eventually changing the state of the vehicle. The bare airframe is known to be unstable, therefore a control system must be enabled during the flight. The pilot command δ_{pilot} and the computer-generated command δ_{auto} are injected into the control system, resulting in four inputs to the mixer: 1) δ_{lat}, the roll control, 2) δ_{lon}, the pitch control, 3) δ_{yaw}, the yaw control, and 4) δ_{thr}, the throttle control. The controller script here is



(a)



(b)

Figure 3.4: Example readings of the two wind sensors a) along the wind direction and b) perpendicular to the wind direction.

customized to provide position feedback keeping the UAV roughly a 4.6 m clearance above the ground and to ensure that the ground effect does not contribute to the identified model [102]. The measured wind as described in Section 3.3.2 is treated as an additional control input, 5) δ_{wind}, into the bare-airframe model.

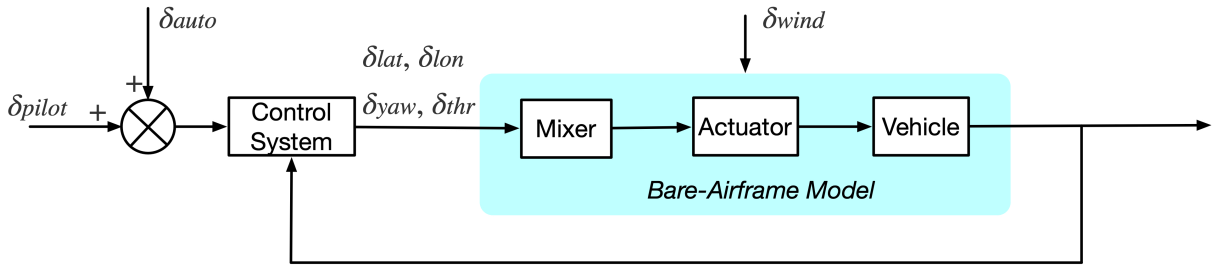


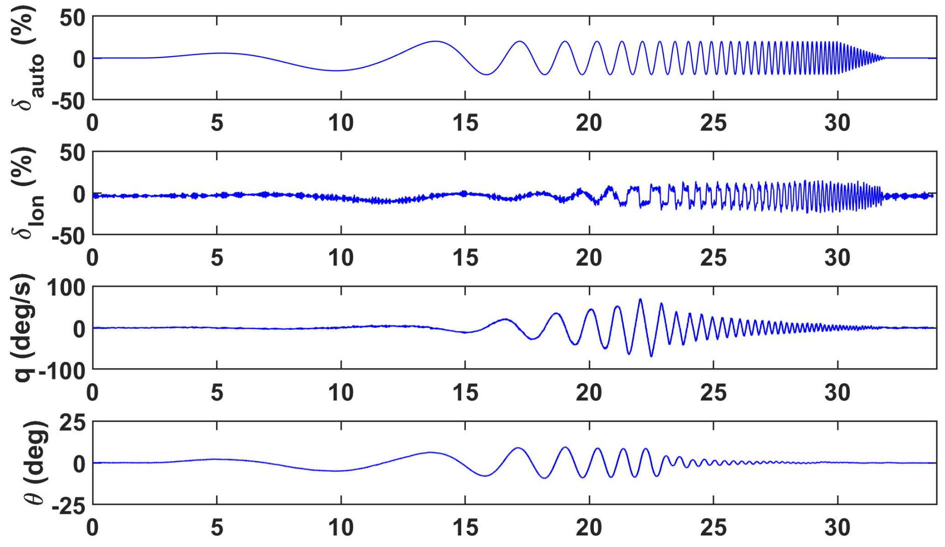
Figure 3.5: Schematic diagram of the bare-airframe model identification framework.

The onboard inertial measurement unit (IMU) measures the translational accelerations and the angular rates of the UAV, while the RTK GPS and the magnetic compass measure the global position and the heading angle, respectively. All these measurements pass through an extended Kalman filter, resulting in the following set of estimated states: 1) the roll, pitch, yaw Euler angles $\{\phi,\ \theta,\ \psi\}$, 2) the roll, pitch, yaw angular velocities $\{p,\ q,\ r\}$ in the body frame, 3) the linear velocities $\{v_n,\ v_e,\ v_d\}$ in the inertial frame where the subscripts $n,\ e,\ d$ represent for the north, east, downward directions, and 4) the linear accelerations $\{a_x,\ a_y,\ a_z\}$ in the body frame where the subscripts $x,\ y,\ z$ align with the vehicle's body axes. All these states, together with the mixer inputs $\{\delta_{lat},\ \delta_{lon},\ \delta_{yaw},\ \delta_{thr}\}$ which are the magnitudes of latitude, longitude, yaw, and throttle commands normalized to 0~100%, were logged at 100 Hz. The horizontal wind speeds $(v_n^w,\ v_e^w)$ in the north and east directions were recorded at 20 Hz by the wind sensor, where the superscript w represents the variable related to wind. They were re-sampled to 100 Hz through the fourth-order spline interpolation in order to be consistent with the estimated states sampling rate. Moreover, the airspeed in the body frame can be calculated with

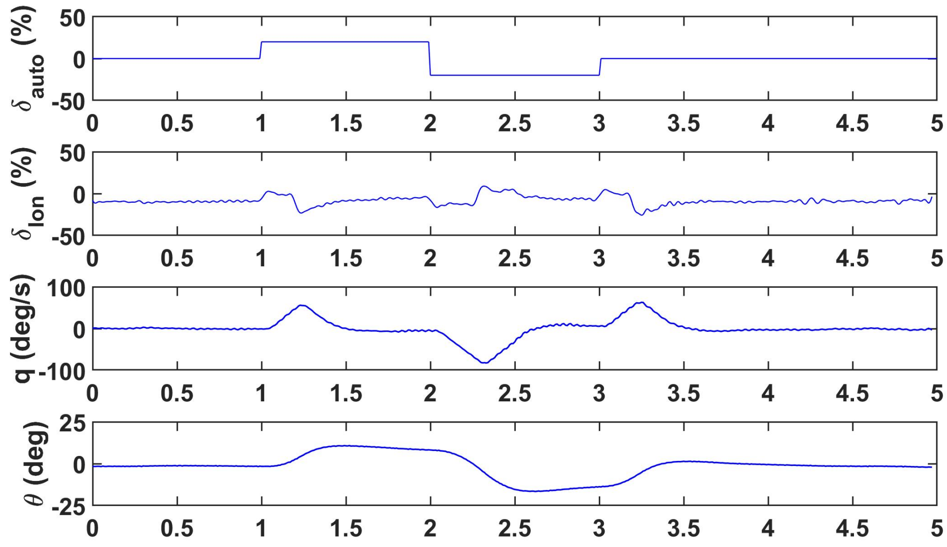
$$\mathbf{V_b} = \mathbf{R_i^b}(\mathbf{V_i} - \mathbf{V_{i,\ wind}}) \tag{3.1}$$

Where $\mathbf{V_b} = [\mathrm{u},\ \mathrm{v},\ \mathrm{w}]^T$ is the body translational velocity vector and $\{u,\ v,\ w\}$ are three velocities aligning with the vehicle's body axes. $\mathbf{V_i} = [v_n,\ v_e,\ v_d]^T$ and $\mathbf{V_{i,\ wind}} = [v_n^w,\ v_e^w,\ 0]^T$ are the inertial velocity and the inertial wind speed vectors, respectively. $\mathbf{R_i^b}$ is the rotation matrix from the inertial frame to the body frame.

Two different types of computer-generated commands (signals) were applied in the flight experiments. The first one is a frequency sweep signal for the frequency-domain identification purpose. A frequency sweep signal spans a frequency range of 0.1-10 Hz with a fade-in added



(a)



(b)

Figure 3.6: Example longitudinal axis flight data with (a) frequency sweep and (b) doublet signals under the calm wind condition.

in the low-frequency portion to avoid sharp inputs to the motors [19]. The second one is a doublet signal for the time-domain model validation purpose. A doublet signal is a two-sided pulse with the same magnitude in both the positive and the negative directions. Two sets of example flight data in the longitudinal axis, one with frequency sweep and the other one with doublet signals injected, are shown in Fig. 3.6.

Each data collection session is defined as a trial and each trial corresponds to one particular signal (frequency-sweep or doublet), one particular axis (longitudinal, lateral, directional, or throttle), and one particular wind condition (calm, light, and strong). During the data collection, each trial started and ended at roughly the same trim condition and lasted 34 seconds for the frequency-sweep signals and 5 seconds for the doublet signals. The duration is carefully selected to capture enough spectral contents and ensure the vehicle's attitude controls do not degrade as the battery weakens [103]). Before each trial, the magnitudes of the to-be-injected signal, whether frequency-sweep or doublet, was determined to sufficiently and effectively excite the multirotor UAV dynamics, e.g., ensuring a good signal-to-noise ratio.

5 trials have been conducted for each signal-axis-wind combination. Therefore, the total trial number is $5 \times 2 \times 4 \times 3 = 120$. To capture the effects of wind on UAV dynamics in a wider range, the experiments were conducted in three days with different wind conditions: a calm wind day (with an average wind speed of 0.9 m/s), a light wind day (with an average wind speed of 2.7 m/s) and a strong wind day (with an average wind speed of 5.4 m/s) and a total of $5 \times 2 \times 4 = 40$ trials on each day. Use the Beaufort Scale [104] to categorize the wind conditions, the three wind speeds fall into Force 0, Force 2, and Force 4, respectively. It is noted that the proposed method can be extended to different wind conditions. The three wind conditions surveyed here are the common wind speeds in the testing field and higher wind speeds are possible but could rarely be seen during the data collection season, so they were not included in the experiments.

3.4 System Identification Methodology

The general procedure prescribed in the book [19] are followed to perform the comprehensive identification of the UAV dynamics and wind effects based on the data collected from the flight experiments. It involves collecting frequency sweep data, identifying a state model with conditioned responses in the frequency domain, and verifying the model in the time domain. This system identification methodology is used to identify three state-space models under three wind conditions as described in the previous chapter.

3.4.1 Model Structure

The six-DOF flight-dynamics equations of motion [105] are used to derive the state-space model of the bare airframe. In addition, a first-order actuator model is explicitly considered to capture the high-frequency roll-off in angular rate frequency responses:

$$\frac{\delta'(s)}{\delta(s)} = \frac{1}{T_a s + 1} \tag{3.2}$$

where δ is the original control signal, δ' is the lagged control signal after considering the first-order dynamics, and T_a is the time constant. To reflect such effect of actuator dynamics, three additional states δ'_{lat}, δ'_{lon}, and δ'_{thr} are added to the original rigid body state vector, resulting the final state vector of the state-space model being

$$\mathbf{x} = [u,\ v,\ w,\ p,\ q,\ r,\ \phi,\ \theta,\ \psi,\ \delta'_{lat},\ \delta'_{lon},\ \delta'_{thr}]^T \tag{3.3}$$

The input (control) vector to the state-space model includes two parts. The first part is the control signals $\mathbf{u} = [\delta_{lat},\ \delta_{lon},\ \delta_{yaw},\ \delta_{thr}]^T$ from the flight control system, and the second part is the wind, modeled as an additional input, $\mathbf{u_w} = [\delta^w_{lat},\ \delta^w_{lon},\ \delta^w_{thr}]^T$. Moreover, the time delay in each control axis is explicitly included in the state-space model to account for unmodeled high-frequency dynamics. The final state-space model has the formula of:

$$\begin{aligned} \mathbf{M\dot{x}} &= \mathbf{Fx} + \mathbf{G_u u}(t-\tau_\mathbf{u}) + \mathbf{G_w u_w}(t-\tau_\mathbf{w}) \\ \mathbf{y} &= \mathbf{H_0 x} + \mathbf{H_1 \dot{x}} \end{aligned} \tag{3.4}$$

where $\mathbf{M}$ is an identity matrix and $\mathbf{F}$ is the state derivative matrix as follows:

$$\mathbf{F} = \begin{bmatrix} X_u & X_v & X_w & X_p & X_q - W_0 & X_r + V_0 & 0 & -g\cos\theta_0 & 0 & X_{lat} & X_{lon} & X_{thr} \\ Y_u & Y_v & Y_w & Y_p + W_0 & Y_q & Y_r - U_0 & g\cos\phi_0\cos\theta_0 & -g\sin\phi_0\sin\theta_0 & 0 & Y_{lat} & Y_{lon} & Y_{thr} \\ Z_u & Z_v & Z_w & Z_p - V_0 & Z_q + U_0 & Z_r & -g\sin\phi_0\cos\theta_0 & -g\cos\phi_0\sin\theta_0 & 0 & Z_{lat} & Z_{lon} & Z_{thr} \\ L_u & L_v & L_w & L_p & L_q & L_r & 0 & 0 & 0 & L_{lat} & L_{lon} & L_{thr} \\ M_u & M_v & M_w & M_p & M_q & M_r & 0 & 0 & 0 & M_{lat} & M_{lon} & M_{thr} \\ N_u & N_v & N_w & N_p & N_q & N_r & 0 & 0 & 0 & N_{lat} & N_{lon} & N_{thr} \\ 0 & 0 & 0 & 1 & \sin\phi_0\tan\theta_0 & \cos\phi_0\tan\theta_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos\phi_0 & -\sin\phi_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sin\phi_0\sec\theta_0 & \cos\phi_0\sec\theta_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1/T_a & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1/T_a & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1/T_a \end{bmatrix} \tag{3.5}$$

with $[U_0,\ V_0,\ W_0,\ \phi_0,\ \theta_0]^T$ being the trim conditions, $\mathbf{G_u}$ and $\mathbf{G_w}$ are the control derivative matrices with respect to the controller and the wind, respectively, as follows:

$$\mathbf{G_u} = \begin{bmatrix} 0 & 0 & X_{yaw} & 0 \\ 0 & 0 & Y_{yaw} & 0 \\ 0 & 0 & Z_{yaw} & 0 \\ 0 & 0 & L_{yaw} & 0 \\ 0 & 0 & M_{yaw} & 0 \\ 0 & 0 & N_{yaw} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1/T_a & 0 & 0 & 0 \\ 0 & 1/T_a & 0 & 0 \\ 0 & 0 & 0 & 1/T_a \end{bmatrix}, \qquad \mathbf{G_w} = \begin{bmatrix} X_{w,lat} & X_{w,lon} & X_{w,thr} \\ Y_{w,lat} & Y_{w,lon} & Y_{w,thr} \\ Z_{w,lat} & Z_{w,lon} & Z_{w,thr} \\ L_{w,lat} & L_{w,lon} & L_{w,thr} \\ M_{w,lat} & M_{w,lon} & M_{w,thr} \\ N_{w,lat} & N_{w,lon} & N_{w,thr} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.6}$$

where $\tau_{\mathbf{u}} = [\tau_{lat},\ \tau_{lon},\ \tau_{yaw},\ \tau_{thr}]^T$ and $\tau_{\mathbf{w}} = [\tau_{lat}^w,\ \tau_{lon}^w,\ \tau_{thr}^w]^T$ are the time delays corresponding to the control input and the wind, respectively. $\mathbf{y} = [u,\ v,\ w,\ p,\ q,\ r,\ a_x,\ a_y,\ a_z]^T$ is the observed output, and $\mathbf{H_0}$ and $\mathbf{H_1}$ are the measurement matrices, allowing the measurement vector $\mathbf{y}$ to be expressed in terms of the state vector $\mathbf{x}$ and its derivative $\dot{\mathbf{x}}$. The $\mathbf{H_0}$ and $\mathbf{H_1}$ can be derived using the following two observations: 1) the body translational and angular velocities can be directly found in the state vector, and 2) the accelerometer measurements are of the form:

$$\begin{aligned} a_x &= \dot{u} + W_0 q - V_0 r + (g\cos\theta_0)\theta \\ a_y &= \dot{v} + U_0 r - W_0 p - (g\cos\theta_0)\phi \\ a_z &= \dot{w} + V_0 p - U_0 q + (g\sin\theta_0)\theta \end{aligned} \tag{3.7}$$

A detailed derivation is provided in Appendix B. The state-space model structure in Eqn. (3.4) can be further reduced by analyzing the coherence of each input and output pair from the flight experiment data. A practical reduction procedure is implemented by following the guideline specified in [19],

$$\begin{aligned} \hat{\gamma}_{xy}^2 &\geq 0.5 \\ log(\omega_{max}/\omega_{min}) &\geq 0.3 \end{aligned} \tag{3.8}$$

where $\hat{\gamma}_{xy}^2$ is the coherence value between an input x (an element of $\mathbf{u}$) and an output y (an element of $\mathbf{y}$), and ω_{min} and ω_{max} are the minimum and maximum frequencies of the fitting range, respectively. Only those responses that meet both requirements will be used in the model parameter identification step.

3.4.2 Model Parameter Identification

The parameters of the determined model in Eqn. (3.4) are estimated by minimizing the following cost function:

$$J = \sum_{l=1}^{n_{TF}} \left\{ \frac{20}{n_\omega} \sum_{\omega_1}^{\omega_{n_\omega}} W_\gamma \left[W_g (|\hat{T}_c| - |T|)^2 + W_p (\angle \hat{T}_c - \angle T)^2 \right] \right\}_l \tag{3.9}$$

where n_{TF} is the number of different frequency-response pairs, n_ω is the number of frequency points in each frequency range (ω_1, ω_{n_ω}), W_γ, W_g, and W_p are the weighting functions, and $\hat{T}_c$ and T are the frequency responses obtained from the flight data and the identification solution, respectively. The cost function is derived from minimizing the summed cost for the n_{TF} transfer functions with high coherency values and for each transfer function, the weighted sum of the square errors of both magnitude and phase are minimized at different frequency points. The optimization problem in Eqn. (3.9) is solved by the secant method [106] which has been proven robust for the model structures involving large numbers of identification parameters and frequency-response pairs [19]. A $CR \leq 20\%$ (Cramer-Rao Bound) and $I \leq 10\%$ (Insensitivity) are considered acceptable for each identified parameter. Finally, an overall average cost function that achieves $J_{ave} \leq 150$ is considered as achieving an acceptable level of accuracy.

It should be pointed out that, different from the computer-generated command δ_{auto} that is pre-determined to sufficiently excite the UAV dynamics, the wind in natural outdoor environments is uncontrollable and cannot be designed. In order to identify the wind effect ($\mathbf{G_w}$) separately from the flight control input ($\mathbf{G_u}$), a working hypothesis is made here that:

Hypothesis 4 (Working hypothesis) *The parameters of* $\mathbf{F}$ *and* $\mathbf{G_u}$ *matrices are decoupled from the parameters of* $\mathbf{G_w}$ *matrix.*

With this hypothesis, the effect of wind can be identified in two steps: 1) identifying the parameters in $\mathbf{F}$ and $\mathbf{G_u}$ matrices without considering the wind term $\mathbf{G_u}$, and 2) the identification of parameters in $\mathbf{G_w}$ matrix, by considering the residual after removing the effects due to the $\mathbf{F}$ and $\mathbf{G_u}$ matrices. Later in this chapter, it will turn out that this hypothesis offers an approximation that leads to a model not only captures the wind effects mathematically but also fits the flight data reasonably well.

Table 3.2: Trim states under different wind conditions.

	Calm Wind	Light Wind	Strong Wind
U_0	0 m/s	2.24 m/s	4.12 m/s
W_0	0 m/s	-0.08 m/s	-0.22 m/s
θ_0	0 deg	-2.35 deg	-4.36 deg

3.5 System Identification Results

Three bare-airframe models are identified at different wind conditions (i.e., calm, light, and strong) as mentioned in Section 3.3.3. The trim states are obtained by forcing the aircraft to hover in the wind for 30 seconds without any pilot input. These trim states are averaged and presented in Table 3.2. Since in all trials, the pilot commanded the UAV to fly in a direction roughly facing the wind, meaning that the wind is in the longitudinal axis of the vehicle, therefore both ϕ_0 and V_0 become zeros. Notice that a higher wind speed in the longitudinal direction increases both the trim pitch angle θ_0 and the trim forward speed U_0. Notice that the calm wind speed is relatively small compared to the other two, so all three trim values are assumed to be zeros under the calm wind condition. These trim values are used in Eqn. (3.5).

3.5.1 Calm Wind Results

The described system identification methods are performed on the data collected on the calm wind day first. All off-axis responses are found to have low coherence values and thus not used in the identification process. This also indicates that the UAV dynamic is well-decoupled. The on-axis angular rate and accelerometer frequency responses along the longitudinal axis are provided in Fig. 3.7. The high coherence value indicates a good linearity of the model. Moreover, a first-order actuator model in Eqn. (3.2) can successfully capture the roll-off at the high-frequency range for the angular rate response. The lateral response has a similar performance. The finalized lateral and longitudinal models are

$$\begin{aligned} \dot{v} &= Y_v v + g\theta + Y_{lat}\delta'_{lat}(t-\tau_{lat}) \\ \dot{p} &= L_v v + L_{lat}\delta'_{lat}(t-\tau_{lat}) \end{aligned} \tag{3.10}$$

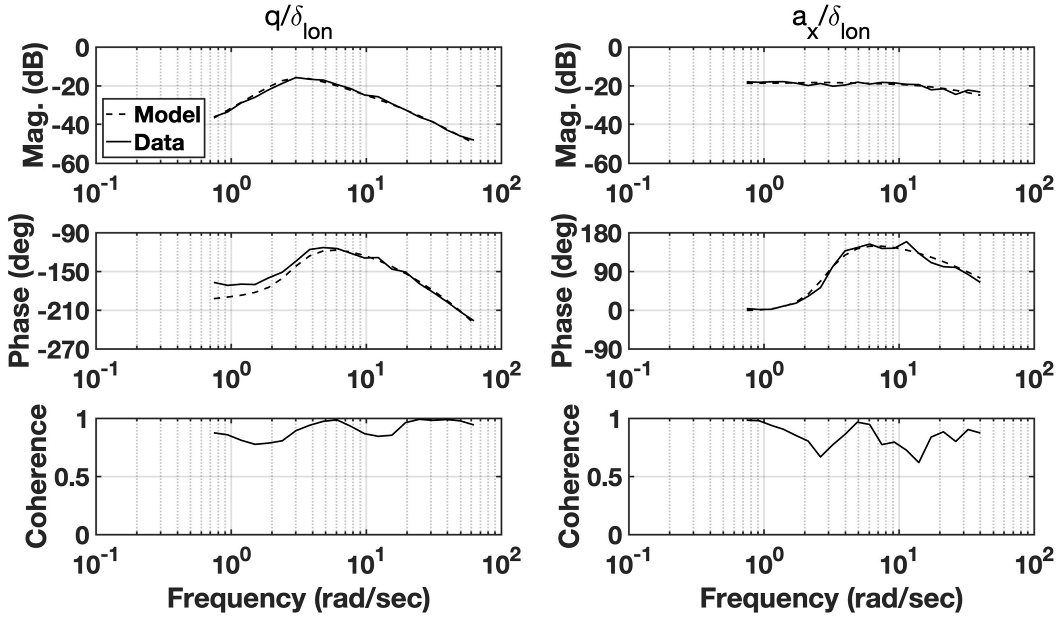


Figure 3.7: Transfer function responses along the longitudinal axis under the calm wind condition.

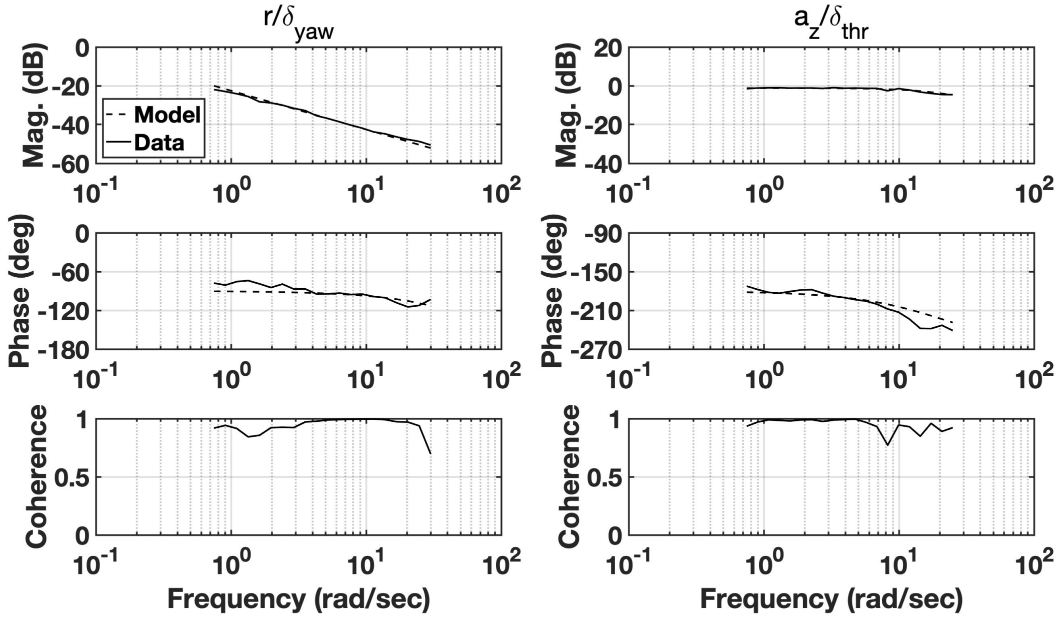


Figure 3.8: Transfer function responses along the yaw and heave axes under the calm wind condition.

and

$$\begin{aligned}\dot{u} &= X_u u - g\theta + X_{lon}\delta'_{lon}(t-\tau_{lon}) \\ \dot{q} &= M_u u + M_{lon}\delta'_{lon}(t-\tau_{lon})\end{aligned} \tag{3.11}$$

The frequency responses of the yaw and heave axes are provided in Fig. 3.8, which show that a first-order model is again adequate to capture the dynamics along each of the two axes. However, the yaw damping N_r and the heave damping Z_w both have high $CR\%$ values and thus are dropped in the finalized models. This indicates that the heave and the yaw rate dampings are not identifiable from the sweep signals currently injected into the bare airframe. Other excitations are needed to capture the value of these two terms. The equivalent time delay τ_{thr} is found to be unrelated and thus taken as zero, which means that the first-order actuator model is sufficient to capture the delay in the vertical axis. The finalized yaw and heave models are

$$\dot{r} = N_{yaw}\ \delta_{yaw}(t - \tau_{yaw}) \tag{3.12}$$

and

$$\dot{w} = Z_{thr}\ \delta'_{thr}(t) \tag{3.13}$$

Eqn. (3.10),(3.11),(3.12),(3.13) together complete the bare-airframe model under the calm wind condition. The identified parameters of these equations are provided in Table 3.3. From the result, the $CR\%$s and $I\%$s for all parameters are all within the guideline prescribed in [19]. Table 3.4 shows the cost function values for all frequency responses used in the identification. The average cost calculated according to Eqn. (3.9) is equal to 50.3972 which shows a high accuracy of the identified model for the calm wind condition.

3.5.2 Light and Strong Wind Results

The longitudinal frequency responses under light and strong wind conditions are provided in Fig. 3.9 and Fig. 3.10, respectively. It can be seen that the coherence in the low-frequency range is affected by the wind. Based on the hypothesis made in Section. 3.4.2, the parameters in $\mathbf{F}$ and $\mathbf{G_u}$ matrices are first determined by following the same procedure in the calm wind condition. The identified parameter values are provided in Table 3.3. It is observed that such a model, i.e., a model with only the $\mathbf{F}$ and $\mathbf{G_u}$ matrices, has a higher cost value than the one under the calm wind condition, indicating that the wind affects the longitudinal dynamic response. When the wind speed becomes higher, the yaw angular rate damping N_r becomes significant, therefore the N_r is added back to the strong wind model. Moreover, the wind is found to have a

Table 3.3: Identified state and control derivative values.

Parameter	Calm Wind			Light Wind			Strong Wind		
	Value	*CR%*	*I%*	Value	*CR%*	*I%*	Value	*CR%*	*I%*
X_u	-0.3172	6.258	1.919	-0.2423	6.056	1.571	-0.5008	15.18	4.727
M_u	0.7690	5.633	1.544	0.7099	5.414	1.324	1.267	17.29	3.444
X_{lon}	-0.0985	5.056	1.959	-0.0976	4.879	1.831	-0.0870	4.729	2.144
M_{lon}	0.5251	3.456	1.070	0.5413	3.141	0.9887	0.4523	3.011	1.308
Y_v	-0.2787	5.777	1.637	-0.3905	7.201	2.028	-0.2519	12.04	4.192
L_v	-0.7406	5.219	1.339	-0.7984	6.293	1.728	-0.7344	8.950	2.322
Y_{lat}	0.1185	5.196	1.873	0.1193	4.772	1.961	0.1194	5.081	2.300
L_{lat}	0.6226	3.385	0.9863	0.6472	2.984	1.096	0.6597	3.181	1.338
N_r	-	-	-	-	-	-	-0.2543	19.89	9.634
N_{yaw}	0.0744	4.795	2.175	0.0719	3.829	1.894	0.0631	4.213	2.107
Z_{thr}	-0.8712	3.824	1.881	-0.8777	3.764	1.882	-0.8531	3.796	1.898
T_a	0.0458	5.526	1.800	0.0458	-	-	0.0458	-	-
τ_{lat}	0.0194	8.514	3.805	0.0160	7.081	3.533	0.0216	6.608	3.201
τ_{lon}	0.0201	9.002	4.061	0.0220	10.30	5.140	0.0193	7.607	3.798
τ_{yaw}	0.0133	23.94	8.764	-	-	-	-	-	-
τ_{thr}	-	-	-	0.0205	15.79	9.90	0.0168	21.80	8.90

limited effect on the yaw and heave dynamics compared to ones in the latitude and longitudinal axes.

The residual is defined as the difference between the actual UAV response and the predicted response by only considering the **F** and $\mathbf{G_u}$ terms:

$$Residual = \mathbf{y} - \mathbf{H_0}\mathbf{x} - \mathbf{H_1}(\mathbf{F}\mathbf{x} + \mathbf{G_u}\mathbf{u}(t - \tau_{\mathbf{u}})) \tag{3.14}$$

which can be accounted by the $\mathbf{G_w}\mathbf{u_w}$ term in Eqn. (3.4). A one-step prediction is performed at each time step by using the partial model identified in Table. 3.3 (i.e., the model with the **F** and $\mathbf{G_u}$ matrices but not the $\mathbf{G_w}$ matrix) and the state vector recorded from flight experiments.

Table 3.4: Transfer function cost.

Response	Cost		
	Calm Wind	Light Wind	Strong Wind
v/δ_{lat}	80.94107	51.15984	89.13262
p/δ_{lat}	53.81091	29.39847	61.59804
a_y/δ_{lat}	54.45253	41.03815	109.67281
u/δ_{lon}	80.77869	71.62726	78.71678
w/δ_{lon}	-	61.98955	74.27301
q/δ_{lon}	41.51912	66.48181	51.47652
a_x/δ_{lon}	34.19383	43.78340	74.21771
r/δ_{yaw}	27.11832	74.95789	65.65873
a_z/δ_{thr}	30.32688	13.72616	41.08481
Average	50.3927	50.4625	71.7590

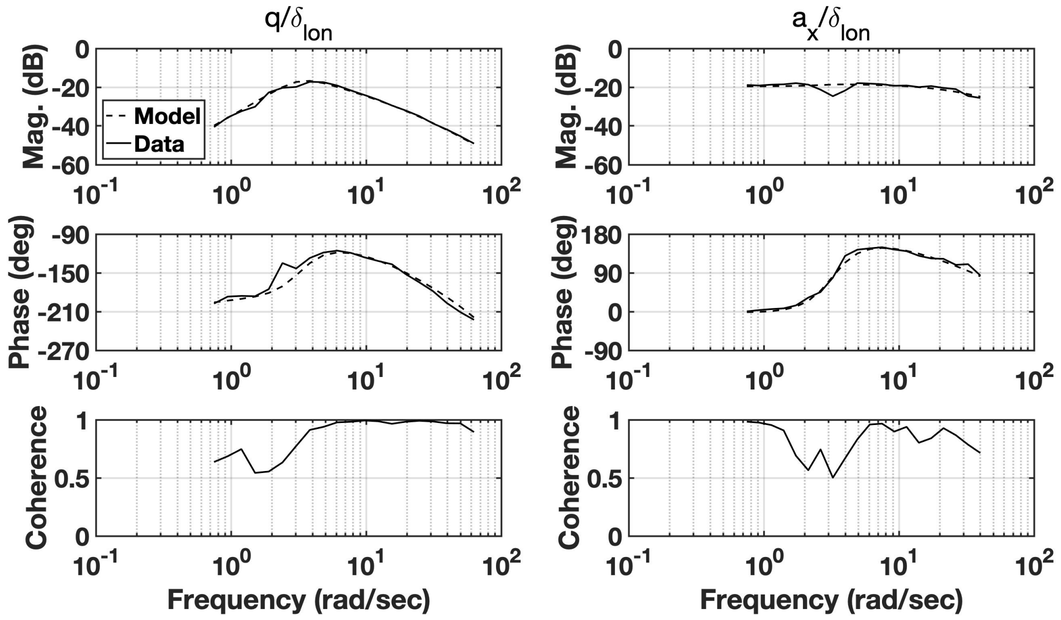


Figure 3.9: Transfer function responses in the longitudinal axis under the light wind condition.

Since the body translational velocities and the angular rates are directly measured, only the accelerometer measurements are left during the residual calculations. Sample residuals of a_x and a_z under light and strong wind conditions are plotted in Fig. 3.11 and 3.12, respectively. These

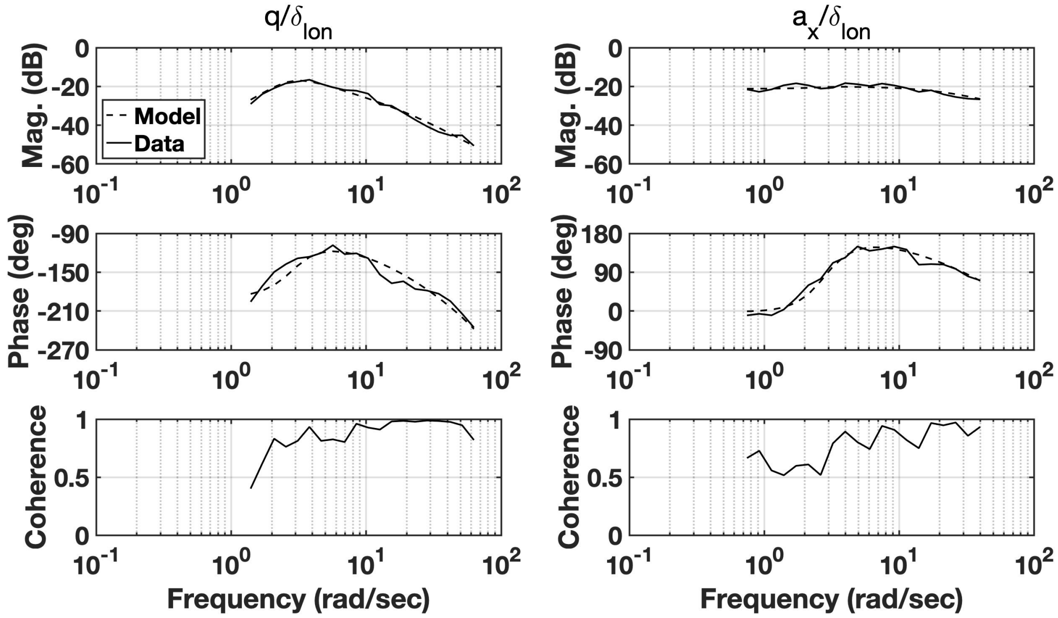


Figure 3.10: Transfer function responses in the longitudinal axis under the strong wind condition.

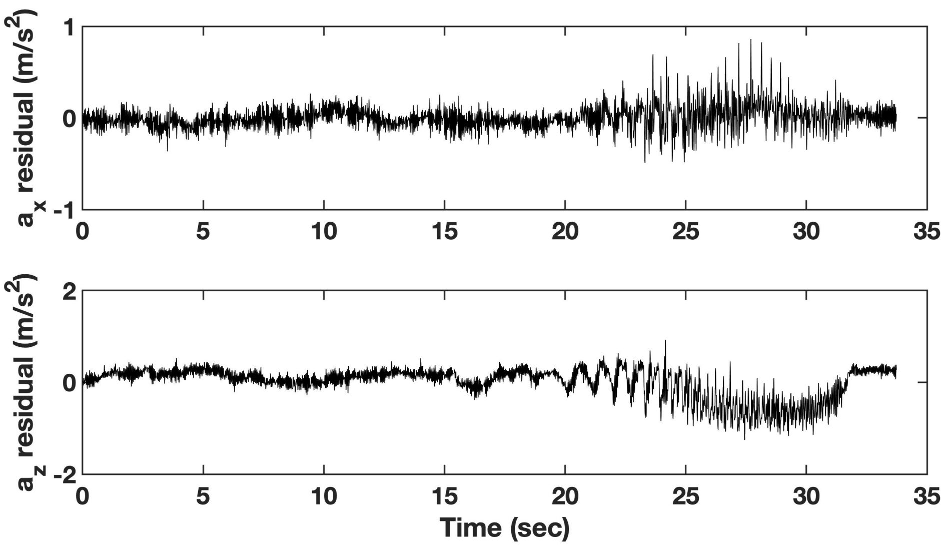


Figure 3.11: Residuals of a_x and a_z under the light wind condition.

residuals are then used to identify the parameters in the $\mathbf{G_w}$ matrix. Due to the symmetrical configuration of the UAV platform (which can be verified from the calm wind results), the wind is assumed to have the same effect along the longitudinal and the lateral axes. Therefore, there are four parameters need to be identified, which are $X_{w,lon}$, $Z_{w,lon}$, τ^w_{lon}, and τ^w_{thr}. The identified values are provided in Table 3.5.

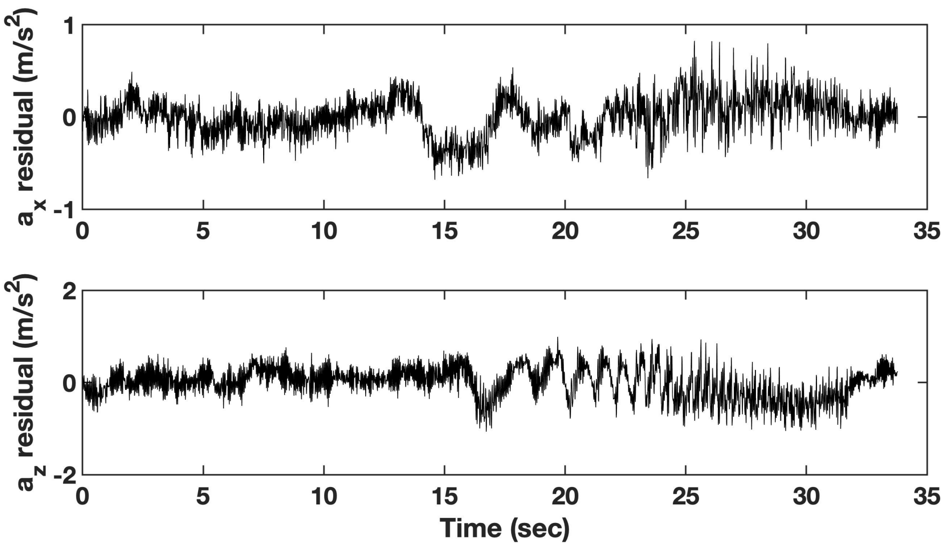


Figure 3.12: Residuals of a_x and a_z under the strong wind condition.

Table 3.5: Identified parameters in G_w matrix.

Parameter	Light Wind			Strong Wind		
	Value	*CR%*	*I%*	Value	*CR%*	*I%*
$X_{w,lon}$	-0.1486	4.222	2.111	-0.3560	10.28	5.667
$Z_{w,lon}$	0.8451	4.736	2.368	1.323	18.98	9.302
τ^w_{lon}	0.0178	3.262	1.508	0.0132	25.45	11.990
τ^w_{thr}	0.0386	7.677	1.980	0.0203	18.97	13.234

3.5.3 Time Domain Verification

The doublet response (the data collected for such response is not used in the system identification process but only for the verification purpose) for the model identified under the calm wind condition is provided in Fig. 3.13. It shows excellent agreement between the model output and the flight data.

The doublet responses for the two models identified under the light wind condition, one without the wind correction term (i.e., $\mathbf{G_w u_w}$) and the other one with the wind correction term are shown in Fig. 3.14. The similar doublet responses for the two models identified under the strong wind condition are shown in Fig.3.15. Furthermore, the RMSE values under different

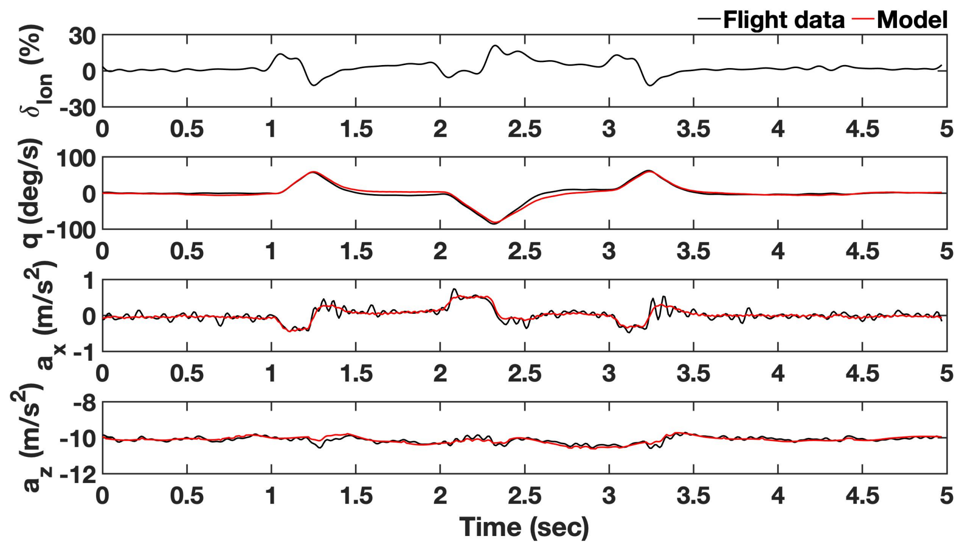


Figure 3.13: Time-domain validation under the calm wind condition.

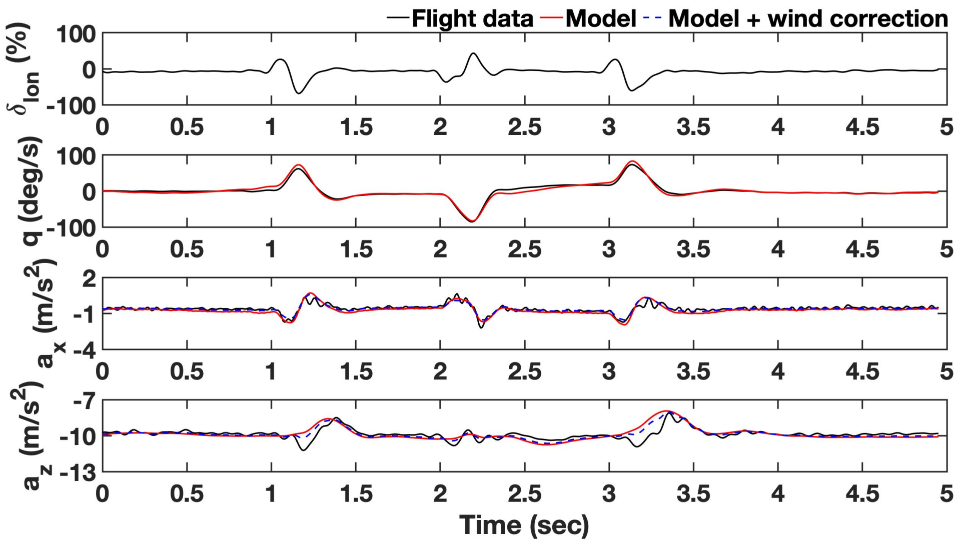


Figure 3.14: Time-domain validation under the light wind condition with and without the wind correction.

wind conditions using different models are given in Table 3.6. The validation result shows that when measuring and considering the wind correction term, $\mathbf{G_w u_w}$ into the state-space model, the RMSE of a_x can be reduced by 13% under the light wind and 16% under the strong wind condition. Meanwhile, the RMSE of a_z is reduced by 15% under the light wind and 7% under the strong wind. All these results show that the predictive power of the identified model is significantly improved by considering the wind effects explicitly.

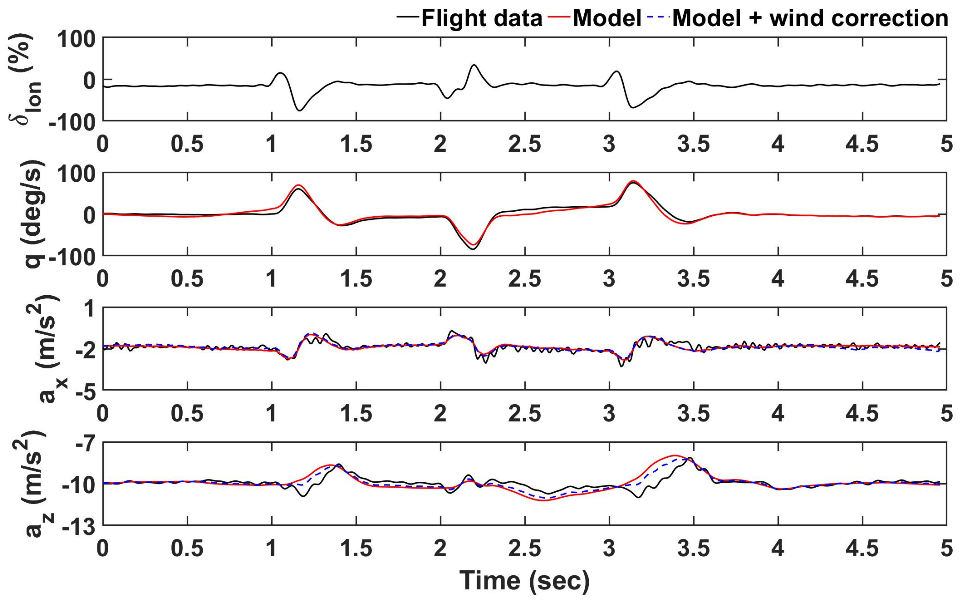


Figure 3.15: Time-domain validation under the strong wind condition with and without the wind correction.

Table 3.6: RMSEs under different wind conditions and with different models.

	$RMSE(a_x)$	$RMSE(a_z)$
calm wind, no wind correction	0.0624	0.1454
light wind, no wind correction	0.2757	0.4270
light wind, with wind correction	0.2399	0.3629
strong wind, no wind correction	0.4129	0.6248
strong wind, with wind correction	0.3480	0.5781

Table 3.7: Eigenvalues and modes under different wind conditions.

λ#	Calm Wind	Light Wind	Strong Wind	Mode Description
1-2	$1.3498 \pm 2.4937i$ (ζ=-0.4760, ω=2.8355)	$1.3640 \pm 2.5037i$ (ζ=-0.4784, ω=2.8511)	$1.3769 \pm 2.5698i$ (ζ=-0.4723, ω=2.9154)	unstable roll mode
3	-2.9783	-2.9798	-3.0134	damped roll mode
4-5	$1.3532 \pm 2.5202i$ (ζ=-0.4730, ω= 2.8605)	$1.5091 \pm 2.8193i$ (ζ=-0.4719, ω=3.1978)	$1.6551 \pm 2.9815i$ (ζ=-0.4854, ω=3.4101)	unstable pitch mode
6	-3.0235	-3.4688	-3.9398	damped pitch mode
7	-	-	-0.2543	damped yaw mode

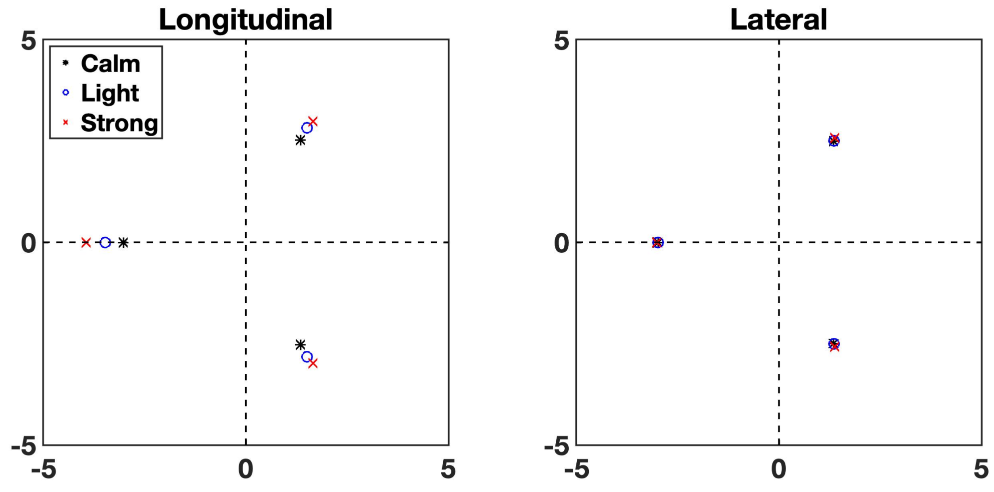


Figure 3.16: Plots of eigenvalues along the longitudinal and lateral axes under different wind conditions.

3.6 Eigenvalue Analysis

The eigenvalues of the bare-airframe models identified under different wind conditions are calculated. These eigenvalues, together with their corresponding dynamics modes, are shown in Table. 3.7. There are two unstable poles in both the lateral and the longitudinal dynamics, indicating that the bare-airframe multirotor airplane itself is inherently unstable, therefore demonstrating the necessity of a control system in the loop. Furthermore, it can be observed that the eigenvalues in the pitch and roll modes are very close to each other, which indicates the high symmetry of the UAV platform. The results regarding eigenvalues and dynamics modes show that when a multirotor UAV is hovering under a windy condition, either with a light or strong wind, its dynamics characteristics are significantly affected by the wind. The wind effects can be summarized as follows:

1. Both the damping ratio and the natural frequency of the longitudinal dynamics increase as the wind speed increases. For instance, the natural frequency is increased by 12% and 20% under light and strong wind conditions, respectively, as compared to that under the calm wind condition.

2. There exists a damped real pole in both the longitudinal and the lateral dynamics. Such a pole is moved further left as wind speed increases.

3. The time-to-double amplitude T_2 decreases as the wind speed increases. For this aircraft, T_2 changes from 0.51 second (under the calm wind) to 0.46 second (under the light wind) and 0.42 second (under the strong wind). This means that the bare airframe becomes more unstable as the wind speed increases and a more robust control system is required to stabilize the vehicle.

4. The eigenvalues in the roll mode remain unchanged with respect to the wind speed as long as the wind direction is aligned with the vehicle's longitudinal axis, indicating that the wind has a less significant effect on the dynamics in the lateral axis.

5. The wind has a limited effect on the yaw response of a multirotor UAV under hovering conditions. A damped yaw mode is only observed when the UAV is under strong wind conditions.

All these findings are helpful in the design of a control system for outdoor multirotor UAVs.

3.7 Discussion

The study in this chapter provides a natural starting point for the systematic understanding of wind effects on multirotor UAVs. The experimental results show that some aerodynamic forces and moments are directly affected by the wind and these effects can be partially accounted for by considering the wind as one of the control inputs entering the bare-airframe model. Such additional aerodynamic forces can arise from: 1) the drag force acting on the rotor plane, 2) the thrust changes generated by the rotors, 3) the interactions between the rotors, and 4) the interactions between the rotors and the body. In order to obtain a more comprehensive understanding of the wind effects on multirotor dynamics, a multi-pronged methodology should be adopted. First, further analysis can be conducted by combining first-principle models and the system identification results. For example, by explicitly calculating the blade flapping and the induced drag from existing literature [107] and comparing such first-principle-based results to flight experiment data, the discrepancies can be explored to improve existing models. The discovery of such discrepancies can also improve the understanding of the complex aerodynamic effects of wind on multirotor aircraft. Second, since the wind in natural environments is uncontrollable, forced oscillation wind tunnel experiments [108] can be conducted to allow systematically injecting winds with proper spectral content to sufficiently excite multirotor UAV dynamics. For instance, a frequency sweep wind can be generated by such type of wind tunnels to excite a multirotor UAV and then a similar approach as presented in this chapter can be used to identify the wind effects on multirotor UAV dynamics.

3.8 Summary

A system identification approach was performed in this chapter to identify the wind effect on multirotor UAV dynamics near a hovering condition. In the flight experiment, the wind was directly measured by a wind sensor and treated as one of the control inputs into the linearized state-space model. The validation result in the time domain shows that, by explicitly considering the wind effect, such a treatment can reduce the prediction error on the acceleration and angular velocities by 15% on average. It is believed that the identified model can offer us unique insights into the effects of wind on multirotor dynamics and the design and development of model-based flight control systems in the future.

Chapter 4

Vision-based Control of UAV in Simulated Riverine Environments Using Imitation Learning

There have been many researchers studying how to enable UAVs to navigate in complex and natural environments autonomously. In this chapter, an imitation learning framework is developed and used to train navigation policies for the UAV flying inside complex and GPS-denied riverine environments. The UAV relies on a forward-pointing camera to perform reactive maneuvers and navigate itself in 2D space by adapting the heading. The performances among a linear regression-based controller, an end-to-end neural network controller, and a variational autoencoder (VAE)-based controller trained with data aggregation method are compared in the simulation environments. The results show that the VAE-based controller outperforms the other two controllers in both training and testing environments and is able to navigate the UAV with a longer traveling distance and a lower intervention rate from the pilots.

4.1 Introduction

Multirotor unmanned aerial vehicles have achieved considerable success in the past few years due to their high maneuverability and vertical take-off and landing capabilities. Modern UAVs have been deployed in a wide range of applications such as remote inspection, precision agriculture, search and rescue, aerial photography, site surveying, and package delivery [5, 6, 7, 8, 9].

Figure 4.1: A drone flying in GPS-denied riverine environments requires a navigation policy.

Despite its success, autonomous navigation of UAVs with obstacle avoidance capability in outdoor environments remains a challenge especially when the environment becomes complex and unknown (e.g., GPS-denied riverine environments involve heavy foliage/forest canopy, see Fig. 4.1). In this chapter, a navigation policy is developed which allows the UAV to fly in riverine environments solely relying on visual inputs and is trained with the machine learning algorithm. The drone is able to navigate itself autonomously while avoiding collision with nearby obstacles in the simulation environments.

Traditional approaches to navigating a robot in complex and GPS-denied environments usually integrate the visual-inertial odometry (VIO) or simultaneous localization and mapping (SLAM) techniques [109, 110] with trajectory planning. The procedure consists of localizing the agent with perception sensors (e.g., LiDAR, camera), building a map of the global or local environment, and planning feasible trajectories within the map. The optimal trajectory is then mapped to the control commands of the robot in order to reach the goal points as well as avoid collisions. While these planning-based approaches have been widely adopted by many scholars [111, 112], the algorithm itself can be very computationally intensive and does not guarantee performance when the environment is non-static. The separation of perception and control may also cause unexpected behaviors as pointed out in [69]. In this chapter, a reactive controller is designed in which the control commands are directly calculated from the visual inputs. The proposed controller is able to provide feasible solutions to UAV navigation inside complex riverine environments efficiently and effectively.

Learning-based approaches have achieved great success in solving sequential decision-making problems, for example, in the field of autonomous driving [113] and playing computer games [60]. Among them, reinforcement learning (RL) has gained much attention due to its strong capability and compelling results [58, 114]. Although proven successful in many tasks, the RL approach is known to be sample inefficient and requires a substantial amount of data in order to achieve good results, which makes it unsuitable for many safety-critical systems in the real world. On the other hand, imitation learning (IL) [64, 65], also called learning from demonstrations (LfD), is another attractive approach for a robot to learn a safe control strategy by mimicking an expert's behavior based on the demonstrations. The IL approach overcomes many of the limitations of reinforcement learning and thus is adopted to learn a navigation policy in this work. Human knowledge can be largely utilized and let the autonomous agent learn a good behavior directly and more efficiently from human demonstrators instead of the expensive trial-and-error methods used in reinforcement learning.

In this chapter, a learning pipeline is built and several vision-based policies are proposed which allow the UAV to fly inside GPS-denied riverine environments. The controller computes yaw rate commands directly from the visual inputs of a front-facing camera and navigates the UAV in two-dimensional space with a fixed altitude. The UAV should learn good behavior through the training data given by the human pilot and demonstrate its performance in novel environments that the agent has never seen before. To overcome the issues of classical supervised learning, an intervention-based data aggregation (DAgger) algorithm is adopted and different control policies are trained that are able to command the UAV to perform reactive maneuvers in the simulation environments. The performance of a VAE-based controller with a linear regression-based controller and an end-to-end neural network controller trained from 15 human subjects are compared in the simulation. The performance of different controllers is evaluated by deploying them in novel environments which the agent has never seen during the training. The simulation results show that the VAE-based controller outperforms the linear regression-based controller and end-to-end neural network controller with a lower intervention rate from the pilots and a longer distance traveled by itself. The VAE-based controller also generalizes well to novel environments.

It's noted that while this work focuses on the problem of UAV navigation in riverine environments, the proposed method can be applied to other complex environments (see Chapter 5) which require vision-based solutions and human demonstrations can be collected to benefit the training and therefore allows the agent to learn good behaviors.

4.2 Related Work

There has been a wide variety of work done on navigating a UAV in challenging environments while assuring collision avoidance in the literature. A motion capture system is generally used indoors to get the accurate state of the UAV and then trajectory optimization-based methods can be applied to achieve safe maneuvers in cluttered indoor environments [115, 116]. For outdoor environments, the GPS and other exteroceptive sensors are commonly used to estimate the position of the robot [117, 118]. However, in real-world scenarios when the UAV is flying in heavy vegetation, foliage, and forest canopy, the GPS signals will be significantly degraded or fully absent, and therefore advanced navigation policy is necessary.

Early researchers have attempted to navigate a rotorcraft inside riverine environments but still required intermittent GPS signal [118, 119]. Recent techniques which have been evolved by researchers to allow UAV navigating inside GPS-denied environments involve lidar-based and vision-based approaches. Carrying a lidar on multirotor aircraft may heavily impact the flight time due to a limited payload and power source [120], therefore making vision-based approach a popular choice [111, 121, 122, 123, 124, 125]. Two main categories of vision-based solutions include planning-based approaches [126, 127] and reactive approaches [124, 66]. For planning-based approaches, visual-inertial odometry [109] or simultaneous localization and mapping [110] are commonly used to localize the agent and build a map of the unknown environment. Path planning is performed after retaining a map of the environment and the optimal trajectory is selected to maneuver the robot [128, 129]. The drawback of this technique is that the state estimation may introduce bias into the system, and an incorrect map or localization result will significantly impact the planning performance and therefore leads to unsafe behaviors. Besides, planning-based approaches require a lot of processing power not only for the 3D mapping, but also calculating the optimal trajectory from the candidates [130, 131]. Visual-teach-and-repeat (VT&R) is another appealing method which allows the robot to operate in challenging environments relying on only visual inputs [132]. In the teaching pass, a human operator or a high-level mission planner provides a demonstrated path, and the map and keyframes from the visual odometry are recorded. In the repeating pass, the robot will localize and plan a path to track the pose of keyframes accordingly. The VT&R has been tested successfully in indoor environments [133] and outdoor long-range rover autonomy by using stereo camera [132] and monocular vision [134], and can also deal with the seasonal changes of the environments [135]. However, the VT&R requires the robots to repeat a previously traversed route since it utilizes the maps built in the teaching phase. This method is useful under the scenario that the operation environments remain the same, such as a UAV emergency return-to-launch during a GPS failure [136]. However, the VT&R does not work in new environments in which no route has been demonstrated before, therefore cannot provide a generalizable performance.

On the other hand, a reactive approach directly generates motion commands based on sensor readings which can generally provide faster responses and require fewer computational resources. Existing researches have shown that a reactive controller is able to navigate a UAV in the forest [124, 66] and urban environments [69] successfully with visual inputs, and can generalize well to the new scenes different from the training environments. Traditional methods calculating optical flow [111, 137] or stereo vision [123, 138] provide depth estimation in unknown environments. However, these techniques request powerful computing resources and add extra delay to the navigation tasks [124]. Since the success of deep neural network (DNN) in ImageNet image classification competition [139], increasing efforts have been spent on adopting DNN to learn low-dimensional control commands from high-dimensional image inputs with deep learning algorithms. Even though the training of deep learning methods takes a long time to converge due to its data-driven nature, the deep neural network structure allows it to generate control action directly from raw image data, and therefore, the total execution time is shorter than the classical perception, planning and control framework. The RL is one popular choice and a lot of work has been done in recent years. For example, Mnih et al. [60] used deep Q-network to train an agent playing Atari games based just on pixel image inputs and achieved super-human level performance. Levine et al. [140] used guided policy search with deep convolutional neural networks to train a robotic visuomotor policy that can perform a range of real-world manipulation tasks on unmodeled objects. However, the RL algorithm is known to be sample inefficient and requires a costly data collection procedure. This causes severe problems for training with a safety-critical system. In contrast, IL is another appealing approach that allows the agent to learn the expert's behaviors from demonstrations and has proven to be useful for many real-world problems. For example, Bojarski et al. [113] proposed an end-to-end learning approach for self-driving cars and tested it on the road. Giusti et al. [68] used the imitation learning approach for UAVs to traverse forest trails based on monocular visual perception of trees and foliage. These advantages make IL a good alternative to learning safe behaviors while expert demonstrations are available.

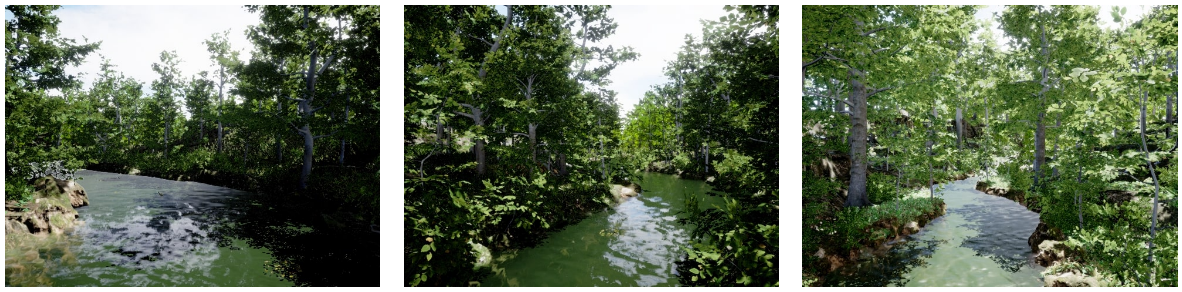
Classical behavioral cloning solves the IL as a supervised learning problem which assumes that the data is independent and identically distributed. This assumption does not hold on real-world data, therefore even small errors may compound over time and lead the agent to a state that it has never seen during the training. This shifted distribution may cause catastrophic failures when the agent does not know how to recover [141]. To overcome these problems, Ross et al. [142] proposed a non-regret online learning approach called data aggregation (DAgger) which learns optimal policy through multiple iterations. The DAgger solves the distribution shift issue by rolling out the learned policy in environments and collecting new data from the learner's own distribution. However, the DAgger has its own drawbacks, for example, the agent is allowed to execute the policy that is not fully trained while the expert does not have sufficient control authority. Human experts also lack feedback from the system trained which is likely to degrade the quality of the provided labels [143]. Considering these two facts, in this work, an intervention-based DAgger algorithm is deployed so that the human pilot can always take over the control when the UAV has reached an unsafe region and provide recovery actions. Relevant work [144, 145, 146] have shown that the intervention-based approach can learn a policy more effectively and achieve better performance.

4.3 Methodology

The proposed method is introduced in this section. It starts with introducing the simulation environments designated for the training, then talks about different options for the vision-based control policies and describes the imitation learning framework.

4.3.1 Simulated Riverine Environments

The synthetic environments used to train the navigation policies are created in Unreal Engine 4. The drone simulator is powered by the Microsoft AirSim [147] which provides useful tools for low-level UAV controls and computer vision and can be directly integrated into the Unreal Engine. High-fidelity riverine environments are designed in the Engine to simulate the real-world looks of rivers and foliage used for the training. These custom riverine environments or maps are divided into three difficulty levels (i.e., easy, medium, and hard). In the easiest maps, rivers are wider and straighter while in the hardest maps, rivers are narrower and more curved. This variety avoids the issue of over-fitting to the specificity of the simulation environments or maps. Example maps are displayed in Fig.4.2.



(a) (b) (c)

Figure 4.2: A subset of the environments used for training with different difficulty levels: (a) easy; (b) medium; (c) hard.

4.3.2 Controller Design

Different from other research which assumes that the pilot has global observability of the environment, here the pilot is allowed to share the same observability as the agent and only rely on the onboard camera to command the drone. For the vision-based navigation, three different controller options have been considered: 1) a linear regression-based controller, 2) an end-to-

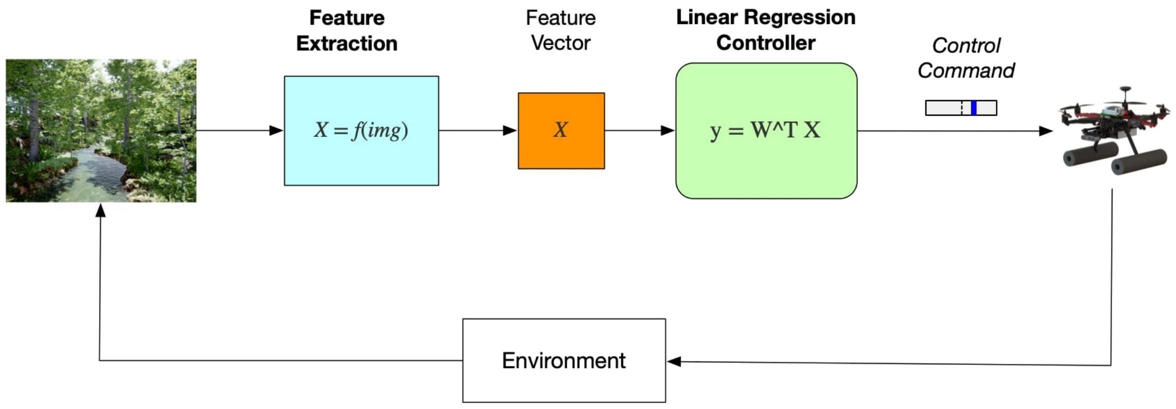


Figure 4.3: System diagram of the linear regression-based controller.

end neural network controller, and 3) a variational autoencoder-based controller, and each of them is introduced in sequence.

4.3.2.1 Linear Regression-Based Controller

The design of the linear regression-based controller is inspired by the work in [111] and [66]. The raw image from the camera is first passed through a feature extraction script defined by human experts and tuned manually. The features extracted from the raw images are common visual features used in navigation (e.g., ground or aerial vehicles) and popular methods in the literature on computer vision. The reason for investigating the performance of this controller is that first it has proven to work on real UAV platforms [66], and second, unlike other complicated methods which use neural networks, this type of controller is more hand-crafted, easier to tune, and also converges faster. The raw RGB image which has a resolution of 320×240 is first split into 6 vertical and 8 horizontal windows without overlap. The sliding windows not only help to accelerate the computation but also provide better results empirically since not all regions of the image have equal importance in making the decision for navigation. The visual features are calculated for each small window. Four visual features are extracted from the image in this work, which are:

(1) *Hough Transform*: Hough transform is used to calculate the dominant lines in each window. A Hough transform is applied to a window across 15 angles. This yields a matrix whose columns represent the angles and whose rows represent the pixel width of the image. The rows are then reduced by averaging every five rows. Finally, the largest two

values for each angle are selected and arranged into a 30-element vector and used as the Hough transformed features for the window.

(2) *Law's Masks*: Law's masks are used to get information about the textures in the image. The masks are selected based on their effectiveness in traditional machine vision applications which are L5E5, L5S5, L5R5, E5E5, E5S5, E5R5, S5S5, S5R5, R5R5. L represents the level, E represents the edge, R represents the ripple and S represents the spot. Each window is first converted to YCrCb colorspace and the Y channel is filtered with all the masks. The filtered windows are then averaged to get the 9-element feature vector.

(3) *Structure Tensor*: Structure tensors represent the structures and shapes in a window. The coherency and orientation angle at each point in a window is first calculated and then the coherency value is accumulated with a 15-bin histogram for the entire window based on the angles. This yields features in the form of a 15-element vector.

(4) *Optical Flow*: Optical flow feature shows the motion in the current frame compared to the previous one and works as a depth estimation to the image. Here the dense optical flow is calculated and the maximum, minimum, and averaged magnitude of the optical flow in each window are taken to form a 3-element vector as the flow features.

The collected feature vectors for each window are stacked and form a big vector X for the whole image. The feature vector is normalized to have zero means and unit variances to balance the contributions of different visual features. After that, ridge regression is performed on the data collected from the human demonstrations and calculates the weight of different feature elements. The controller outputs the control command and sends it to the drone. The structure of the linear regression-based controller can be visualized in Fig. 4.3.

4.3.2.2 End-to-End Neural Network Controller

The second controller considered is a pure end-to-end controller composed of neural networks. The end-to-end training combines perception and control so the control policy is trained all at once. The control command is generated autonomously from the pixel values read from the camera. The design of the end-to-end network is similar to the one in [69]. However, the prediction on the probability of collision is removed in the final layer and it takes RGB images as the inputs instead of grayscale images in the original paper (see Fig. 4.5). The architecture of the network is a single convolutional neural network consisting of a ResNet-8 with 3 residual blocks followed by dropout and ReLU activation functions. The output is then passed to a fully-connected layer to carry out yaw angular velocity prediction. The dimension of the input image is 64×64 and the output is the control command to the UAV.

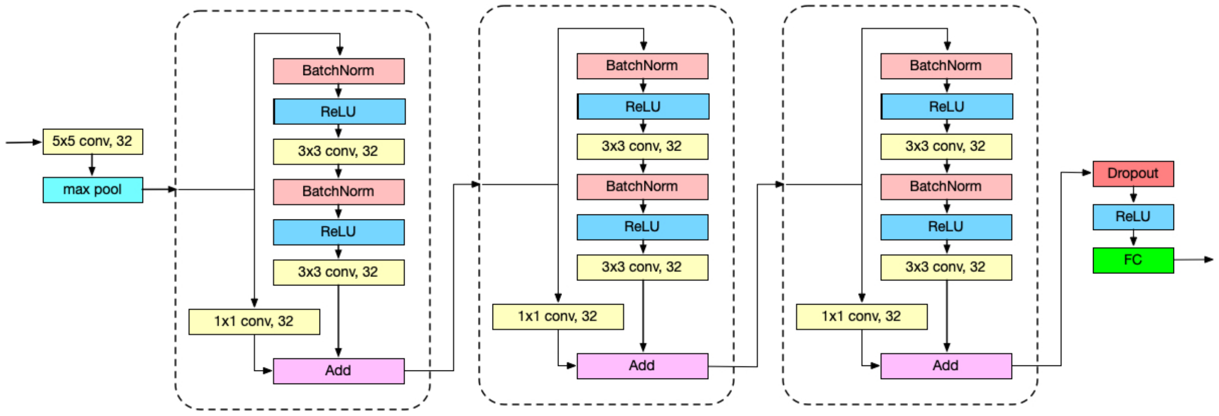


Figure 4.4: The architecture of end-to-end neural network.

The structure of the end-to-end neural network is displayed in Fig. 4.4. When deciding the network structure for the end-to-end controller, other popular models have also been considered, such as the ResNet-34, ResNet-50 [148], and VGG-16 [149]. To decide which model fit here the best, different models were trained on the same dataset and it has been found that the neural network in Fig. 4.4 can achieve an equivalent performance while it has a much smaller network size as well as fewer parameters to tune compared with the other candidates, therefore the training time is significantly reduced. As a result, this simple, lightweight, and powerful neural network was used as one of the three controllers.

4.3.2.3 Variational Autoencoder (VAE)-Based Controller

The third controller considered is based on the variational autoencoder. The VAE is known to have a good capability to compress the data from high-dimensional space into low-dimensional space [150]. The extracted low-dimensional result, also called latent variables, normally represents some dominant aspects of the original images (e.g., position, scale, rotation, lighting).

The steps to training the VAE-based controller are twofold: First, a VAE model is trained based on the image data collected from the simulation environments. After the results of the VAE network are satisfying, the encoder network is taken out from the VAE network and used to compute latent representations. The latent representations are regarded as the input to the controller network which is composed of a neural network. Next, the controller network is trained by freezing the parameter values in the pre-trained encoder network and only the weights in the controller network are updated based on the gradients.

$$\mathscr{L} = \mathbb{E}_{q(z|X)}[log(p(X|z))] - \beta\ D_{KL}[q(z|X)||p(z)] \tag{4.1}$$

Different types of VAE algorithms have been considered at the early stage, which include the vanilla-VAE [150], β-VAE_h [151], β-VAE_b [152], factor-VAE [153], and VAE-GAN [154]. Empirically, it has been found out that the β-VAE_h works better than all the other structures in terms of the training stability and disentanglement ability, thus the β-VAE_h method is adopted here to train the VAE model. The objective function can be seen in Eqn. (4.1), and β is selected to be 10. The architecture of the VAE network is presented in Fig. 4.6 which consists of 4 convolutional layers in the encoder network and 4 transpose convolutional layers in the decoder network. The reconstruction performance of the VAE is shown in Fig. 4.7. The reconstructed images look similar to the raw images which shows a good training result of the VAE. The structure of the VAE-based controller can be visualized in Fig. 4.8. The raw image is first resized to 64×64 and fed into the encoder network which was pre-trained on 200K images

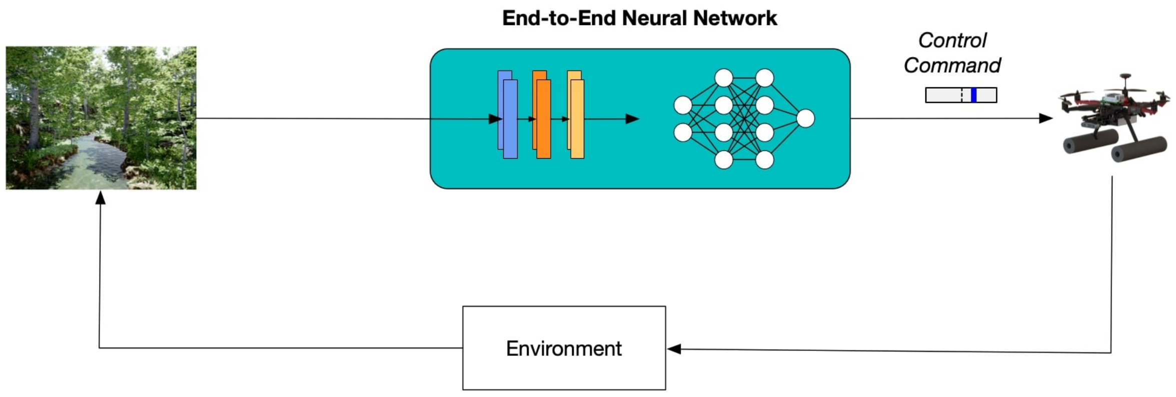


Figure 4.5: System diagram of the end-to-end neural network controller.

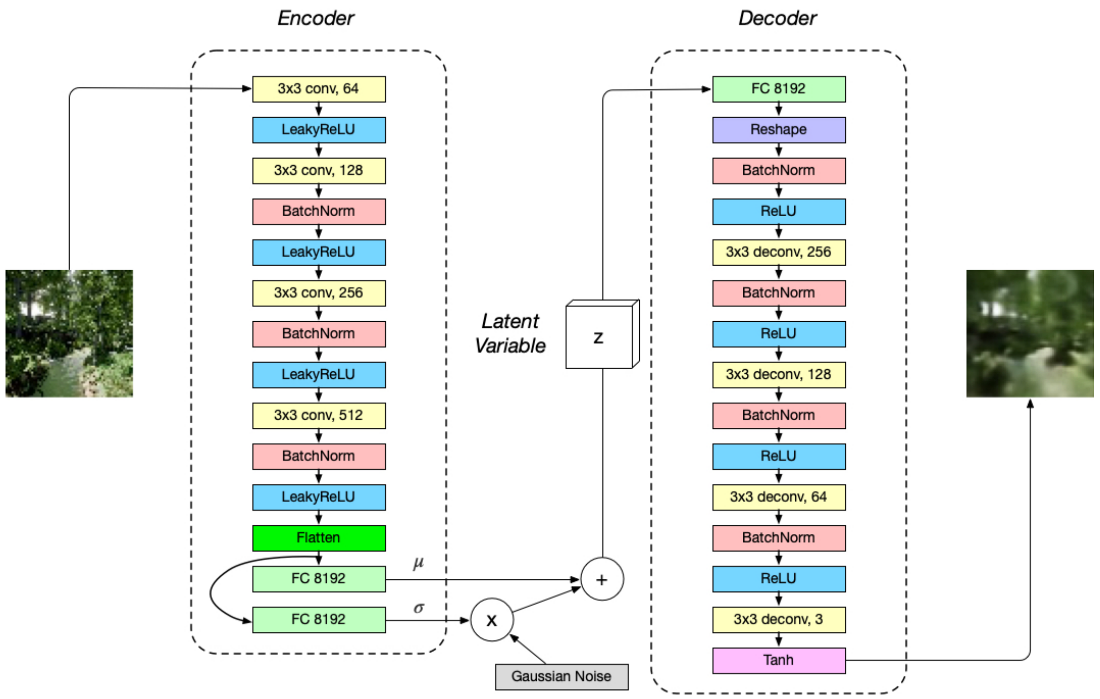
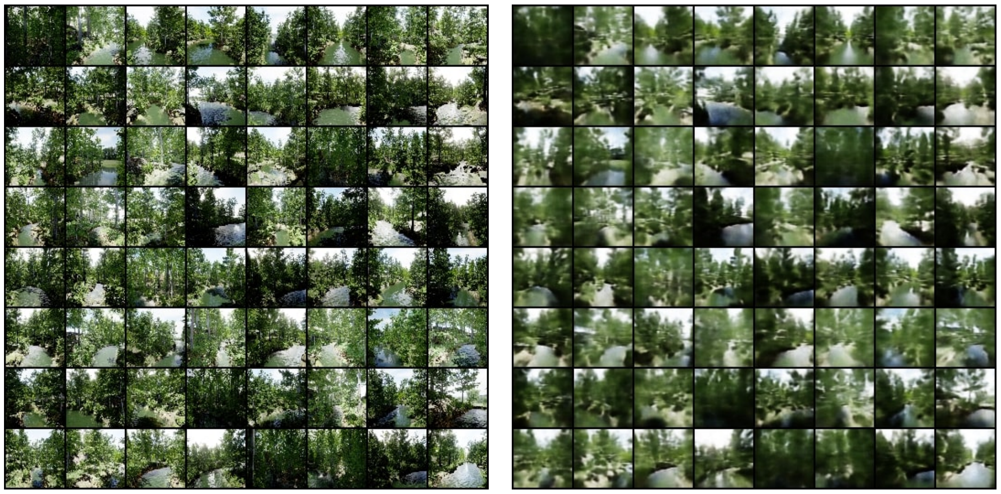


Figure 4.6: The architecture of variational autoencoder network.



(a) sample images (b) reconstructed images

Figure 4.7: The reconstructing performance of the VAE, which (a) are the sample images and (b) are the reconstructed images trained after 150 epochs.

collected in the simulation environment, and then frozen during the training of the controller network. The controller network is trained through imitation learning. The latent representation has a dimension of 64 and the controller network is composed of the fully-connected network

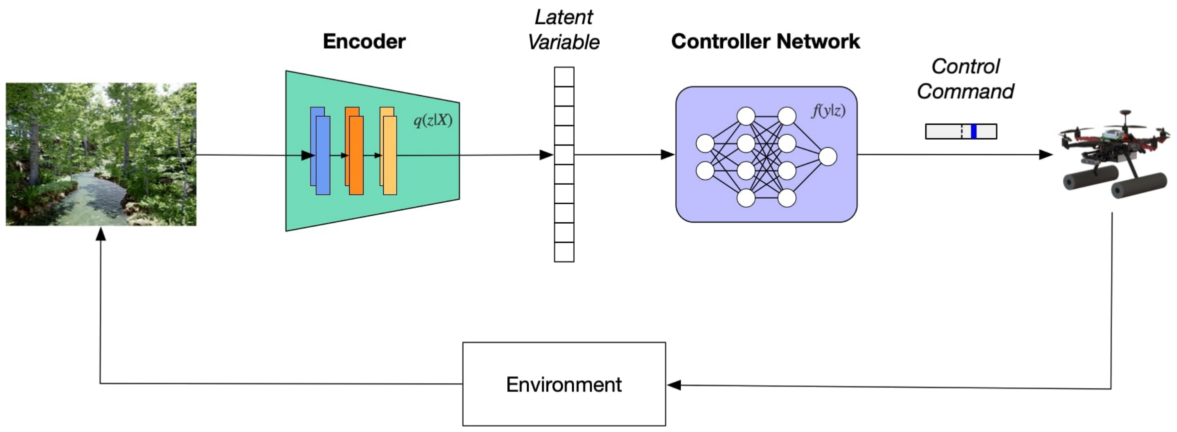


Figure 4.8: System diagram of the VAE-based controller.

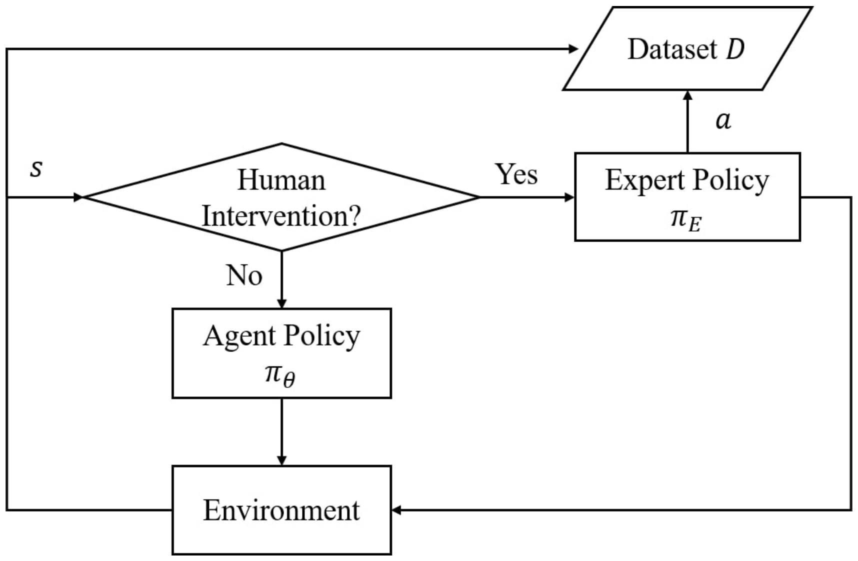


Figure 4.9: A conceptual illustration of the intervention-based learning method.

which has two hidden layers. The output of the controller network is the control command to the UAV.

4.3.3 Learn from Demonstrations

Classical behavioral cloning does not perform well due to the distribution shift between the training and testing datasets and may cause catastrophic results. A more proper way to utilize human knowledge is by taking the DAgger which the control policy is updated through multiple iterations. At each iteration, the agent control policy is executed and afterward human experts are queried to provide correction commands. This method works well in many applications which typically have slow dynamics and discrete action space.

However, there are several issues with the DAgger, for example, the agent is allowed to control the robot at the early stage of the training which can be dangerous. Meanwhile, since the querying process is taken offline, the experts lack visual feedback from the environment which means they do not know how their corrective demonstrations will perform on the robots. Instead of querying an expert offline, the approach taken in this work is that the expert is queried in real time when the drone is flying in the simulation environment. The human expert or pilot has the authority to take over the control of the agent policy when he thinks the agent output may lead to a failure or unsafe situation. When the pilot takes over the control from the agent, the new demonstrations are appended to the training dataset. After a certain period of time when the pilot puts the drone back to a normal and safe condition, he can return the control back to the agent. The policy is retrained after each iteration as the normal DAgger is conducted. This method is also called intervention-based DAgger which the human expert can intervene in the agent's control online anytime. Another benefit of this method is that the expert has continuous time to control the robot which is more natural in real life. This type of imitation learning approach has been adopted in the literature [144, 145] and proven beneficial. The structure of the adopted intervention-based DAgger framework is displayed in Fig. 4.9.

4.3.4 Experiment Design

It is realized that allowing the human pilots to control the height makes imitation learning quite challenging due to the multi-model behaviors from the human pilot and causes the state space to become too large. Also, sometimes the pilot preferred to stay at extreme locations (e.g., very close to the water or high above the river) to increase the success rate but it lacks obstacles at these locations. Therefore, the learned policy is not useful under normal situations. To address these issues and simplify the problem, the height of the UAV is fixed to 5 m and the forward speed is fixed to 2 m/s which is achieved by a low-level controller. The yaw angular velocity is set to 0 by default. Therefore, without any input, the UAV will keep flying forward and not consider collisions along the way. The vision-based controller is designed to provide the UAV ability to avoid obstacles along the river path based on the visual features by generating the yaw rate control command. The human pilots are only allowed to adjust the yaw angular velocity during the demonstration. The maximum yaw angular rate is limited to 45 deg/s to avoid instability. The front-facing camera data is gathered at 10 Hz and the control command is updated at the same rate.

To compare the performance of different controllers, 15 human subjects were recruited to collect demonstrated data in the simulation environments. The 15 human subjects were divided into 3 groups, and each group contains 5 people. Group one was trained with the linear regression-based controller, group two was trained with the end-to-end neural network controller, and group three was trained with the VAE-based controller. To ensure the results are compatible, both new pilots and experienced pilots were assigned to each group to balance the knowledge level of the human subjects. The expertise levels of the human subjects are given in Table. 4.1. Each human subject was required to fly the drone in six maps and provide corrective demonstrations using the intervention-based DAgger method to collect training data. The difficulty levels of the maps are displayed in Table. 4.2.

With three different controllers described above, two hypotheses have been made and going to be tested in this chapter:

Table 4.1: The expertise level of different human subjects. The human subjects are split into three groups and each group consists of both new and expert pilots.

Group One	Pilot 1	Pilot 2	Pilot 3	Pilot 4	Pilot 5
Expertise Level	Beginner	Beginner	Expert	Beginner	Expert
Group Two	Pilot 6	Pilot 7	Pilot 8	Pilot 9	Pilot 10
Expertise Level	Beginner	Expert	Expert	Beginner	Beginner
Group Three	Pilot 11	Pilot 12	Pilot 13	Pilot 14	Pilot 15
Expertise Level	Expert	Beginner	Beginner	Beginner	Expert

Table 4.2: The level of difficulty for different maps.

	Map 1	Map 2	Map 3	Map 4	Map 5	Map 6
Difficulty level	Easy	Easy	Medium	Medium	Hard	Hard

Hypothesis 5 *The agent trained with a VAE-based controller can achieve a better performance than an agent trained with a linear regression-based controller and an agent trained with an end-to-end neural network controller using the same DAgger method and number of iterations.*

Hypothesis 6 *The VAE-based controller generalizes well to the novel environments which the agent has never seen during the training compared to a linear regression-based controller and an end-to-end neural network controller.*

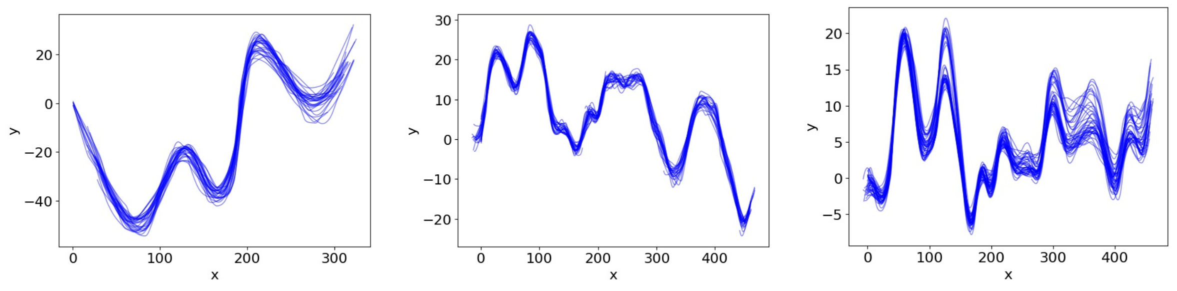
Due to the strong capability of deep neural network and the outstanding performance of VAE in the task of computer vision, in the first hypothesis, it is hypothesized that the VAE-based controller will achieve the best performance among all three controller options if they are trained with the same imitation learning method and the level of training. Since the VAE compresses the high-dimensional visual inputs to low-dimensional latent representations, it is believed that such compression will extract the key information from the raw data and makes the training more effective and efficient. In the second hypothesis, the goal is to test the generalization capability of three controllers. Since the VAE is expected to extract the hidden dominant features in the latent space, it is believed that such intermediate representations make the VAE learn a more general control policy and are less sensitive to the distribution shift between the training and testing datasets. Therefore, it is hypothesized that the VAE-based controller will have a better performance compared to the other two controllers when deployed in the novel environment which the agent has never seen during the training. The evaluation metrics used to compare the performance of different controllers and test the two hypotheses are described as follows:

- Intervention rate from the human pilots during the training.
- Maximum distance that an agent can travel without human interventions.
- Average distance flown by the agent before a failure.
- Processing time for each frame to generate the control command.

4.4 Experimental Results

This section presents the results of three controllers trained from 15 human subjects in the simulation environments. Section 4.4.1 shows the results in training environments and section 4.4.2 describes the performance of the three controllers in testing environments. The results will be used to evaluate the performance of different vision-based control policies navigating a UAV in riverine environments and also test the two hypotheses. All the simulations and training were running on an AMD Ryzen 3900X CPU, 64 GB RAM, and RTX 2080Ti GPU computer in the lab.

4.4.1 Training Results



(a) (b) (c)

Figure 4.10: Example trajectories of the UAV in simulation environments (a) Map 1 (easy); (b) Map 3 (medium); (c) Map 5 (hard).

As mentioned in the previous section, each human subject group was trained with a specific controller type using the intervention-based DAgger approach. More specifically, group one was trained with the linear regression-based controller, group two was trained with the end-to-end neural network controller, and group three was trained with the VAE-based controller. For each group, all three difficulty levels of the maps were utilized and the pilots were requested to navigate the drone inside each map along the river path with random initial positions and avoid collisions. Example trajectories of the UAV flying in some environments are provided in Fig. 4.10.

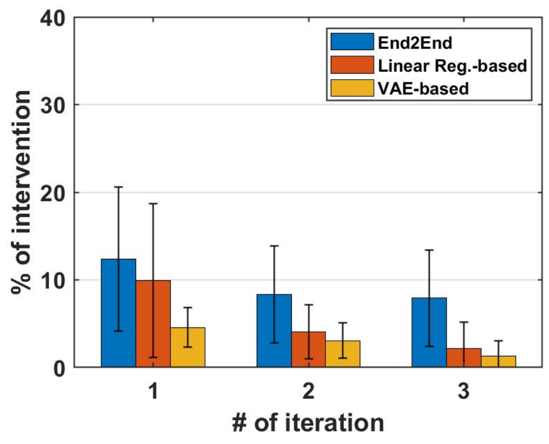
Ideally, the performance can keep improving with more and more data collected from the environments as an increasing number of iterations. However, empirically it has been found out that the pilots had a difficult time judging whether the decision made by the agent would lead to a collision or not as the training iteration increases and the agent policy performance gets improved. As a result, the corresponding demonstrations collected from the human at those iterations may degrade the agent performance and the retrained agent policy will actually become worse. Some early training results show that the pilots are able to provide accurate and good demonstrations up to 3 iterations and therefore, the training is terminated after 3 iterations.

To test the hypotheses, first, the percentage of intervention from human pilots during the training is calculated for each controller type. The results are averaged among all human subjects in the same group and the mean and variance values are provided in Fig. 4.11. Based on the results, it is observed that the intervention rate from human pilots decreases as the training iteration increases which shows the effectiveness of the imitation learning method. Since the hard maps contain sharp curves and therefore become more challenging for the UAV to navigate, the intervention rate in these maps is higher than the intervention rate in the easy and medium maps. In addition, it can be seen that the VAE-based controller has the lowest intervention rate across various maps with different difficulty levels while the end-to-end neural network controller requires the most human intervention compared to the other two controllers. This remains true over different training iterations. A T-test is also performed which calculates the t-score and p-value about the intervention rate between the linear regression-based and VAE-based controller, as well as the end-to-end neural network and VAE-based controller. The results are provided in Table. 4.3. The significance level is chosen to be $\alpha = 0.10$ and a p-value smaller than α means that the null hypothesis can be rejected. Based on the calculated T-test results, it is found that the VAE-based controller outperforms the linear regression-based controller and end-to-end neural network controller with significance.

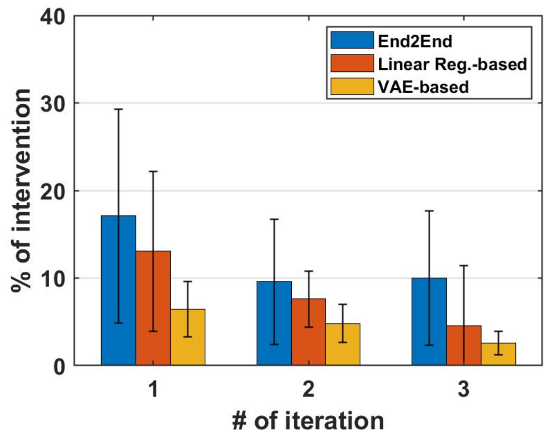
4.4.2 Testing Results

Next, the trained controllers are deployed into the testing environments. An agent that has been trained to perform well in the training environments does not inherently guarantee to perform well in a target or novel environment. Therefore, the controllers trained after 3 iterations are executed in the testing environments which the agent has never seen during the training time to evaluate their performances in novel environments.

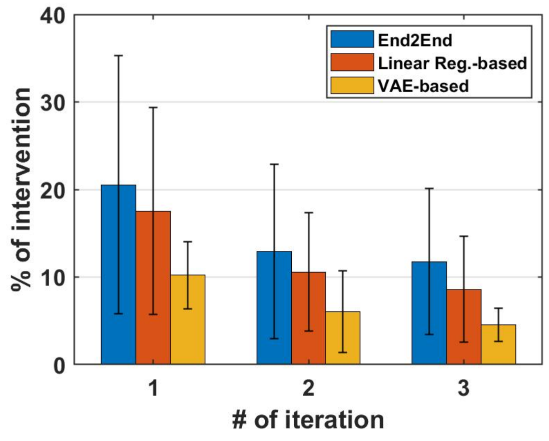
The maximum distance that the drone can fly by itself before a failure is displayed in



(a)



(b)



(c)

Figure 4.11: Percentage of interventions from human pilots in (a) easy maps; (b) medium maps; and (c) hard maps during the training.

Table 4.3: T-Test results with a significance level $\alpha = 0.10$.

Maps	T-test	Linear reg.-based vs. VAE-based			End-to-end vs. VAE-based		
		iteration1	iteration2	iteration3	iteration1	iteration2	iteration3
Easy	t-score	1.8785	0.8205	0.7302	2.8860	2.8364	3.6180
	p-value	**0.08292**	0.42328	0.47583	**0.01275**	**0.01319**	**0.00280**
Medium	t-score	2.1600	2.2705	0.8890	2.6620	2.0282	2.9990
	p-value	**0.04859**	**0.03647**	0.39147	**0.01956**	**0.06201**	**0.01109**
Hard	t-score	1.8563	1.7503	2.0190	2.1440	1.9757	2.6537
	p-value	**0.08458**	**0.09809**	**0.06308**	**0.05153**	**0.06689**	**0.02103**

Fig. 4.12. The testing map has a total length of 225 meters and the agent starts from the same location every time. Since all the agents start from the same initial location, the scenes they experienced remain the same and therefore the results are compatible. The distances they traveled are averaged across 5 trials and sorted by the mean value. Based on the results, it can be seen that the VAE-based controller can achieve the longest maximum traveling distance while the end-to-end neural network controller has the shortest maximum traveling distance before a failure. In fact, two VAE-based controllers can almost reach the end of the river without any intervention or failure. The reason why the end-to-end neural network controller achieves the worst behavior is probably that the end-to-end method highly relies on the distribution of datasets, and therefore, when the testing dataset is different from the training dataset, it cannot provide consistent performance. The linear regression-based controller achieves better performance compared to the end-to-end neural network controller but still crashes earlier than a VAE-based controller. From this result, the VAE-based controller beats the other two controllers with significance and generalizes well to scenarios completely unseen at training time which supports the two hypotheses.

The average distance traveled per failure for each controller type is shown in Fig. 4.13. An intervention from a human pilot is performed and considered a failure when the UAV is about to crash. The UAV started from random locations in the testing map and the total distance it traveled and the number of failures it experienced were counted. The average distances per

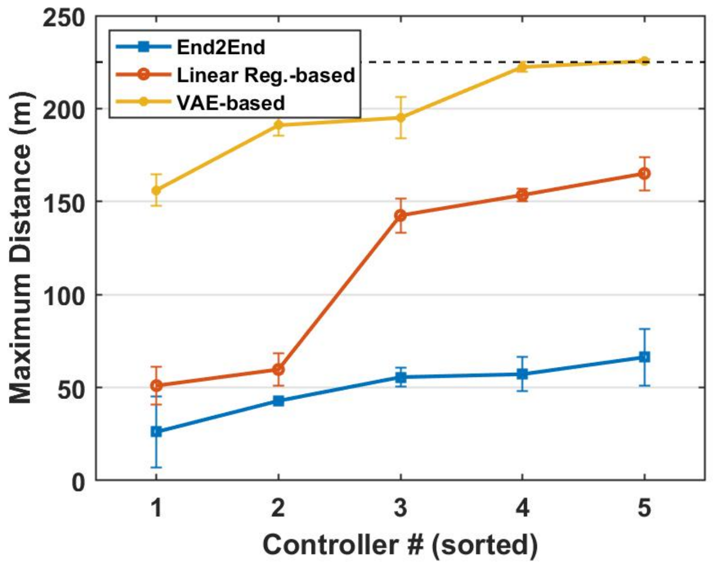


Figure 4.12: Maximum distances traveled before a failure by three different controllers.

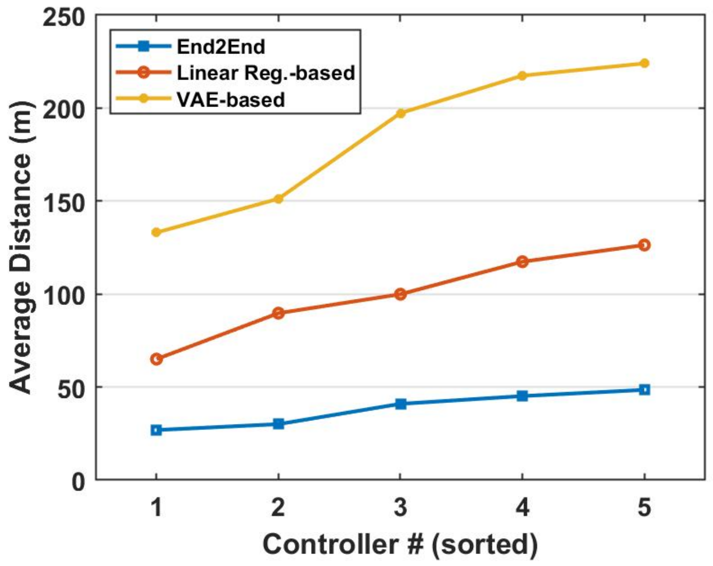


Figure 4.13: Average distances traveled per failure by three different controllers.

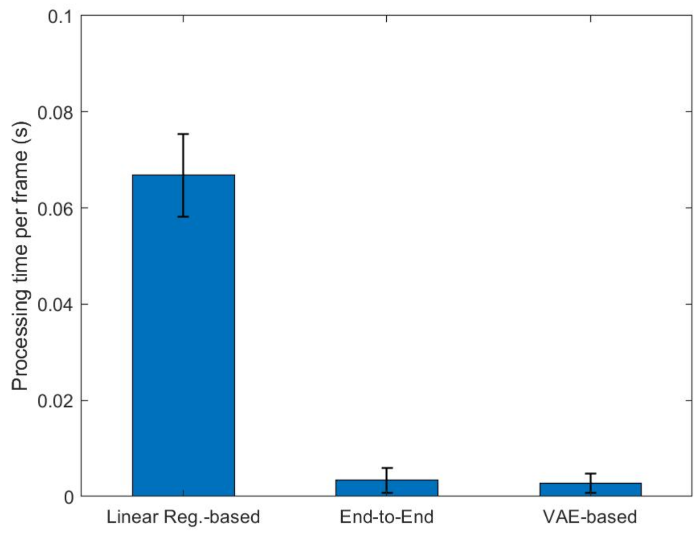


Figure 4.14: Processing time of three different controller types.

failure are then calculated and sorted by the values. Similar to the results from the maximum distance, it can be seen from the figure that the VAE-based controller achieves better performance compared to the linear regression-based controller and the end-to-end neural network controller with a higher average distance flown by itself.

Since the processing time for different maps varies slightly due to the rendering speed of the simulation, only the processing time in the same map during testing gets compared. The processing time results are provided in Fig. 4.14 and the numeric values are shown in Table. 4.4. Based on the results, it can be seen that the linear regression-based controller takes the longest time to process due to the feature extraction part while the two neural network-based controllers can predict the control command at much faster rates. The VAE-based controller is slightly faster than the end-to-end neural network controller due to a simpler structure but the difference is negligible.

The performance of the controllers is further investigated under different lighting conditions to see how robust each controller is to the changes in the environment. To achieve that, the best controllers from each human subject group are selected based on the average distance performance in the testing map. This gives three different controllers and their performances under normal lighting conditions. The lighting of the environment is then changed to a dark condition

Table 4.4: Numerical values for the processing time of three different controllers.

	Processing time per frame (s)
Linear-Regression-Based Controller	0.0668 ± 0.0086
End-to-End Neural Network Controller	0.0034 ± 0.0026
VAE-Based Controller	0.0028 ± 0.0020

and a bright condition and deployed the controllers in the same map. The changes in the camera view can be seen in Fig. 4.15. The comparison results of three controllers under different lighting conditions are displayed in Fig. 4.16. It has been found out that the VAE-based controller is less sensitive to environmental lighting changes and has a robust performance. The linear-regression-based controller performance does not change much in bright light but suffers in dark conditions. The performance of the end-to-end neural network controller degrades significantly for any lighting changes in the environment.

Another set of experiments was performed to investigate the effect of height errors on the controller performance. Since the UAV was commanded to fly at a fixed height during training, it is interesting to see whether the controller is robust enough when height error is introduced. Therefore, the same setting from previous experiments was used which consists of three controllers under nominal conditions. During the experiment, the target height was set to 3m, 5m, 7m, and 9m, respectively, where 5m is the height used for training and the average distance traveled per failure was calculated under different height conditions. It is noticed that since the environment is unstructured, the drone sometimes flew above the river bank to avoid obstacles at high altitudes while no collisions occurred at those locations. In order to make the comparison fair, the UAV was restricted to only fly along the river path and it was considered a failure if the drone attempted to pass the river bound. The performance is plotted in Fig. 4.17. Based on the results, it can be seen that the controllers are robust to small height errors. The VAE-based controller outperforms the other two controllers under various height conditions. An interesting finding is that the linear-regression-based controller experienced fewer collisions at lower

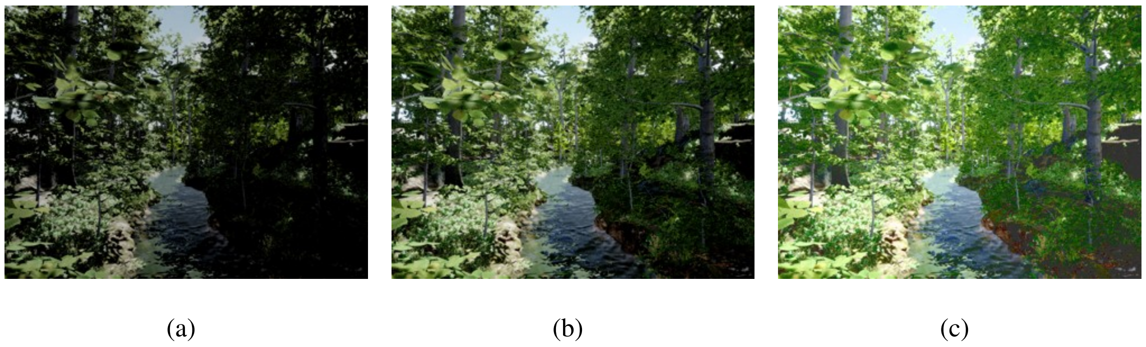


Figure 4.15: Three different lighting conditions: (a) dark (b) normal and (c) bright from the same environment.

altitudes which may be due to the fact that the images at low altitudes contain more edge and texture information. The performances of all three controllers dropped significantly when the drone was flying at 9m. This is due to the change of perspective and leads to a distribution shift issue in which the images contain a large portion of the sky and fewer visual features at high altitudes compared to the training data. One solution to it is to use a gimbal-stabilized camera to compensate for the perspective change similar to the one in [136] and collect more demonstration data at various altitudes.

Based on these results, the two hypotheses have been successfully tested and it can be concluded that the VAE-based controller is better than the linear regression-based controller and the end-to-end neural network controller in the task of navigating UAV inside simulated riverine environments with a lower intervention rate from the human pilot, a longer maximum traveling distance before a failure, a longer average traveling distance per failure, and a faster processing time. The VAE-based controller also generalizes well to novel environments and is robust to the lighting changes of the environment. For supplementary video see: `https://www.youtube.com/watch?v=aPOqHHGbZgs`

The performance of the VAE-based controller is also compared with a visual-teach-and-repeat controller in [132, 134]. The training and testing maps are different in terms of the river shape, landscapes, and lighting conditions. The trajectories for both environments as well as sample images along the river are plotted in Fig. 4.18. From the figure, it can be seen that the training and testing map have quite different trajectories. Three metrics are selected

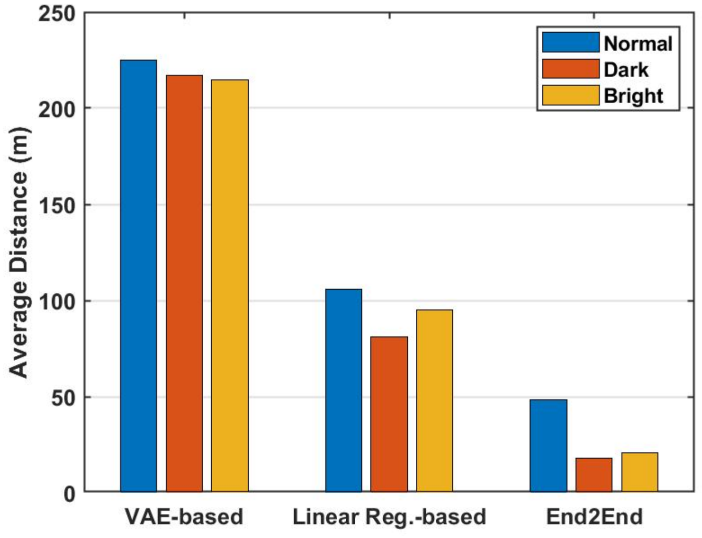


Figure 4.16: Average distances traveled per failure of three controller types under different lighting conditions.

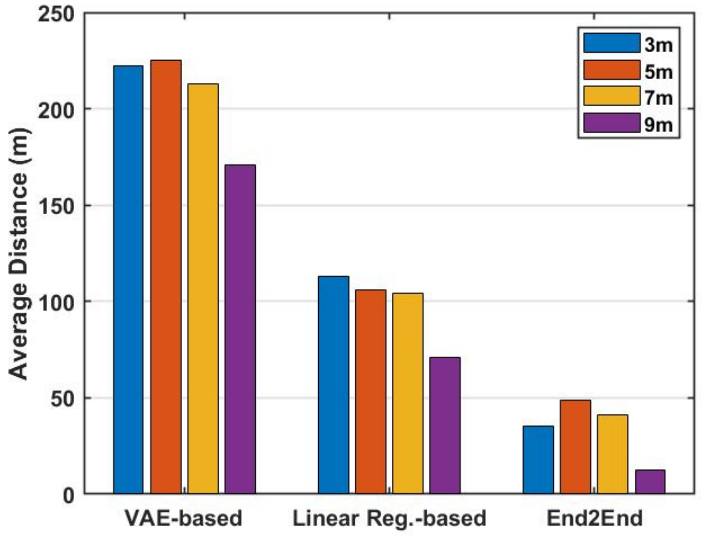
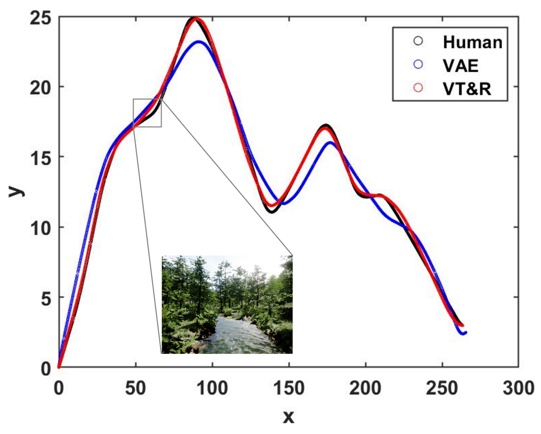
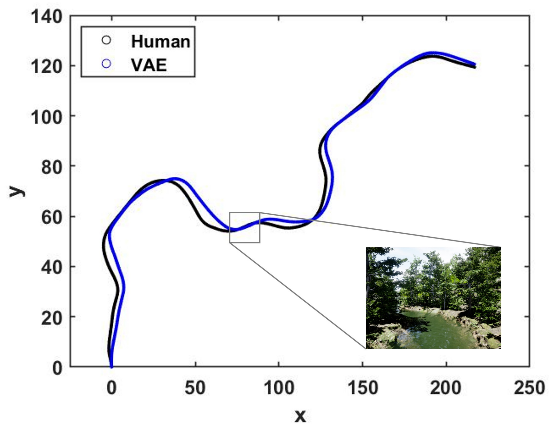


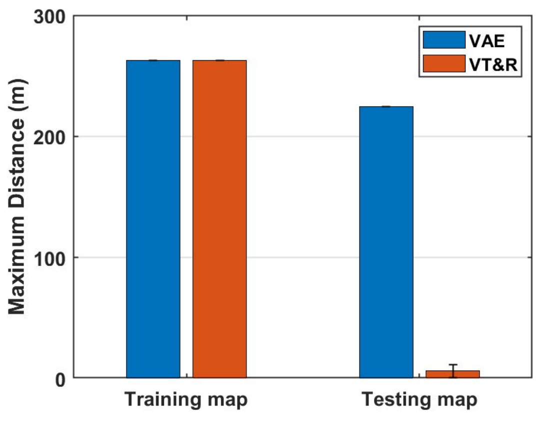
Figure 4.17: Average distances traveled per failure of three controller types under different heights.



(a)



(b)



(c)

Figure 4.18: The proposed VAE-based controller versus a visual-teach-and-repeat controller. (a) Trajectories of both controllers in the training (demonstrated) map, (b) Trajectories of the VAE-based controller in the testing map, and (c) Maximum distance that the agent traveled before a failure in the training and testing map.

to evaluate the image similarity between two maps at a similar left-turn location (two sample images displayed in Fig. 4.18(a) and (b)). The metrics include the root mean squared error (RMSE), structural similarity index (SSIM), and perceptual hash (PHASH). The results of them are RMSE = 91.6 ([0,255], 0 means identical), SSIM = 0.112 ([0,1], 1 means identical), and PHASH = 0.507 ([0,1], 1 means identical), respectively, which shows that the visual inputs from the testing map can be considered novel compared to the training one. It's noted that no new trajectory information should be provided in the testing map because the goal is to evaluate the generalization capability of the controller in novel environments which the agent has never seen before. Therefore, the VT&R agent used the stored trajectory from the training map and visual inputs to navigate in the testing map while the VAE-based controller only relied on the visual inputs. The trajectory in Fig. 4.18 (b) from the human is only used for visualizing the river path and not used by either controller. Based on the experiment result, it can be seen that in the training map (263m long in total) the VT&R controller tracked the demonstrated route very precisely and reached the final point without failure. The VAE-based controller also managed to navigate the vehicle to the final point without a crash, but since it learned a reactive behavior, the VAE-based controller does not necessarily need to follow the human trajectory. Then the two controllers were tested in the new environment (225m long in total) which both agents have never seen before. The VT&R cannot manage the task since the correct trajectory in this map was not provided. By contrast, the proposed VAE-based controller was able to complete the mission. The trajectory of the VT&R controller in the testing map is not plotted since it always failed at the beginning.

4.5 Discussion

Different reactive navigation policies are trained based on human demonstrations that can reliably control a drone from a single forward-looking camera in this chapter. Compared with other approaches, the proposed controller requires no planning and localization in the environment, thus saving a lot of processing power and computing time. It does not require performing any image pre-processing except the down-sampling in order to increase the training speed. Results show that the controllers can handle the noise pretty well and learn a robust control policy. It is

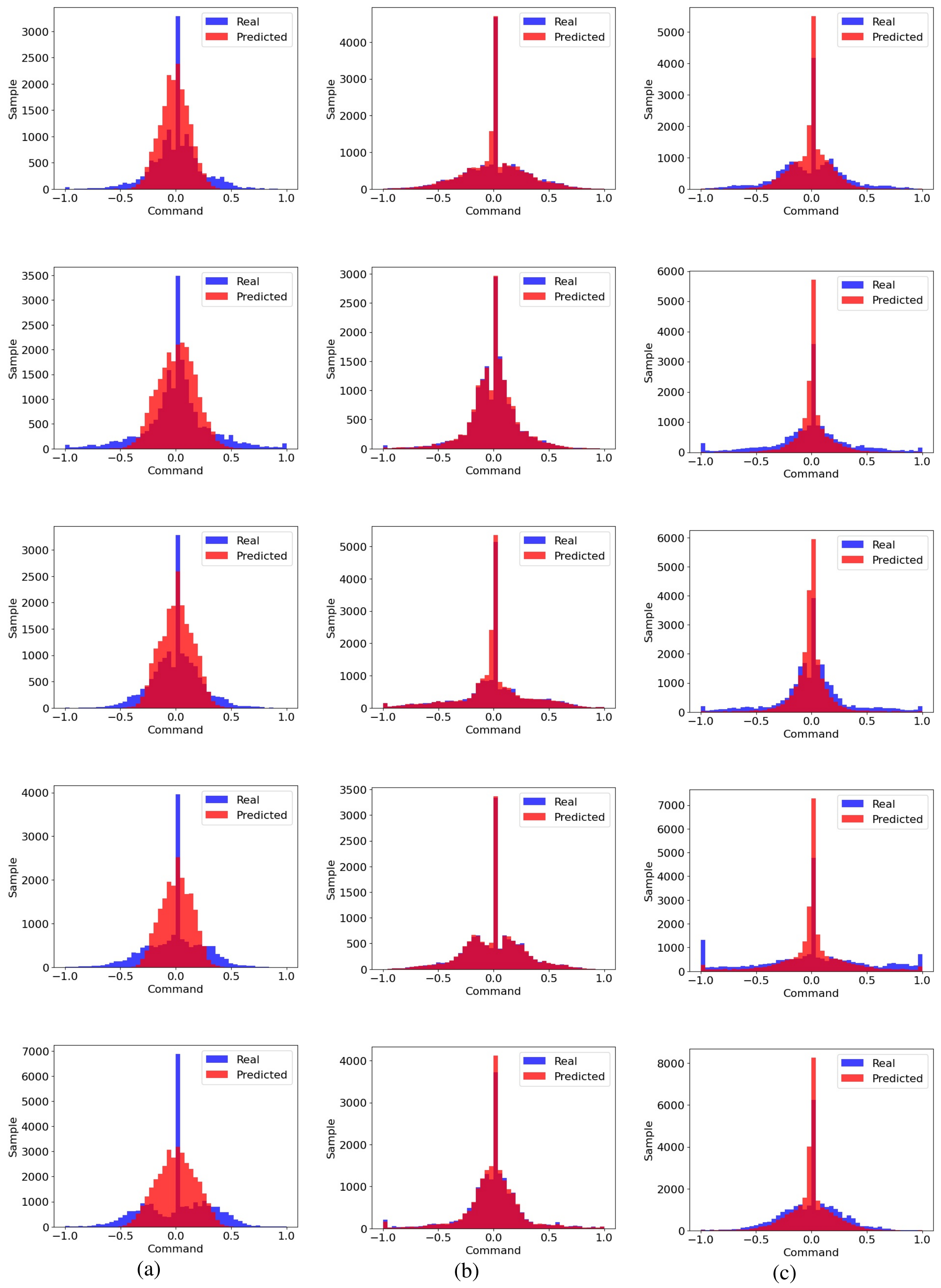


Figure 4.19: Histogram of pilot demonstration data compared to the model prediction. The pilots are divided into three groups evenly, and each group contains five members. Each group is trained on a different controller type: (a) linear regression-based controller; (b) end-to-end neural network controller; and (c) VAE-based controller.

also observed that the non-collision can be achieved without information on the current velocity of the UAV.

Based on the training and testing results, it can be found out that the end-to-end neural network controller performs badly in both training and testing environments. In order to understand why it happened, the histogram of the pilot demonstration data is plotted in Fig. 4.19. The pilots are divided into three groups evenly and each column in the figure represents the data trained from a different controller type. Based on the histogram results, it is noticed that the end-to-end neural network controller tends to overfit the existing data in general and therefore does not generalize well to the new state-action pairs if their distributions are different. With a limited amount of data, the end-to-end neural network controller will result in worse performance compared to the other two controller types. To improve its performance, more data needs to be collected to cover the distribution of the entire space, however, this can be extremely labor intensive since the demonstrations in this work are provided by human experts. It's also observed that since ridge regression is used to calculate the weight for the linear regression-based control, the shape of the predicted control command is more conservative and concentrates in the middle region compared to the other two neural network-based controllers. This causes the linear regression-based controller unable to generate aggressive enough commands when the drone needs a large yaw angular rate command in some maneuvers. The VAE-based controller, on the other hand, balances the distribution variance and the fitting performance, therefore compete the other two controllers.

The method in [48] is applied to investigate which part of the image is the most important for the neural network to make the decision. In Fig. 4.20, the VAE-based controller and end-to-end controller are supplied with the same image and the activation maps are displayed from both controllers. To make the comparison fair, the two controllers are trained by the same human pilot, therefore the behavior is consistent. From the figures, it is noticed that the VAE-based controller concentrated on the vegetation on the right side, and it generated a left-yaw command to avoid that region. In contrast, the end-to-end controller was less concentrated, attracted by different parts of the frame, and resulted in a near-zero output that may lead the drone to crash. Based on this result, it is believed that the VAE part benefits the controller to

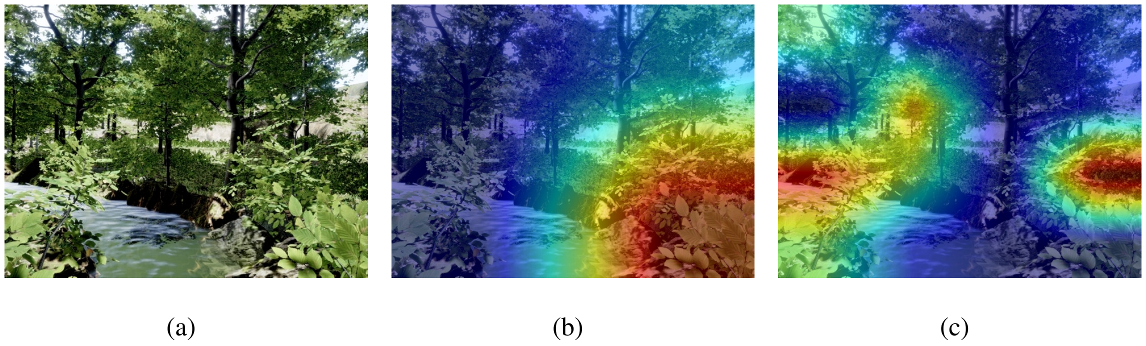


Figure 4.20: Visualization of the neural network decision by showing: (a) the input image (b) the VAE-based controller activation map, and (c) the end-to-end neural network controller activation map on the same input image.

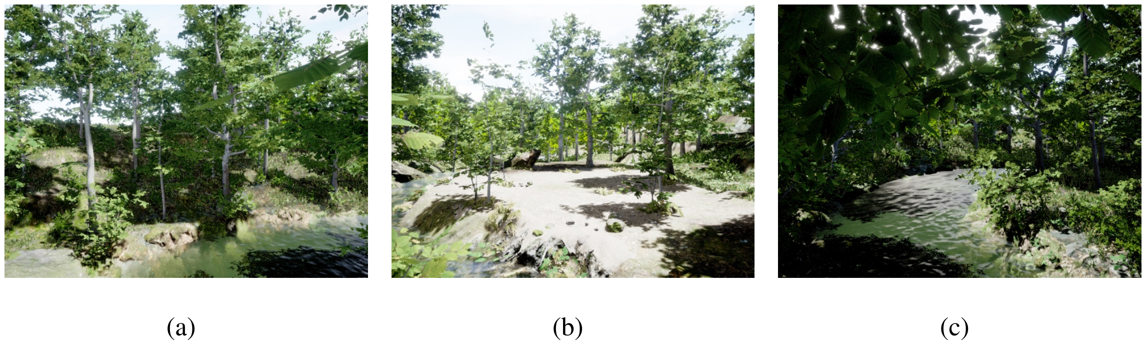


Figure 4.21: The typical failure locations (a) thin branches and leaves (b) no river in the field of view, and (c) shadow from trees.

learn more important information from the high-dimensional visual inputs, therefore making the training more efficient and the controller more effective. For supplementary video see: `https://www.youtube.com/watch?v=xQW2wcYjHus`.

To further investigate when the controller may lose its performance, the locations at which the drone commonly crashed in the simulation environments are plotted in Fig. 4.21. It is observed that all three controllers are likely to fail and lead to a crash when the drone is flying towards thin tree branches and leaves (see Fig. 4.21 (a)) which the visual algorithm may not be able to detect them. After a large control command when the UAV is off by the river path too much and there is no river in the field of view, the UAV may also generate unexpected behaviors (see Fig. 4.21 (b)). The shadow existing in the environment from the trees also confuses the control policy and leads to extensive or incorrect control commands (see Fig. 4.21 (c)). In addition, since the drone is flying at a constant forward speed, if the obstacle is right in front of the drone and too close, it's hard to provide safe navigation even for human operators, and this usually leads to crashes. All of these common failure locations will be considered in future work to improve the proposed vision-based navigation policies.

4.6 Summary

The chapter presents an imitation learning framework and compares several vision-based control policies for the task of UAV autonomous navigation in complex outdoor environments. Training the agent in simulation allows us to demonstrate the capability of the developed framework and avoid unnecessary losses before deploying it in the real world. An intervention-based DAgger framework is introduced to train a vision-based UAV capable of flying inside complex and GPS-denied riverine environments. The performance of a VAE-based controller with a linear regression-based controller and an end-to-end neural network controller trained with human demonstrations are compared in the simulation environments. The results show that the VAE-based controller outperforms the other two controller types in both training and testing processes and is able to navigate the UAV autonomously.

Chapter 5

Validation of Vision and Learning-based UAV Control in Real Orchard Environments

In Chapter 4, a variational autoencoder-based controller was developed and trained by an imitation learning approach. The performance was compared against several other vision-based algorithms for UAV navigation in riverine environments and the proposed VAE-based controller showed competitive results in simulations. The framework is validated and demonstrated in a real orchard environment (see Fig. 5.1) in this chapter. The control policy is trained through an interactive imitation learning framework from real flight data. The results show that the drone can navigate without GPS in this complex environment and avoid obstacles (e.g., tree branches) autonomously during the flight. The controller can fly the drone a longer distance with less human assistance compared to the existing baseline algorithms and has a good generalization capability.

5.1 Introduction

Autonomous navigation of unmanned aerial vehicles (UAVs) has made significant progress in recent years. However, most UAVs still rely on GPS to navigate for applications such as remote sensing [155], precision agriculture [156], pest management [6], and bridge inspection [157]. Thus, UAVs lack the autonomous capability to fly in confined and GPS-deprived environments.

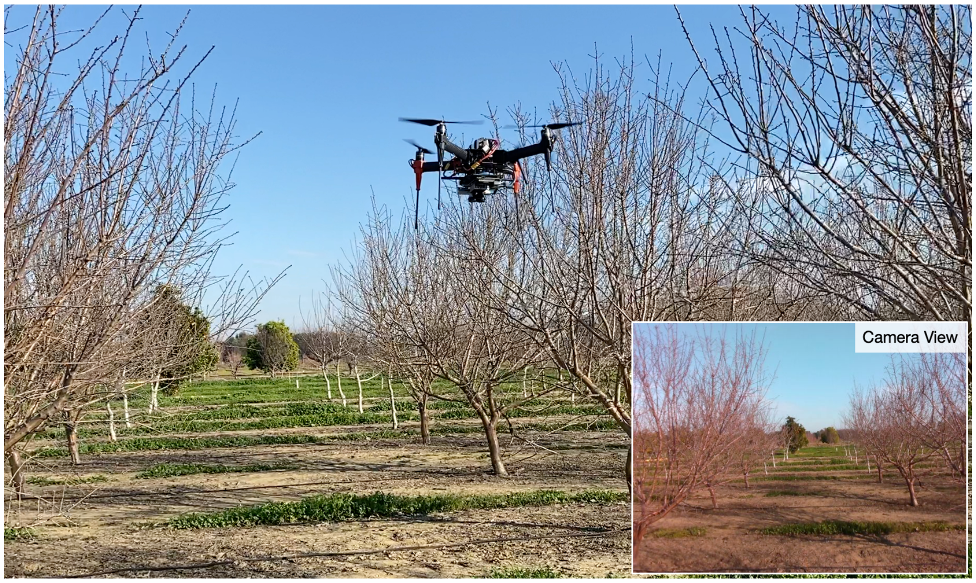


Figure 5.1: The drone is flying autonomously in this orchard without GPS and knows how to avoid the tree branches along the way. The view of onboard camera is provided.

Besides, human pilots are frequently queried to provide avoidance maneuvers in the loop, which heavily increases the workload. Autonomous navigation in complex and unknown environments especially when the GPS signal is weak poses several challenging problems and largely hinders the deployment of UAVs.

When GPS becomes unreliable or unavailable, the UAV needs to rely on exteroceptive sensors to sense the environment and navigate. Advanced techniques to enable UAV autonomous operations without GPS include: 1) lidar-based [158], and 2) vision-based [159] approaches. Typical lidar sensors are too large to fit on lightweight UAVs and can significantly reduce the flight time when size and endurance are considerations. It leaves the vision-based algorithm a better option for UAV autonomous navigation in complex environments where a compact size and longer battery life are pursued. Also, a camera usually costs less than a lidar sensor.

Reinforcement learning (RL) has produced promising results in the field of self-driving car [160] and strategy game playing [161] recently. However, one key drawback of RL is its sample inefficiency. RL requires a substantial amount of data to achieve good results. Also, designing an effective reward function is tricky. Additionally, its trial-and-error nature hinders its use in safety-critical systems since one failure may lead to enormous losses. One appealing workaround to the above-mentioned shortcomings is to train a policy in simulation first and transfer it to the real world during deployment [162]. This transfer learning strategy can reduce the costs and risks to train a policy without worrying about damaging the system. However, an inaccurate model of the system in simulation (e.g., modeling of environmental changes can be difficult) may lead to unmatched performance on real robots, and minimizing the sim-to-real gap remains an open question. In this work, an imitation learning (IL) framework [113] is adopted to train a visuomotor policy. The robot learns a strategy by mimicking an expert's behavior from a small number of demonstrations. The aim is to utilize human knowledge to guide the learning process and achieve human-like behaviors while overcoming the limitations of the aforementioned methods. IL has been proven successful in training useful policies on real robots [66, 68].

5.2 Related Work

Autonomous navigation in complex and unknown environments is challenging. State-of-the-act technologies divide autonomous navigation into perception, planning, and control problems [163, 164, 165]. Car and lane detection [166, 167] are standard perception tasks to enable autonomous driving. Semantic segmentation [168] helps to label the objects and learn representations of the environment. For orchard environments in particular, 3D point cloud data can be used to localize the robot in the row [169]. [170] proposed a novel template-based method and achieved a better localization performance with respect to the row centerlines. For general visual navigation in GPS-denied environments, visual-inertial odometry [109] and simultaneous localization and mapping [110] are commonly used algorithms to localize the agent and build a map of the unknown environment, which provide essential information for the trajectory planning stage.

A collision-free path or trajectory can be planned after retaining a global or local map of the environment. Widely used planning algorithms include: 1) graph-search-based approaches, for instance, A* [171] and its variants [172], and 2) sampling-based approaches, for instance, probabilistic roadmap [173], rapidly exploring random tree [174] and its variants [172, 175]. An energy-optimal [115, 176] or time-optimal trajectory [177] may also be designed to satisfy additional requirements. After that, a tracking algorithm is targeted to control the robot to follow the planned path or trajectory. Common path tracking algorithms include pure pursuit [178], PID control [179], stanley control [180], and model predictive control [181]. Under this framework, [164] achieved fast drone flights in GPS-denied and cluttered environments. [182] developed vision-based aerial system to monitor orchards with path planning.

Though this modular design paradigm has achieved some success, the separation of perception and control may cause unexpected behaviors [69]. The authors in [59] also pointed out that there is a discrepancy between the trajectory planning and the deployment stage. The requirements to build a 3D map of the environment [158], search feasible collision-free paths, and solve constrained optimal control problems can add extra cost and delay to the real-time system [124]. Conversely, human pilots rely on nothing more than a camera stream to navigate a robot in complex and unknown environments. They can provide outstanding reactive behaviors and generalizable skills in a wide range of scenarios [66, 69]. A similar visuomotor policy which directly maps states to action can be designed. The policy can be represented by a deep neural network [183] that is trained end-to-end using different learning-based approaches.

Among these approaches, reinforcement learning finds the optimal policy by maximizing a reward function. RL has been successfully applied in a wide range of applications. For example, [60] used a deep Q-network to train an agent playing Atari games based on pixel inputs. [61] trained a deep neural network with proximal policy optimization (PPO) to fly drones across race tracks at fast speed. Recently, [62] solved the autonomous car racing problem using a soft actor-critic (SAC) algorithm and the policy achieved super-human performance. [63] successfully trained quadruped locomotion in real world with deep reinforcement learning. Nevertheless, RL is notorious for its bad sample complexity. Generally, it requires a substantial amount of data to obtain good results and the training may converge slowly. The trial-and-error learning process also causes severe problems for safety-critical systems.

Other researchers attempt to train and deploy the model in different domains with a variety of techniques. [184] trained a racing drone policy in simulation through domain randomization and deployed it on a physical platform without fine-tuning. Domain adaptation [185, 186] can be applied to narrow the gap between simulation and reality. [187] used real-world data to adapt the model learned in simulation and achieved a better generalization capability. However, transferring a model learned in simulation to the real world is still a challenging problem.

Imitation learning-based approaches, on the other hand, learn a policy by imitating expert behaviors directly and without a need for a hand-crafted reward function. The supervision from expert demonstrations greatly accelerates the learning process and makes IL more data efficient. The effectiveness of the learned policy has been demonstrated in solving many autonomous driving tasks. For example, [188] trained a car to follow a road with supervised learning. [113] replaced the policy with a convolutional neural network and improved the self-driving performance. A visual affordance can be added as guided auxiliary supervision to obtain a higher success rate [189]. Accomplishing UAV navigation in complex and GPS-denied environments using supervised information has also been explored by many researchers. For instance, [66] trained a drone to fly inside low-altitude forests and perform tree avoidance with a data aggregation approach. By representing the policy with neural networks, [68] used imitation learning to train a UAV to traverse forest trails based on monocular visual perception of trees and foliage. [190] taught a quadrotor how to fly indoors and avoid obstacles from a recorded crash dataset. [69] learned a drone navigation policy in urban environments from the data collected by cars and bicycles. The policy has a good generalization capability in both indoor and outdoor environments. Learning a more agile behavior in challenging environments with IL has also been studied. For example, [191] learned an agile drone flight policy and demonstrated it in dynamic environments. [192] achieved agile and off-road autonomous driving using end-to-end imitation learning.

Recently, researchers found out that privileged information [193] can be used to improve performance on top of traditional deep learning methods. The training can be split into two steps [194]. A teacher policy who has access to privileged information is trained first. Next, a student policy that emulates the teacher's policy gets trained without knowing the privileged information. The student policy is then deployed in the real world. This paradigm has proven successful in achieving vision-based urban driving in simulation [194], performing high-speed flight of drones in the wild [195], and learning quadruped locomotion over challenging environments [196]. However, the selection criteria of privileged information are quite vague and the performance is not guaranteed without training in a high-fidelity simulator [197, 198]. The simulator is expected to accurately model the physical dynamics as well as environment changes (e.g., photo-realistic rendering) in order to properly deploy the trained policy in the real world, and this certainly demands a lot of engineering work.

5.3 Methodology

In this section, an intervention-based imitation learning framework is introduced with a discussion about its advantages. Next, the neural network architecture in the VAE-based controller is presented. Finally, the data collection procedure in the real world is described.

5.3.1 Intervention-based Learning

An imitation learning approach is used to train a vision-based control policy in this work. As discussed above, imitation learning has several advantages over reinforcement learning since it reduces the sample complexity and eliminates the need for a hand-crafted reward function. A common strategy of imitation learning is to learn an agent policy π_θ parameterized by θ that mimics the expert policy π_E. The optimal policy π_θ^* is found by satisfying (5.1) where $\mathbb{D}$ is a similarity measurement between the policies.

$$\pi_\theta^* = \arg\min_{\pi_\theta} \mathbb{D}(\pi_\theta, \pi_E) \tag{5.1}$$

The simplest approach in IL formulates the problem as a supervised learning process with the objective of matching the learned policy π_θ to the expert policy π_E from a dataset of expert demonstrations $\mathscr{D}$, namely, behavior cloning [188]. However, behavior cloning typically fails in most applications because the distribution of states encountered when executing the learned policy π_θ may not match the distribution found in the demonstrated dataset. To overcome this limitation, [142] proposed a data aggregation (DAgger) method with a queryable expert to provide new demonstrations in the loop. However, there are two issues with the DAgger algorithm. First, the querying step is very inefficient because the expert needs to explicitly relabel all new data points. Second, when evaluating the agent policy rollout, the human expert lacks direct feedback from the robot and therefore is unaware of how the correction affects the future state.

In this work, an intervention-based DAgger approach is used to train the agent policy. A human pilot is in charge of intervening when the agent drifts away from the desired behavior.

Instead of querying an expert offline, the expert is queried online when the drone is executing the agent policy in the real world. During the rollout, the human pilot has the authority to take over the control from the agent when they think the agent's action may lead to a failure or an unsafe state. Only the demonstrations that the pilot is taking over control will now be appended to the training dataset. After the drone has reached a safe state which is defined by the human pilot, the pilot can return the control back to the agent. The policy is re-trained after each iteration as the normal DAgger conducts. Because human expert only provides feedback intermittently during robot execution, this largely simplifies the querying procedure. Also, the robot's behavior can improve online since it's based upon an iterative approach. Additionally, the expert has uninterrupted control of the robot with feedback during demonstrations. This way of interaction is more natural in the real world and will benefit the collected data quality. A pseudo-code of the intervention-based learning algorithm is provided.

5.3.2 Network Architecture

The proposed VAE-based controller is composed of two components, a variational autoencoder network followed by a policy network. Each one is explained in detail in this section.

5.3.2.1 Variational Autoencoder Network

A variational autoencoder network [150] is introduced to the control policy. In the previous chapter, the network was shown to have a good capability to extract useful information from high-dimensional spaces into low-dimensional spaces and benefit both training and generalization in a vision-based navigation task. A VAE network is normally composed of two components. The first component is an Encoder network which compresses image data from a high-dimensional space to a low-dimensional space. The extracted result, also called latent variables, normally represents some dominant aspects of the original images (e.g., position, scale, rotation, lighting). The latent variables are then fed into a Decoder network to reconstruct the image. The loss function in (5.2) is minimized to train the VAE network. The first term in the loss function is the reconstruction error between the original image x and the reconstructed image $\hat{x}$ calculated by the decoder network $p_{\varphi}(\hat{x}|z)$, where z is the latent variable and the function is parameterized by φ. The second term is the Kullback–Leibler (KL) divergence between the extracted latent variables $q_{\psi}(z|x)$ from the encoder network parameterized by ψ and a series

```
Algorithm 1 Intervention-based Learning
Input: maximum iteration number $i_{max}$
Output: optimal agent policy $\pi_\theta^*$
Initialize: $i = 0, \mathscr{D} = []$
if $i = 0$ then
    append all human demonstrations to $\mathscr{D}$
    train agent policy $\pi_{\theta_0}$ on $\mathscr{D}$
else
    for $i = 1 : i_{max}$ do
        roll out agent policy $\pi_{\theta_{i-1}}$ with human interventions
        get a series of rollout trajectories $\mathscr{T}_j$
        for each $(s_t, a_t) \in \mathscr{T}_j$ do
            if $(s_t, a_t) \in \pi_E$ then
                append $(s_t, a_t)$ to $\mathscr{D}$
        re-train agent policy $\pi_{\theta_i}$ on current $\mathscr{D}$
$\pi_\theta^* \leftarrow \pi_{\theta_i}$
return $\pi_\theta^*$
```

of distributions $p(z)$ which are Gaussian distributed. A reparameterization trick is used in order to backpropagate the gradient properly. β is a weight factor and is selected to be equal to three to balance the penalty on the reconstruction and KL divergence.

$$\mathscr{L}_{VAE}(\varphi, \psi) = \|x - p_\varphi(\hat{x}|z)\|^2 - \beta \cdot D_{KL}(q_\psi(z|x), p(z)) \tag{5.2}$$

The structure of the VAE is illustrated in Fig. 5.2, which consists of five convolutional layers with LeakyReLu activations in encoder network and five transpose convolutional layers with ReLu activations in decoder network. The encoder network takes an image in the size of 128x128x3 and generates the latent variable which has a dimension of 256. The decoder network reconstructs the image in the same size. An example pair of the original image and the reconstructed image is also provided in Fig. 5.2 to visualize the reconstruction performance. During the deployment, only the trained encoder network is used to compute the latent variable

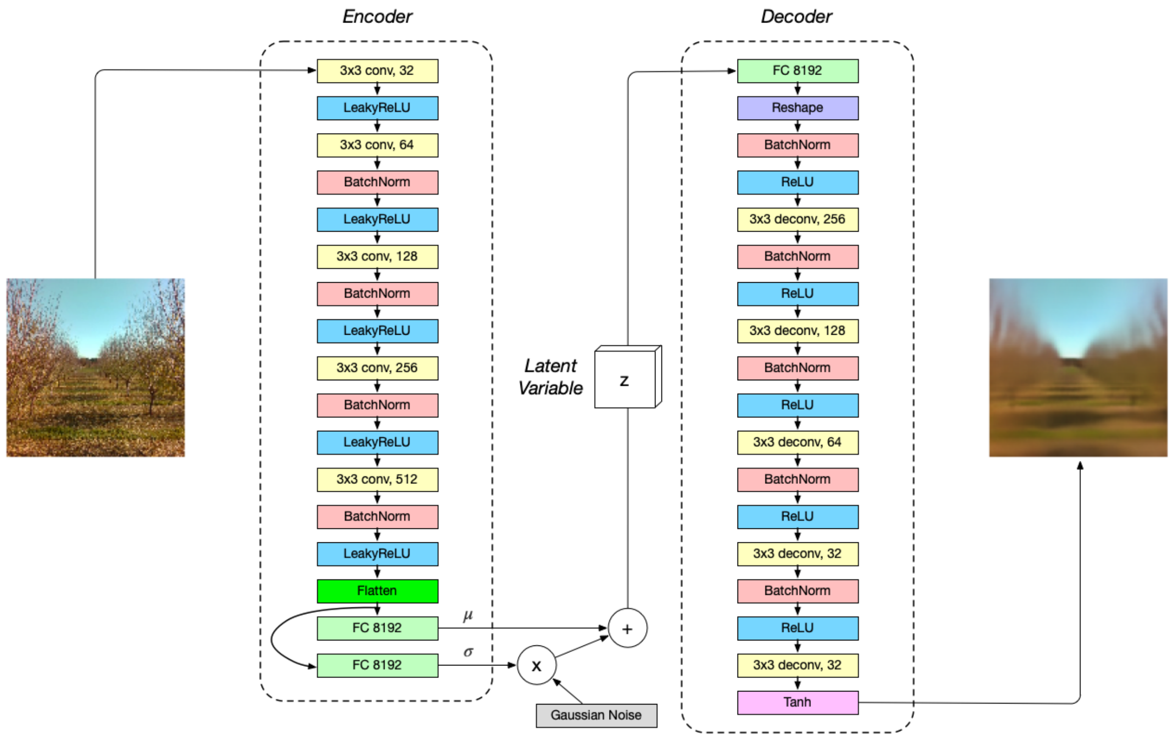


Figure 5.2: The structure of the variational autoencoder network.

(the network weight is frozen) and then fed to a policy network which is introduced in the next section.

5.3.2.2 Policy Network

The policy network is built upon a multi-layer perceptron (MLP) with two hidden layers. Each hidden layer has a dimension of 256 with ReLU activation. The output layer has a hyperbolic tangent activation function to generate the control command in the range of $[-1,1]$. The input to the policy network is the latent vector extracted by the VAE encoder concatenated with the current drone state including the attitude and velocity from the EKF estimator. The network is trained by minimizing the mean-square-error (MSE) between the agent policy output $\pi_\theta(a|s)$ and human expert policy output $\pi_E(a|s)$ in (5.3) on the demonstrated dataset $\mathscr{D}$.

$$\mathscr{L}_{policy}(\theta) = \|\pi_\theta(a|s) - \pi_E(a|s)\|^2, s \in \mathscr{D} \tag{5.3}$$

5.3.3 Data Collection in Real World

The experiments were conducted in an orchard environment near Davis, California. To train the VAE model, six rows were selected from the orchard, so that each row has mixed tree

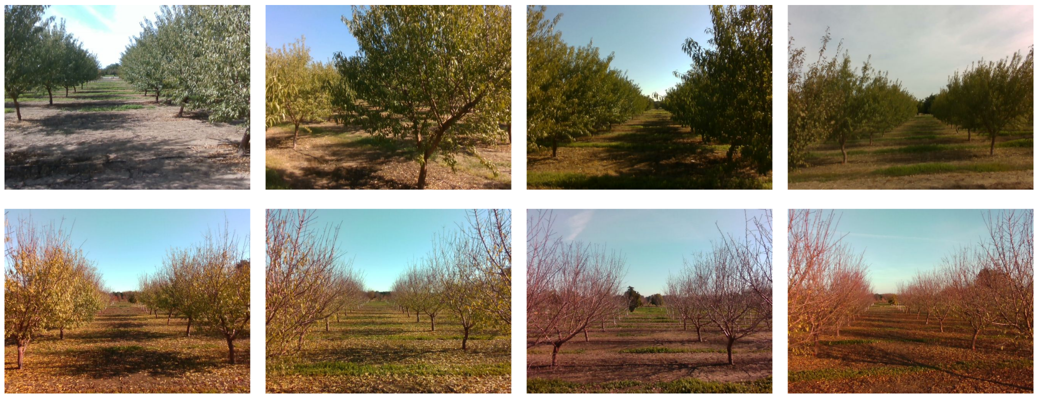


Figure 5.3: Sample image data collected in a variety of seasons, weathers, and times in a day from the field.

species and different appearances. Image data were collected from these six rows in a variety of seasons (i.e., summer, autumn, winter), weather conditions (i.e., sunny, cloudy), and times in a day (i.e., morning, noon, afternoon). A selection of image data taken from the field is provided in Fig. 5.3. The VAE dataset has 110k images in total and was used to train the VAE network. Standard data augmentation was performed which includes changing the brightness, contrast, saturation and sharpness of the original dataset. Random horizontal flipping was also applied to the images. . After the VAE network was trained, the encoder network was taken out (its weight was frozen) and used to generate the latent variables in order to train the policy network.

The policy network was trained based on the Algorithm 1 discussed in Section 5.3.1. All human demonstrations were collected in the real world from the same orchard rows where the VAE data was recorded. Due to the multi-model behavior of humans, it's infeasible to achieve a good policy if the human pilot is granted full control over the drone flying in this environment. For example, the pilot can fly at a high altitude to avoid all trees or command the drone to fly slowly and provide a very conservative response. This also makes the training hard since the network is learning from a very large and sparse space. Therefore, in this work, the pilot is only allowed to provide yaw rate correction while a high-level controller is implemented to keep the drone flying at a constant height above the ground and at a constant forward speed in the body-fixed frame. It's worth mentioning that, due to the imperfect balance of the drone and the environmental disturbance (i.e., wind gusts), the drone will not always fly in a straight line and may have a side speed. Navigating in this environment is a challenging problem due to unstructured obstacles (e.g., tree branches) and unpredictable disturbances. Continuous yaw correction is needed to put the drone back in safe regions when it starts drifting away, thus rendering navigation inside orchard rows a suitable scenario to test the proposed vision-based algorithm.

In the experiment, the height was fixed in a range of 1.6-2.0 m to avoid the ground effect [102] and to ensure visible obstacles in the view. The drone was commanded to fly forward at a speed of 0.6 m/s to allow the pilot to apply effective demonstrations. The max yaw angular speed was limited to 45 deg/s to avoid large state estimation error because the camera used in the experiments (see Section 5.4.1) tends to lose tracking under rapid yaw rotations. Ideally, the human pilot should share the same view as the agent from the onboard camera. However, in practice, it's challenging to achieve due to a couple of restrictions. Streaming images back in real-time to the laptop is unstable, as it is often interrupted because of the limited Wi-Fi range. The trees in the field block the signal and further degrade the Wi-Fi performance. Also, wireless data transmission adds extra time delay to the system which is hard to quantify. Therefore, in this work, the human pilot was allowed to have full state information about the environment. It is observed that the agent can still learn a good policy and the results are presented in the next section.

5.4 Experimental Results

In this section, a description of the experimental platform is presented first. Then the results of the vision-based algorithm are shown in both training and generalization environments and compared against a set of baseline algorithms from the literature.

5.4.1 Experimental Platform

A custom quadcopter platform was built to validate the proposed vision-based algorithm. The main frame has a wheelbase of 450 mm. An Intel Realsense D435i camera is mounted forward and streams RGB images with a field of view of 70°. The visual odometry is provided by an Intel Realsense T265 tracking camera. The altitude measurement is found not accurate from the vision sensor, therefore a Lightware SF11/C laser rangefinder is added to get the relative height above ground. PixRacer R15 is chosen as the flight unit with the PX4 autopilot stack as the low-level controller. The onboard computational unit is an NVIDIA Jetson Xavier NX. An external SSD is mounted to store the data. A telemetry radio is also added to get real-time telemetry status from the drone. A 4S Lipo battery is used to power the entire system and can support a flight time of 13 minutes with a take-off weight of 1.8 kg. Pictures of the platform can be seen in Fig. 5.4.

Fig. 5.5 shows the architecture of the system. The RGB image is streamed from the D435i camera at a rate of 30 Hz. The t265 camera is selected here because it has an internal visual inertial-odometry (VIO) pipeline. Other VIO methods [199, 200] can also be used at the cost of extra computational load to the onboard computer. The visual odometry, altitude, and IMU measurements are fused to an Extended Kalman Filter (EKF) and perform state estimation. A high-level controller generates the necessary commands to fly the drone at a constant height and speed. Meanwhile, the VAE-based controller provides yaw rate commands at 30 Hz for navigation in the field. The state estimation, neural network inference, and high-level control are performed on the Xavier NX. The low-level flight control is handled by the PX4 and outputs

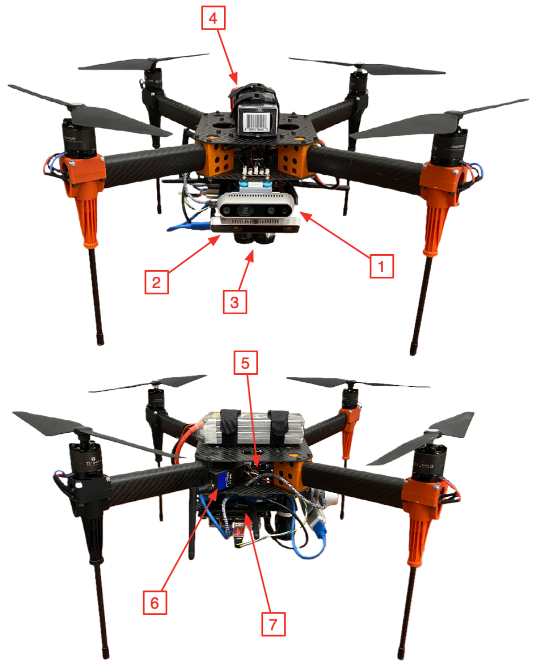


Figure 5.4: The custom-built quadcopter platform equipped with 1) Intel® Realsense D435i camera; 2) Intel® Realsense T265 camera; 3) Lightware SF11/C laser rangefinder; 4) Tattu 4s 5200mhA LiPo battery; 5) Pixracer R15 flight controller; 6) mRo Ski Telemetry radio; and 7) NVIDIA Jetson Xavier NX.

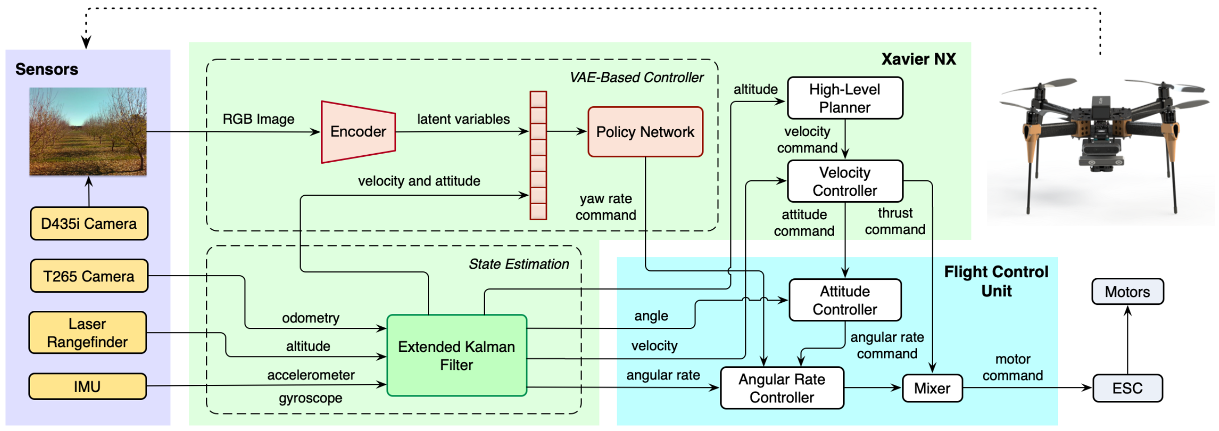


Figure 5.5: System architecture diagram.

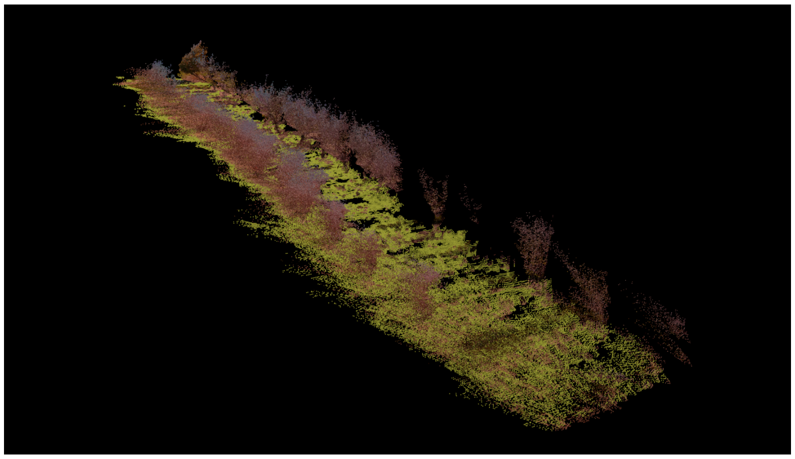


Figure 5.6: A filtered point cloud generated from the experiment data.

motor signals. The flight control unit communicates with Xavier NX through a Robot Operating System (ROS) interface. The onboard computer is connected to a laptop through Wi-Fi, thus allowing the users to start the scripts and monitor the onboard status. Human pilots can teleoperate the drone with a remote transmitter to provide demonstrations for training and take over the control during any emergency.

The training was done on a lab computer with AMD Ryzen 9 3900X CPU, 64GB memory, NVIDIA Geoforce RTX 2080Ti GPU. ADAM was used as the optimizer to minimize both VAE loss and MLP loss.

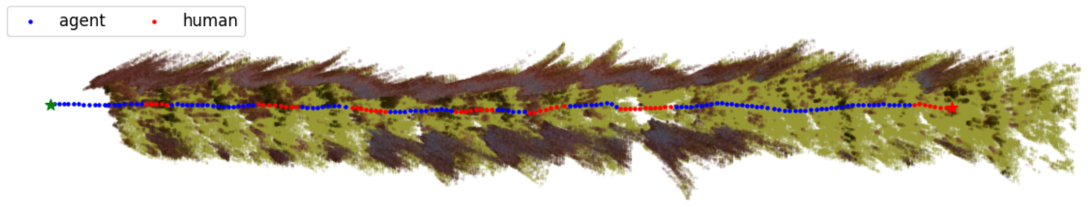


Figure 5.7: Top-down view of the trajectory performed by the VAE-based controller after one iteration of training.

5.4.2 Qualitative Results

The qualitative results of the proposed algorithm are shown first. The drone was flown in the selected six rows mentioned above and the VAE-based controller was trained through the intervention-based learning method discussed in Section 5.3.1. To visualize the evolution of the agent policy performance with the training iteration, an open-source SLAM library [201] was used to plot the drone trajectory in a reconstructed 3D map offline. Since the D435i camera has a stereo setup and can generate depth images, the depth and RGB images were fused together with the drone localizations into the SLAM library and generated a point cloud map of the environment. The raw point cloud was filtered to remove outliers and unrelated points (e.g., sky). The ground was identified through the RANSAC algorithm [202] and re-colored for better visualization. A sample image of the filtered point cloud can be seen in Fig. 5.6. A top-down view of the drone trajectory in the filtered point cloud map executing the policy after one iteration of training is provided in Fig. 5.7. The trajectory controlled by the agent policy is plotted in blue and the human intervention portion is plotted in red. As noticed, the agent policy learned some skills after one iteration of training but still needed human assistance sometimes to avoid obstacles and navigate in the field. Based on the experiment, it is found that the performance improved significantly after three iterations of training. A sample trajectory in the 3D map after three iterations of training is depicted in Fig. 5.8. It can be clearly seen that the agent successfully learned how to avoid obstacles and achieved a fully autonomous flight without human intervention.

To evaluate the generalization performance of the vision-based algorithm in novel environments, two extra rows were selected from the orchard which the agent had never seen before during the training. An agent that has been trained to perform well in the training environments

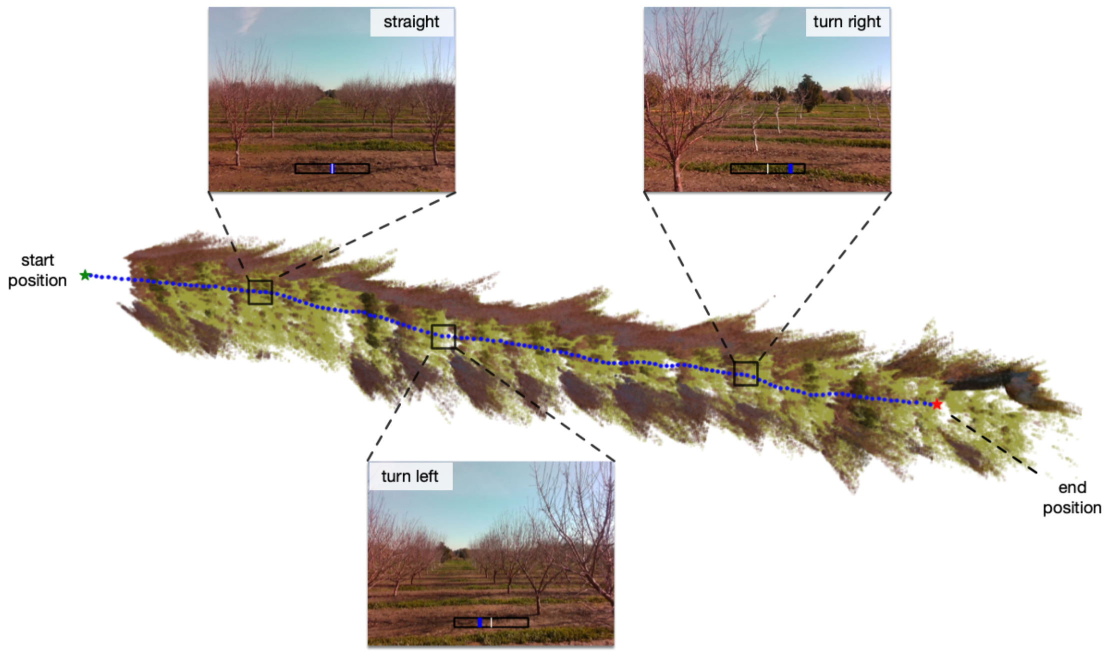


Figure 5.8: Top-down view of the trajectory performed by the VAE-based controller after three iterations of training. The views from the onboard camera at different locations are provided.

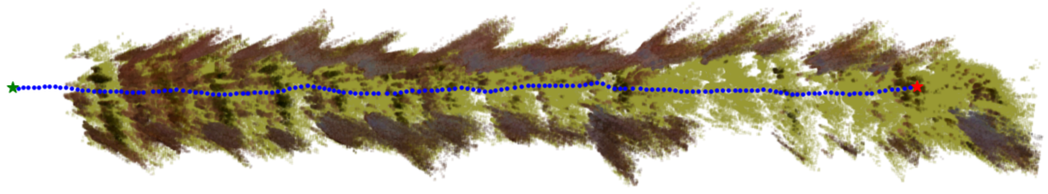


Figure 5.9: Top-down view of the trajectory performed by the VAE-based controller in a generalization environment.

is not inherently guaranteed to perform well in a novel environment. There is a necessity to see its generalization ability. The VAE-based controller trained after three iterations was deployed in these two additional rows. The results confirmed that the agent could fly pretty well in these novel environments with little human input. A fully autonomous trajectory executing the VAE-based control in the novel environment is plotted in Fig. 5.9. For supplementary video see: `https://www.youtube.com/watch?v=VPlTaJbdo7U`

5.4.3 Quantitative Results

To provide quantitative results and investigate the advantages of the VAE-based controller, the controller was compared with two baseline algorithms from the literature. 1) The first one

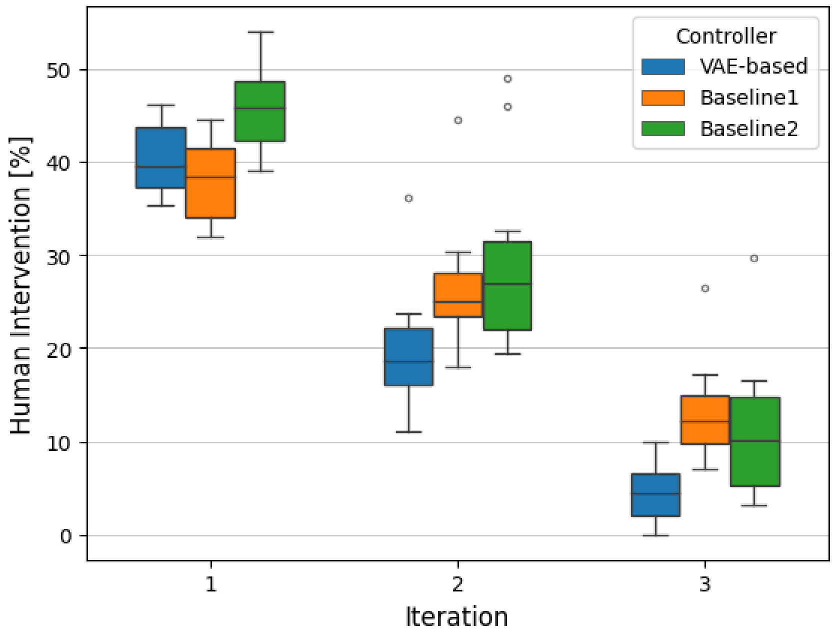


Figure 5.10: The box plot of human intervention rate against training iterations for all three controllers.

is a non-neural-network-based controller [66] in which the control policy is trained based on human-selected visual features and extracted by linear regression. Call it *baseline1*. 2) The second baseline is a neural-network-based controller [69] which includes a compact convolutional neural network (CNN) to infer the control command and is called *baseline2*. Both controllers provide similar reactive behaviors and have demonstrated their performances in the real world. To make a fair comparison study, the original implementations were slightly modified, so that both controllers take an input image size of 128x128x3 and output a yaw rate command, similar to the proposed VAE-based controller. All three controllers were then trained using the intervention-based learning method and with the same number of iterations from the human pilot. To further reduce the bias from human awareness, the sequence of executing the controller was randomly decided in the experiment, therefore the pilot did not know which controller was being trained.

The human intervention rate during the training process was first calculated for all three controllers and the result is depicted in Fig. 5.10. Each controller was rolled out in the six training orchard rows for three iterations and the demonstration data was collected from the same human

Table 5.1: T-Test results for human intervention rate with a significance level $\alpha = 10\%$.

	VAE-based vs. Baseline1			VAE-based vs. Baseline2		
T-Test	iteration 1	iteration 2	iteration 3	iteration 1	iteration 2	iteration 3
t-score	1.3125	3.3187	3.7547	2.8887	3.1981	1.9527
p-value	0.2058	0.00507	0.00274	0.00941	0.00644	0.07458

pilot. The box plot shows the mean and distribution of the human intervention rate with a 95% confidence interval at different iterations. One can see that the intervention rate decreases as the training iteration grows for all three controllers. This proves the effectiveness of the proposed learning framework. Among the three controllers, it is observed that the VAE-based controller has the lowest averaged intervention rate across all iterations except the first iteration, in which baseline1 has a slightly lower averaged intervention rate compared to the other two. After three iterations of training, it is found that the VAE-based controller can keep the intervention rate below 10% while the other controllers have higher intervention rates and therefore require more human assistance. To verify the hypothesis rigorously, a t-test was performed to test the significance of the experiment result. The null hypothesis for the t-test is that the human intervention rate between the VAE-based controller and the two baselines has no significant difference. The t-test results are provided in Table 5.1. A significance level $\alpha = 10\%$ was selected and a p-value smaller than α means that the null hypothesis can be rejected. Based on the calculated p-values and the mean values, it can be concluded that the proposed VAE-based controller outperforms both baseline controllers with significance.

Next, the fully-trained models over 3 DAgger iterations were taken and used to conduct additional experiments to evaluate their flight performances in both training and generalization environments. Here, the baseline2 model with its pre-trained weights from the original paper [69] was added into comparison with slight modification on its output, and call it *baseline2 (pre-trained)*. For the evaluation metric, the average distance flow by the drone before a failure was selected; a failure in this case means requiring human intervention. To assess the performances, the experiments were conducted in two selected training environments and two generalization environments, which the agent had never seen before. The experiments were carried out in various weather conditions (i.e., sunny, cloudy, windy), and at different times of the day (i.e., morning, afternoon). All controllers, including the VAE-based one, were executed in the field, resulting in ten flights in both training and generalization environments for each controller type. The flight data was recorded and used to determine the average distance traveled by the agent before a failure, as shown in Figure 5.11. As expected, the performances drop when deploying the policies in environments and scenes that the controllers had never seen before. Based on the results, it can be seen that the proposed VAE-based controller achieves better performance compared to all baselines and can fly a longer distance before human intervention is needed. The conclusion remains true for both training and test environments, a manifestation of the good generalization capability of the proposed controller. Actually, during the flight experiments, it was noticed that the VAE-based controller could navigate the drone autonomously most of the time, while the other baseline algorithms required extensive human assistance. A discussion on why the VAE-based controller performs better is provided in Section 5.5. It is also observed that the pre-trained baseline2 did not perform well in the experiments, which may be due to the reason that it relies on line-like features to navigate and such kind of features are missing in this particular environment.

The processing time for each algorithm running on the embedded computer was calculated and presented in Table 5.2. The baseline2 (pre-trained) was dropped in this comparison since it shares a similar structure with baseline2. Note that all neural network parts were recompiled and accelerated by TensorRT on the device. The computer vision part of baseline1 also utilized onboard GPU to accelerate. From the table result, it's easy to see that both neural network-based methods (i.e., baseline2 and VAE-based) process the image and generate an inference output faster than the regression-based method in baseline1. The reason is that baseline1 needs to explicitly extract visual features such as optical flow from the raw image; hence the computation time is expected to be higher than the other two approaches. Baseline2 runs slightly faster than the proposed VAE-based controller due to its more compact network structure. Both of them can run on the onboard computer in real-time, although baseline1 intermittently triggered a timeout warning.

The controller performance was further evaluated under a velocity change. Considering

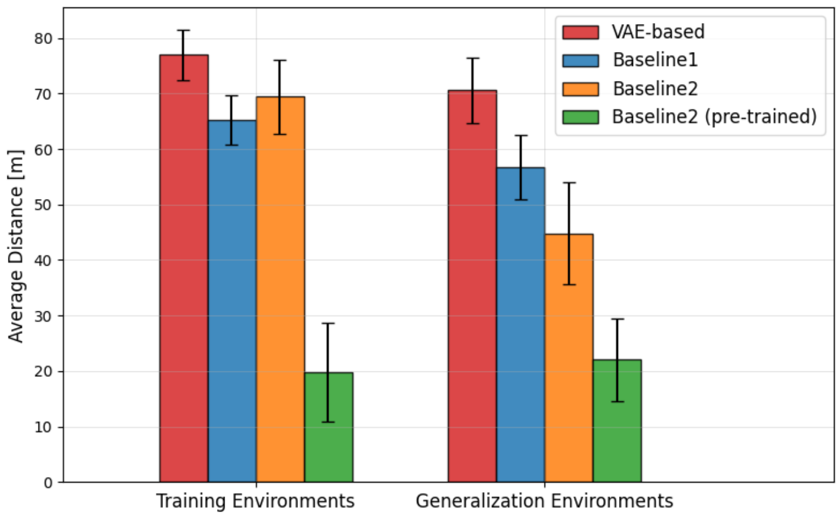


Figure 5.11: Average distance traveled before a failure for all controllers in training and generalization environments.

all the training was performed at the same forward speed, the forward speed was increased incrementally from 0.6 m/s to 0.8 m/s and 1.0 m/s in the generalization environments, and the performance was compared among all controllers. The baseline2 (pre-trained) was dropped here because its human intervention rate was too high even at the nominal speed. Increasing the speed to a much higher value is possible but gives the human pilot too little time to provide recovery maneuvers; therefore the speed was only tested up to 1.0 m/s. All models trained after three iterations were taken and executed to fly the drone in two generalization environments; with two flights in each environment. This provided four trails at each speed value and the results are shown in Fig. 5.12, which present the average distance flown by the drone without a failure against the speed changes. As expected, more failures occurred when the speed increased and, as a result, the distance traveled by the agent became shorter, and more human interventions

Table 5.2: The onboard processing time comparison.

	Processing time per frame (ms)
Baseline1	34.10 ± 4.13
Baseline2	2.43 ± 0.17
VAE-based	2.85 ± 0.19

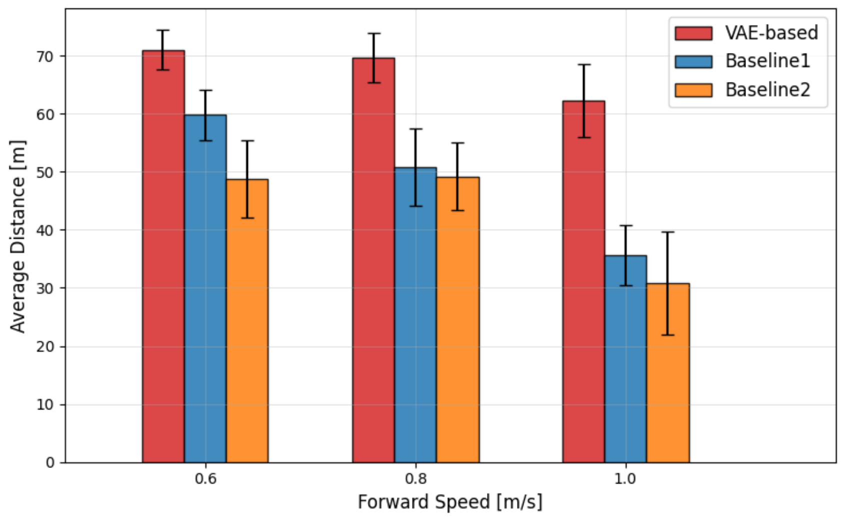


Figure 5.12: Average distance traveled before a failure under different forward speeds.

were required. However, it is found that the VAE-based controller still generalized well to a speed change and achieved a better result compared to the baselines. The results demonstrate that the proposed VAE-based is robust to speed changes and delivers good performance even though it has never experienced these speeds during the training.

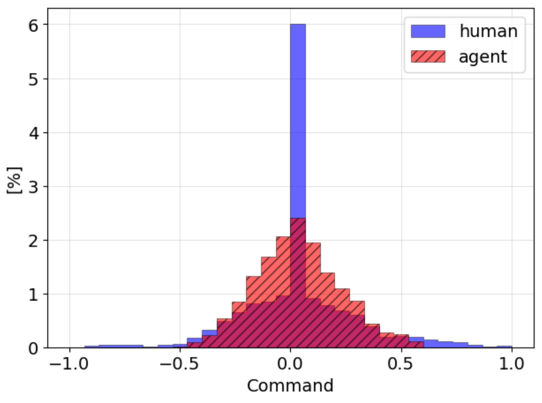
5.5 Discussion

The proposed framework does not require explicit perception-plan-control task decomposition and therefore saves a lot of resources and computational time. It provides a reactive visuomotor policy directly learned from human expert experience and can reliably control a drone from a single forward-looking camera. Based on the experiment results, the VAE-based controller can navigate the drone quite well and lead to no collisions after only three iterations of training. A discussion on why the VAE-based controller may outperform the existing baselines is provided as the following.

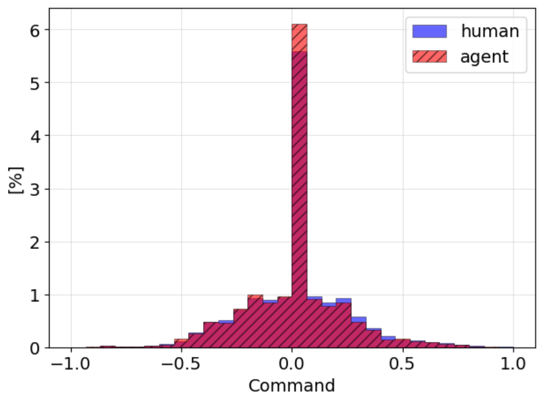
The histograms of human demonstration data distribution against the agent prediction are plotted in Fig. 5.13. The horizontal axis represents the normalized control command while the vertical axis shows the probability density of the human pilot commands against the model predicted commands on the training dataset. Regarding the behaviors of the baseline2 controller against the VAE controller, it is observed that the distribution of the baseline2 controller almost coincides with the human distribution, which indicates that the baseline2 controller overfits the training data. It explains why baseline2 controller does not generalize well to novel environments when the distributions become different. Regarding the performance of the baseline1 controller vs. the VAE-based controller, since a linear regression method is used to calculate the weight for baseline1 controller, the shape of the predicted control command looks more conservative and concentrates in the middle region compared to the other two neural network-based controllers. This hampers the baseline1 controller;s ability to generate aggressive (large) enough commands when the UAV needs a large yaw angular rate command in some maneuvers. The VAE-based controller, on the other hand, balances the distribution variance and the fitting performance, therefore outperforming the other two methods.

The proposed VAE-based controller can navigate the drone autonomously in the field most of the time and knows how to avoid obstacles correctly. However, the controller is definitely not perfect and it encounters some challenges when flying in certain locations. The main lessons learned during the experiments are discussed next. It's observed that the controller is more likely to fail when some trees are missing on the side. In that case, the orientation of the row path becomes confusing to the drone (see Fig. 5.14(a)). This happened to some of the experiments and a human-pilot takeover was required to pull the drone back in the right direction. A second case in which the performance dropped is when the drone would approach the end of the row (see Fig. 5.14(b)). The visual features for navigation are sparse at the row exit and the policy output tends to become unreliable. Currently, it was switched back to manual mode when the drone crossed the last pair of trees and landed it. These failure cases will be considered in future work to improve the proposed vision-based algorithm. A possible solution is to add a memory to the agent so that the generated heading command will not change too aggressively under these circumstances. A virtual end target can also be attached to the mission which allows the agent to exit the row properly.

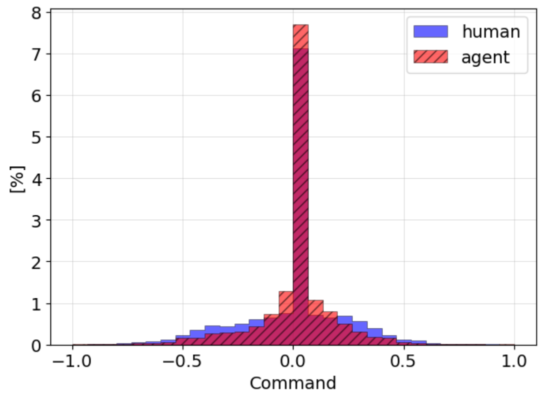
Another thing noticed is that ideally, the learning performance should keep improving as more data are collected from humans and the training datasets grows over multiple training iterations. However, in practice, the human pilot has difficulty in judging whether the decision made by the agent will lead to a collision or not, particularly at the iterations when the agent



(a)



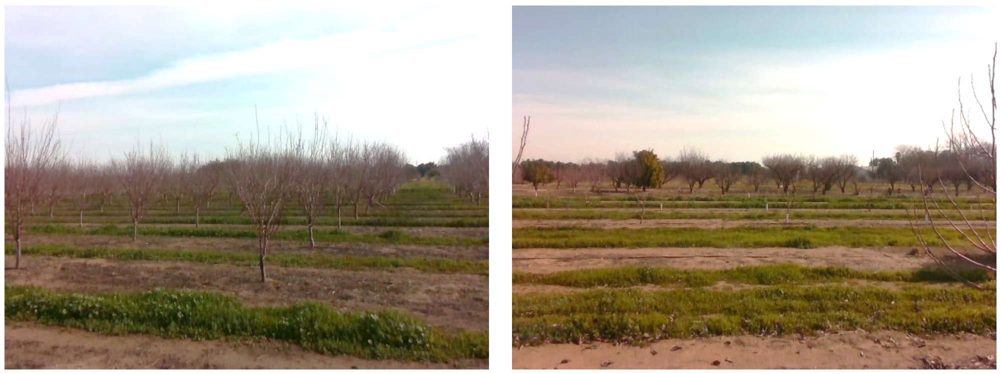
(b)



(c)

Figure 5.13: Histogram of human demonstration compared to the model prediction. The horizontal axis is the normalized control command and the vertical axis is the probability density. The model prediction comes from (a) baseline1 controller, (b) baseline2 controller, and (3) VAE-based controller.

policy has already learned some good skills. Besides, it was observed that the pilot tended to wait and see how the agent would perform, and not intervene until the agent was about to crash into an obstacle. The data collected from these locations is inconsistent with the majority distribution. As a result, re-training the policy with these data may degrade the agent performance and cause the learning process to become open-ended. Several experiments were conducted in the field and it was empirically found that the pilots are able to provide accurate and good performance in up to three iterations. Therefore, the training process terminated at three. There are certain challenges to be considered when a human teacher is in the learning loop because hu-



(a) (b)

Figure 5.14: Locations where the policy is more likely to fail (a) missing trees on the side; and (b) at the end of the row.

mans can be inconsistent and make mistakes. To improve consistency in human demonstrations and evaluations across multiple iterations, a computer-aided module can be added to determine when intervention is necessary.

Currently, the trained policy cannot be deployed 'zero-shot' to other platforms with different sizes and dynamics. This is because the human demonstrations are collected from a specific drone and the same control input may not be applicable to another drone with a different mass and inertia. One possible solution is to train a robust low-level controller with an adaptive module similar to the one presented in [203]. In this way, the controller can be transferred to other drones more easily during the deployment phase.

5.6 Summary

In this work, a vision-based navigation policy was presented which enables a UAV to fly in complex and GPS-denied environments autonomously. The policy is based upon a variational autoencoder neural network which extracts latent information from the image data of a front-mounted camera and generates a reactive yaw rate command. An intervention-based learning approach was adopted to train the control policy using human experience to guide the learning process. This approach involves human pilots intervening in the agent's policy in real-time to provide safe corrections during rollout. By doing so, they can efficiently provide new demonstrations and observe the impact of their feedback as the incremental learning process progresses. The performance of the navigation policy has been demonstrated in a real orchard environment. The results show that the proposed VAE-based controller can fly the UAV a longer distance with less human assistance requirements compared to the existing baselines and the controller generalizes well to novel environments and speed changes.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Several conclusions can be made in this dissertation:

In Chapter 2, the study examined the ground effect on small quadcopters and a model reference adaptive controller was developed to mitigate this effect in the control architecture. The experimental results reveal that the thrust generated by rotors to hover in the sky has a linear relation with the distance from the ground surface, which switches to a quadratic function when the separation distance between rotors is large enough. The proposed MRAC-based control architecture, which combines an LQR feedback + feedforward attitude controller with a PID + MRAC position controller, outperformed a traditional PID controller in terms of altitude tracking. Specifically, the adaptive controller reduced the rise time by at least 80% compared to a pure PID controller, enabling the vehicle to reach the target altitude during taking-off and landing in a timely manner. Furthermore, the study found that the MRAC with RBFs had better performance than the MRAC with a linear model, achieving a 45% smaller mean square error and a 15% shorter rise time. These findings suggest that the MRAC-based control architecture is an effective approach for mitigating the ground effect on small quadcopters and improving their altitude tracking performance.

Chapter 3 focused on identifying the wind effect on multirotor UAV dynamics near a hovering condition using a data-driven approach. In the flight experiment, the wind was directly measured by a wind sensor and treated as one of the control inputs into the linearized state-space model of an octocopter UAV. The coefficients in the state-space model were identified using a system identification approach in the frequency domain and their changes were compared under different wind speeds. The results regarding eigenvalues and dynamics modes show that when a multirotor UAV is hovering under a windy condition, either with a light or strong wind, its dynamics characteristics are significantly affected by the wind. The validation results in the time domain demonstrated that by explicitly considering the wind effect, the prediction error on the acceleration and angular velocities can be reduced by 15% on average. These findings offer unique insights into the effects of wind on multirotor UAV dynamics and can provide a valuable foundation for designing and developing model-based flight control systems. The approach can help improve the accuracy and reliability of multirotor UAVs in challenging outdoor environments and ultimately contribute to advancing the capabilities of autonomous aerial systems.

In Chapter 4, an imitation learning framework was presented and the comparisons were made among several vision-based control policies for the task of UAV autonomous navigation in complex outdoor environments. Training the agent in simulation allows us to demonstrate the capability of the developed framework and avoid unnecessary losses before deploying it in the real world. An intervention-based learning framework was introduced to train a vision-based UAV capable of flying inside complex and GPS-denied riverine environments in simulations. The performance of a VAE-based controller with a linear regression-based controller and an end-to-end neural network controller trained with human demonstrations were compared in the simulation environments. The results show that the VAE-based controller outperforms the other two controller types in both training and testing processes and is able to navigate the UAV autonomously.

In Chapter 5, the proposed vision-based policy was validated on a custom-built quadcopter platform and its performance was demonstrated in a real orchard environment. The policy was based upon a variational autoencoder neural network to extract latent information from the image data of a front-mounted camera and generate a reactive yaw rate command. The intervention-based learning approach was used to train the control policy from human experience and guide the learning process. Human pilots could transfer their knowledge to the learning agent through interactions and were able to observe the effect of their feedback throughout the incremental learning process. The results show that the developed VAE-based controller can fly the UAV a longer distance with less human assistance requirements compared to the existing baseline algorithms, and the controller generalizes well to novel environments and speed changes, enabling a UAV to fly in complex and GPS-denied environments more effectively and efficiently.

6.2 Future Work

In the study of ground effect, the main focus was on the altitude tracking performance of the multirotor, as this is the most significantly affected parameter. However, it's worth noting that ground effect can also affect longitudinal and lateral movement, especially when the UAV has a non-zero attitude angle. To overcome the model uncertainty, future research can consider adding model reference adaptive controllers to the longitudinal and lateral directions. Additionally, the attitude control of multirotor can be further improved by incorporating a full state feedback controller with attitude command error feedback terms.

This work provides an initial step towards the systematic understanding of wind effects on multirotor UAVs. To gain a comprehensive understanding, a multi-pronged methodology can be taken in future research. First, physical-based models can be combined with system identification results to analyze the effects of wind. For example, explicitly calculating blade flapping and induced drag from existing literature and comparing these results with flight experiment data can help identify discrepancies and improve existing models. Second, conducting forced oscillation wind tunnel experiments can benefit the identification process by systematically injecting winds with proper spectral content. A similar frequency swept wind can be generated in the wind tunnel to help identify the wind effects on multirotor UAVs more precisely.

For the vision-based control tasks, several future directions can be explored to further improve the system. One important direction is to incorporate low-level control and the UAV dynamics in the design process to make the learned policy more generalizable to different platforms. An adaptive module can be designed to transfer the policy to UAVs with different sizes and dynamics. In addition, the robustness of the controller can be improved by investigating common failure situations and incorporating the ability to handle dynamic environments with moving objects. Currently, the image data used to train the VAE network are collected by humans, which can be time-consuming and labor-intensive. To address this, a ground robot equipped with a camera can be used to collect data automatically. This will reduce the need for human labor and enable more efficient data collection. Another limitation of the current system is the need for human intervention when the agent displays unsafe behaviors. To overcome this, a recovery planning module can be designed to enable the agent to automatically recover from unsafe states and continue the mission without human assistance. Taken together, these future directions have the potential to enhance the value and significance of the research presented in this dissertation.

References

[1] A. M. Award. (2017, July) Want to fly a drone over a wildfire? don't, forest service says. here's wh. [Online]. Available: https://www.cpr.org/2017/07/14/want-to-fly-a-drone-over-a-wildfire-dont-forest-service-says-heres-why/

[2] C. H. Office. (2017, November) Researchers deploy autonomous drone to improve operations for mining industry. [Online]. Available: https://www.processonline.com.au/content/business/news/researchers-deploy-autonomous-drone-to-improve-operations-for-mining-industry/

[3] U. M. Corps, *Scouting and Patrolling*. Marine Corps Warfighting Publication, 2000.

[4] U. S. of American Department of Defense, "Us unmanned systems integrated roadmap fy2011-2036," 2017.

[5] J. Irizarry, M. Gheisari, and B. N. Walker, "Usability assessment of drone technology as safety inspection tools," *Journal of Information Technology in Construction (ITcon)*, vol. 17, no. 12, pp. 194–212, 2012.

[6] F. H. Iost Filho, W. B. Heldens, Z. Kong, and E. S. de Lange, "Drones: Innovative technology for use in precision pest management," *Journal of Economic Entomology*, vol. 113, no. 1, pp. 1–25, 2020.

[7] Y. Naidoo, R. Stopforth, and G. Bright, "Development of an uav for search & rescue applications," in *IEEE Africon'11*. IEEE, 2011, pp. 1–6.

[8] E. Natalizio, R. Surace, V. Loscri, F. Guerriero, and T. Melodia, "Filming sport events with mobile camera drones: Mathematical modeling and algorithms," 2012.

[9] K. P. Valavanis and G. J. Vachtsevanos, *Handbook of unmanned aerial vehicles*. Springer, 2015, vol. 1.

[10] D. Muoio. (2017, April) Uber partners with aurora flight sciences on flying taxis - business insider. [Online]. Available: https://www.businessinsider.com/uber-partners-with-aurora-flight-sciences-on-flying-taxis-2017-4

[11] A. Momont. (2014, October) Ambulance drone. [Online]. Available: https://www.tudelft.nl/en/ide/research/research-labs/applied-labs/ambulance-drone

[12] A. Gong, F. C. Sanders, R. A. Hess, and M. B. Tischler, "System identification and full flight-envelope model stitching of a package-delivery octocopter," in *AIAA Scitech 2019 Forum*, 2019, p. 1076.

[13] G. J. Leishman, *Principles of helicopter aerodynamics with CD extra*. Cambridge university press, 2006.

[14] P. Martin and E. Salaun, "The true role of accelerometer feedback in quadrotor control," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, May May 2010, pp. 1623–1629.

[15] P.-J. Bristeau, P. Martin, E. Salaun, and N. Petit, "The role of propeller aerodynamics in the model of a quadrotor UAV," in *2009 European Control Conference (ECC)*. IEEE, Aug Aug 2009, pp. 683–688.

[16] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *AIAA Guidance, Navigation and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, Aug Aug 2007.

[17] S. Yoon, P. V. Diaz, D. D. Boyd Jr, W. M. Chan, and C. R. Theodore, "Computational aerodynamic modeling of small quadcopter vehicles," in *American Helicopter Society 73rd Annual Forum*. Fort Worth, TX: Paper 73-2017-0015, May 2017.

[18] C. R. Russell, J. Jung, G. Willink, and B. Glasner, "Wind tunnel and hover performance test results for multicopter uas vehicles," in *American Helicopter Society 72nd Annual Forum*. West Palm Beach, FL: Paper 72-2016-374, May May 2016.

[19] M. B. Tischler and R. K. Remple, "Aircraft and rotorcraft system identification, second edition," *AIAA Education Series*, pp. 1–723, Apr 2012.

[20] J. Valasek and W. Chen, "Observer/kalman filter identification for online system identification of aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 2, pp. 347–353, 2003.

[21] J. Grauer and E. Morelli, "Method for real-time frequency response and uncertainty estimation," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 1, pp. 336–344, 2014.

[22] E. A. Morelli, "Flight test maneuvers for efficient aerodynamic modeling," *Journal of Aircraft*, vol. 49, no. 6, pp. 1857–1867, 2012.

[23] ——, "Real-time parameter estimation in the frequency domain," *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 5, pp. 812–818, 2000.

[24] B. Mettler, M. B. Tischler, and T. Kanade, "System identification of small-size unmanned helicopter dynamics," in *American Helicopter Society 55th Annual Forum*. Montreal, Quebec, Canada: Paper 55-00161, May 1999.

[25] W. Wei, K. Cohen, and M. B. Tischler, "System identification and controller optimization of a quadrotor uav," in *American Helicopter Society 71st Annual Forum*. Virginia Beach, VA: Paper 71-2015-049, May 2015.

[26] A. Gong, F. C. Sanders, R. A. Hess, and M. B. Tischler, "System identification and full flight-envelope model stitching of a package-delivery octocopter," in *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics, Jan Jan 2019.

[27] E. A. Fradenburgh, “The helicopter and the ground effect machine,” *Journal of the American Helicopter Society*, vol. 5, no. 4, pp. 24–33, 1960.

[28] J. S. Hayden, “The effect of the ground on helicopter hovering power required,” in *Proc. AHS 32nd Annual Forum*, 1976.

[29] D. P. Pulla, “A study of helicopter aerodynamics in ground effect,” Ph.D. dissertation, The Ohio State University, 2006.

[30] I. Cheeseman and W. Bennett, “The effect of ground on a helicopter rotor in forward flight,” Aeronautical Research Council, London, UK, Tech. Rep. 3021, September 1955.

[31] H. H. Heyson, *Ground effect for lifting rotors in forward flight*. National Aeronautics and Space Administration, 1960, vol. 234.

[32] H. Curtiss, M. Sun, W. Putman, and E. Hanker, “Rotor aerodynamics in ground effect at low advance ratios,” *Journal of the American Helicopter Society*, vol. 29, no. 1, pp. 48–55, 1984.

[33] B. Ganesh, N. Komerath, D. P. Pulla, and A. Conlisk, “Unsteady aerodynamics of rotorcraft in ground effect,” in *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 2005, p. 1407.

[34] N. Nathan and R. Green, “The flow around a model helicopter main rotor in ground effect,” *Experiments in fluids*, vol. 52, no. 1, pp. 151–166, 2012.

[35] C. Powers, D. Mellinger, A. Kushleyev, B. Kothmann, and V. Kumar, “Influence of aerodynamics and proximity effects in quadrotor flight,” in *Experimental robotics*. Springer, 2013, pp. 289–302.

[36] I. Sharf, M. Nahon, A. Harmat, W. Khan, M. Michini, N. Speal, M. Trentini, T. Tsadok, and T. Wang, “Ground effect experiments and model validation with draganflyer x8 rotorcraft,” in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2014, pp. 1158–1166.

[37] S. A. Conyers, M. J. Rutherford, and K. P. Valavanis, “An empirical evaluation of ground effect for small-scale rotorcraft,” in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1244–1250.

[38] L. E. Hale, M. Patil, and C. J. Roy, “Aerodynamic parameter identification and uncertainty quantification for small unmanned aircraft,” *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 3, pp. 680–691, 2017.

[39] H. Dahl and A. Faulkner, “Helicopter simulation in atmospheric turbulence,” in *Proceedings of the 4th European Rotorcraft Forum*, Stresa, Italy, Sep Sep 1978.

[40] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, “Precision flight control for a multi-vehicle quadrotor helicopter testbed,” *Control Engineering Practice*, vol. 19, no. 9, pp. 1023–1036, Sep 2011.

[41] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 20–32, Sep 2012.

[42] J. A. Lusardi, M. B. Tischler, C. L. Blanken, and S. J. Labows, "Empirically derived helicopter response model and control system requirements for flight in turbulence," *Journal of the American Helicopter Society*, vol. 49, no. 3, pp. 340–349, Jul 2004.

[43] J. Lusardi, "Control equivalent turbulence input model for the uh-60 helicopter," Ph.D. dissertation, Mechanical and Aeronautical Engineering Department, University of California, Davis, Davis, CA, 2004, proQuest Dissertations and Theses.

[44] O. Juhasz, M. J. Lopez, M. G. Berrios, T. Berger, and M. B. Tischler, "Turbulence modeling of a small quadrotor uas using system identification from flight data," in *7th American Helicopter Society Technical Meeting on VTOL Unmanned Aircraft Systems*. Mesa, AZ: Paper sm_uas_2017_017, Jan 2017.

[45] A. L. Salih, M. Moghavvemi, H. A. Mohamed, and K. S. Gaeid, "Flight pid controller design for a uav quadrotor," *Scientific research and essays*, vol. 5, no. 23, pp. 3660–3667, 2010.

[46] P. Castillo, R. Lozano, and A. Dzul, "Stabilization of a mini-rotorcraft having four rotors," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3. IEEE, 2004, pp. 2693–2698.

[47] J. Ferrin, R. Leishman, R. Beard, and T. McLain, "Differential flatness based control of a rotorcraft for aggressive maneuvers," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2011, pp. 2688–2693.

[48] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE Conference on Decision and Control (CDC)*. IEEE, 2010, pp. 5420–5425.

[49] S. A. Snell, D. F. Enns, and W. L. Garrard Jr, "Nonlinear inversion flight control for a supermaneuverable aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 15, no. 4, pp. 976–984, 1992.

[50] H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, "Model predictive control for micro aerial vehicles: A survey," in *2021 European Control Conference (ECC)*. IEEE, 2021, pp. 1556–1563.

[51] F. Fan, M. Lin, R. Ding, Z. Zheng, and Y. Liu, "Augmented-mrac for quadrotor uavs with parameter change," in *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2016, pp. 501–506.

[52] J. M. Selfridge and G. Tao, "Multivariable output feedback mrac for a quadrotor uav," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 492–499.

[53] Y. Zhang, J. Li, N. Xu, Y. Niu, and H. Zhu, “3d path following control for uavs using l1 adaptive method,” in *2015 Chinese Automation Congress (CAC)*. IEEE, 2015, pp. 1098–1104.

[54] I. Palunko and R. Fierro, “Adaptive control of a quadrotor with dynamic changes in the center of gravity,” *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 2626–2631, 2011.

[55] B. J. Emran, J. Dias, L. Seneviratne, and G. Cai, “Robust adaptive control design for quadcopter payload add and drop applications,” in *2015 34th Chinese Control Conference (CCC)*. IEEE, 2015, pp. 3252–3257.

[56] Y. Xiao, Y. Fu, C. Wu, and P. Shao, “Modified model reference adaptive control of uav with wing damage,” in *2016 2nd International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, 2016, pp. 189–193.

[57] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang Jr, “Perception, planning, control, and coordination for autonomous vehicles,” *Machines*, vol. 5, no. 1, p. 6, 2017.

[58] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, “Autonomous inverted helicopter flight via reinforcement learning,” in *Experimental robotics IX*. Springer, 2006, pp. 363–372.

[59] C. Pfeiffer and D. Scaramuzza, “Human-piloted drone racing: Visual processing and control,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3467–3474, 2021.

[60] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[61] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, “Autonomous drone racing with deep reinforcement learning,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1205–1212.

[62] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs *et al.*, “Outracing champion gran turismo drivers with deep reinforcement learning,” *Nature*, vol. 602, no. 7896, pp. 223–228, 2022.

[63] L. Smith, I. Kostrikov, and S. Levine, “A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning,” *arXiv preprint arXiv:2208.07860*, 2022.

[64] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[65] P. Abbeel and A. Y. Ng, “Apprenticeship learning via inverse reinforcement learning,” in *Proceedings of the 21st International Conference on Machine learning*, 2004, p. 1.

[66] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 1765–1772.

[67] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3758–3765.

[68] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2015.

[69] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[70] CNN. (2017, October) Watch dubai police test its latest gadget – a flying motorbike. [Online]. Available: https://www.cnn.com/2017/10/13/middleeast/dubai-police-hoversurf-flying-motorbike/index.html

[71] F. H. F. Handbook, "Us department of transportation, faah808321 edition," *URL www. faa. gov*, vol. 91, 2012.

[72] A. C. Chapman and P. G. Wright, "Ground effect vehicle," Jun. 7 1983, uS Patent 4,386,801.

[73] S. Komissarov and E. Gordon, *Soviet and Russian Ekranoplans*. Midland, 2010.

[74] G. Van Es, "Running out of runway: Analysis of 35 years of landing overrun accidents," *NLR TP*, vol. 498, p. 2005, 2005.

[75] T. Sorensen, "Aviation accident final report," National Transportation Safety Board, non-fatal CEN16LA039, November 2016.

[76] B. C. Rayner, "Aviation accident final report," National Transportation Safety Board, non-fatal ERA16CA160, June 2016.

[77] M. F. Gallo, "Aviation accident final report," National Transportation Safety Board, fatal CEN11FA507, February 2013.

[78] A. Lemishko, "Aviation accident final report," National Transportation Safety Board, fatal ERA11FA272, August 2014.

[79] R. Staufenbiel and U.-J. Schlichting, "Stability of airplanes in ground effect," *Journal of Aircraft*, vol. 25, no. 4, pp. 289–294, 1988.

[80] W. Johnson, *Helicopter theory*. Courier Corporation, 2012.

[81] E. Davis, J. Spollard, and P. Pounds, "Passive height stability and trajectory repeatability of a quadrotor maneuvering in ground effect with regulated voltage bus," in *Australasian conference on robotics and automation (ACRA 2015)*. ARAA, 2015.

[82] D. A. Griffiths, S. Ananthan, and J. G. Leishman, "Predictions of rotor performance in ground effect using a free-vortex wake model," *Journal of the American Helicopter Society*, vol. 50, no. 4, pp. 302–314, 2005.

[83] M. Bangura and R. Mahony, "Nonlinear dynamic modeling for high performance control of a quadrotor," in *Proceedings of Australasian Conference on Robotics and Automation*. Wellington, New Zealand: Australian Robotics and Automation Association, Dec 2012.

[84] P. Sanchez-Cuevas, G. Heredia, and A. Ollero, "Characterization of the aerodynamic ground effect and its influence in multirotor control," *International Journal of Aerospace Engineering*, vol. 2017, pp. 1–17, 08 2017.

[85] L. Danjun, Z. Yan, S. Zongying, and L. Geng, "Autonomous landing of quadrotor based on ground effect modelling," in *Control Conference (CCC), 2015 34th Chinese*. IEEE, 2015, pp. 5647–5652.

[86] X. He, M. Calaf, and K. K. Leang, "Modeling and adaptive nonlinear disturbance observer for closed-loop control of in-ground-effects on multi-rotor uavs," in *ASME 2017 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers, 2017, pp. V003T39A004–V003T39A004.

[87] T. Ryan and H. J. Kim, "Modelling of quadrotor ground effect forces via simple visual feedback and support vector regression," in *AIAA Guidance, Navigation, and Control Conference*, 2012, p. 4833.

[88] D. D. C. Bernard, F. Riccardi, M. Giurato, and M. Lovera, "A dynamic analysis of ground effect for a quadrotor platform," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 10 311–10 316, 2017.

[89] P. Castillo, R. Lozano, and A. Dzul, "Stabilization of a mini rotorcraft with four rotors," *IEEE control systems*, vol. 25, no. 6, pp. 45–55, 2005.

[90] F. Ruggiero, M. A. Trujillo, R. Cano, H. Ascorbe, A. Viguria, C. Peréz, V. Lippiello, A. Ollero, and B. Siciliano, "A multilayer control for multirotor uavs equipped with a servo robot arm," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4014–4020.

[91] M. Krstic, I. Kanellakopoulos, P. V. Kokotovic *et al.*, *Nonlinear and adaptive control design*. Wiley New York, 1995, vol. 222.

[92] P. A. Ioannou and J. Sun, *Robust adaptive control*. PTR Prentice-Hall Upper Saddle River, NJ, 1996, vol. 1.

[93] E. C. Suicmez and A. T. Kutay, "Optimal path tracking control of a quadrotor uav," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2014, pp. 115–125.

[94] G. V. Raffo, M. G. Ortega, and F. R. Rubio, "An integral predictive/nonlinear h_∞ control structure for a quadrotor helicopter," *Automatica*, vol. 46, no. 1, pp. 29–39, 2010.

[95] T. K. Roy and A. Suman, "Adaptive backstepping controller for altitude control of a small scale helicopter by considering the ground effect compensation," in *Informatics, Electronics & Vision (ICIEV), 2013 International Conference on*. IEEE, 2013, pp. 1–5.

[96] B. Landry *et al.*, "Planning and control for quadrotor flight through cluttered environments," Ph.D. dissertation, Massachusetts Institute of Technology, 2015.

[97] C. R. Gomes and J. A. C. C. Medeiros, "Neural network of gaussian radial basis functions applied to the problem of identification of nuclear accidents in a pwr nuclear power plant," *Annals of Nuclear Energy*, vol. 77, pp. 285–293, 2015.

[98] J. Rad, S. Kazem, and K. Parand, "Optimal control of a parabolic distributed parameter system via radial basis functions," *Communications in Nonlinear Science and Numerical Simulation*, vol. 19, no. 8, pp. 2559–2567, 2014.

[99] P. Pounds, R. Mahony, and P. Corke, "Modelling and control of a large quadrotor robot," *Control Engineering Practice*, vol. 18, no. 7, pp. 691–699, Jul 2010.

[100] K. P. Jain, T. Fortmuller, J. Byun, S. A. Mäkiharju, and M. W. Mueller, "Modeling of aerodynamic disturbances for proximity flight of multirotors," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019, pp. 1261–1269.

[101] R. B. Stull, *An introduction to boundary layer meteorology*. Springer Science & Business Media, 2012, vol. 13.

[102] P. Wei, S. N. Chan, S. Lee, and Z. Kong, "Mitigating ground effect on mini quadcopters with model reference adaptive control," *International Journal of Intelligent Robotics and Applications*, vol. 3, no. 3, pp. 283–297, 2019.

[103] N. Michel, A. K. Sinha, Z. Kong, and X. Lin, "Multiphysical modeling of energy dynamics for multirotor unmanned aerial vehicles," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, Jun Jun 2019, pp. 738–747.

[104] S. Huler, *Defining the wind: the Beaufort scale and how a 19th-century admiral turned science into poetry*. Crown, 2007.

[105] C. Powers, D. Mellinger, and V. Kumar, "Quadrotor kinematics and dynamics," *Handbook of unmanned aerial vehicles*, pp. 307–328, 2015.

[106] G. MCDANIEL and R. BURROWS, *A method of trajectory analysis with multi- mission capability and guidance application*. American Institute of Aeronautics and Astronautics, aug 1968.

[107] W. Johnson, *Rotorcraft aeromechanics*. Cambridge University Press, 2013, vol. 36.

[108] H. Yeo and E. A. Romander, "Loads correlation of a full-scale uh-60a airloads rotor in a wind tunnel," *Journal of the American Helicopter Society*, vol. 58, no. 2, pp. 1–18, 2013.

[109] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2502–2509.

[110] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 573–580.

[111] D. Dey, K. S. Shankar, S. Zeng, R. Mehta, M. T. Agcayazi, C. Eriksen, S. Daftry, M. Hebert, and J. A. Bagnell, "Vision and learning for deliberative monocular cluttered flight," in *Field and Service Robotics*. Springer, 2016, pp. 391–409.

[112] J. Yang, D. Rao, S.-J. Chung, and S. Hutchinson, "Monocular vision based navigation in gps-denied riverine environments," in *Infotech@ Aerospace 2011*, 2011, p. 1403.

[113] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

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

[115] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[116] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics research*. Springer, 2016, pp. 649–666.

[117] J. Kwak and Y. Sung, "Autonomous uav flight control for gps-based navigation," *IEEE Access*, vol. 6, pp. 37 947–37 955, 2018.

[118] S. Scherer, J. Rehder, S. Achar, H. Cover, A. Chambers, S. Nuske, and S. Singh, "River mapping from a flying robot: state estimation, river detection, and obstacle mapping," *Autonomous Robots*, vol. 33, no. 1-2, pp. 189–214, apr 2012.

[119] A. Chambers, S. Achar, S. Nuske, J. Rehder, B. Kitt, L. Chamberlain, J. Haines, S. Scherer, and S. Singh, "Perception for a river mapping robot," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 227–234.

[120] S. Scherer, S. Singh, L. Chamberlain, and S. Saripalli, "Flying fast and low among obstacles," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 2023–2029.

[121] H. Alvarez, L. M. Paz, J. Sturm, and D. Cremers, "Collision avoidance for quadrotors with a monocular camera," in *Experimental Robotics*. Springer, 2016, pp. 195–209.

[122] S. Daftry, S. Zeng, A. Khan, D. Dey, N. Melik-Barkhudarov, J. A. Bagnell, and M. Hebert, "Robust monocular flight in cluttered outdoor environments," *arXiv preprint arXiv:1604.04779*, 2016.

[123] L. Matthies, R. Brockers, Y. Kuwata, and S. Weiss, "Stereo vision-based obstacle avoidance for micro air vehicles using disparity space," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 3242–3249.

[124] H. Oleynikova, D. Honegger, and M. Pollefeys, "Reactive avoidance using embedded stereo vision for mav flight," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 50–56.

[125] N. J. Sanket, C. D. Singh, K. Ganguly, C. Fermüller, and Y. Aloimonos, "Gapflyt: Active vision based minimalist structure-less gap detection for quadrotor flight," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2799–2806, 2018.

[126] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous uav guidance," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1-4, p. 65, 2010.

[127] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, vol. 29, no. 2, pp. 315–378, 2012.

[128] J. Tordesillas and J. P. How, "PANTHER: Perception-aware trajectory planner in dynamic environments," *arXiv preprint arXiv:2103.06372*, 2021.

[129] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[130] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, "Safe local exploration for replanning in cluttered unknown environments for microaerial vehicles," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1474–1481, 2018.

[131] J. Tordesillas, B. T. Lopez, J. Carter, J. Ware, and J. P. How, "Real-time planning with multi-fidelity models for agile flights in unknown environments," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 725–731.

[132] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, vol. 27, no. 5, pp. 534–560, 2010.

[133] H. Wen, R. Clark, S. Wang, X. Lu, B. Du, W. Hu, and N. Trigoni, "Efficient indoor positioning with visual experiences via lifelong learning," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 814–829, 2018.

[134] L. Clement, J. Kelly, and T. D. Barfoot, “Robust monocular visual teach and repeat aided by local ground planarity and color-constant imagery,” *Journal of field robotics*, vol. 34, no. 1, pp. 74–97, 2017.

[135] T. Krajník, P. Cristóforis, K. Kusumam, P. Neubert, and T. Duckett, “Image features for visual teach-and-repeat navigation in changing environments,” *Robotics and Autonomous Systems*, vol. 88, pp. 127–141, 2017.

[136] M. Warren, M. Greeff, B. Patel, J. Collier, A. P. Schoellig, and T. D. Barfoot, “There’s no place like home: Visual teach and repeat for emergency return of multirotor uavs during gps failure,” *IEEE Robotics and automation letters*, vol. 4, no. 1, pp. 161–168, 2018.

[137] D.-J. Lee, P. Merrell, Z. Wei, and B. E. Nelson, “Two-frame structure from motion using optical flow probability distributions for unmanned air vehicle obstacle avoidance,” *Machine Vision and Applications*, vol. 21, no. 3, pp. 229–240, 2010.

[138] S. Hrabar, G. S. Sukhatme, P. Corke, K. Usher, and J. Roberts, “Combined optic-flow and stereo-based navigation of urban canyons for a uav,” in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 3309–3316.

[139] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[140] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[141] D. A. Pomerleau, “Alvinn: An autonomous land vehicle in a neural network,” in *Proceedings of the 1st International Conference on Neural Information Processing Systems*, 1988, p. 305–313.

[142] S. Ross and D. Bagnell, “Efficient reductions for imitation learning,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 661–668.

[143] S. Ross, G. Gordon, and D. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 2011, pp. 627–635.

[144] T. Ablett, F. Marić, and J. Kelly, “Fighting failures with fire: Failure identification to reduce expert burden in intervention-based learning,” *arXiv preprint arXiv:2007.00245*, 2020.

[145] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, “Hg-dagger: Interactive imitation learning with human experts,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8077–8083.

[146] V. G. Goecks, G. M. Gremillion, V. J. Lawhern, J. Valasek, and N. R. Waytowich, "Efficiently combining human demonstrations and interventions for safe training of autonomous systems in real-time," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 2462–2470.

[147] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017.

[148] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[149] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[150] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[151] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," 2016.

[152] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β-vae," *arXiv preprint arXiv:1804.03599*, 2018.

[153] H. Kim and A. Mnih, "Disentangling by factorising," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2649–2658.

[154] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *International conference on machine learning*. PMLR, 2016, pp. 1558–1566.

[155] G. Pajares, "Overview and current status of remote sensing applications based on unmanned aerial vehicles (uavs)," *Photogrammetric Engineering & Remote Sensing*, vol. 81, no. 4, pp. 281–329, 2015.

[156] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[157] Z. Ameli, Y. Aremanda, W. A. Friess, and E. N. Landis, "Impact of uav hardware options on bridge inspection mission capabilities," *Drones*, vol. 6, no. 3, p. 64, 2022.

[158] S. W. Chen, G. V. Nardari, E. S. Lee, C. Qu, X. Liu, R. A. F. Romero, and V. Kumar, "Sloam: Semantic lidar odometry and mapping for forest inventory," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 612–619, 2020.

[159] G. Loianno and V. Kumar, "Vision-based fast navigation of micro aerial vehicles," in *Micro-and Nanotechnology Sensors, Systems, and Applications VIII*, vol. 9836. SPIE, 2016, pp. 280–293.

[160] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *arXiv preprint arXiv:1704.02532*, 2017.

[161] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[162] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[163] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor," in *Robotics: Science and Systems IX*. Robotics: Science and Systems Foundation, 2013.

[164] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico *et al.*, "Fast, autonomous flight in gps-denied and cluttered environments," *Journal of Field Robotics*, vol. 35, no. 1, pp. 101–120, 2018.

[165] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart *et al.*, "Cerberus in the darpa subterranean challenge," *Science Robotics*, vol. 7, no. 66, p. eabp9742, 2022.

[166] M. Aly, "Real time detection of lane markers in urban streets," in *2008 IEEE Intelligent Vehicles Symposium*. IEEE, 2008, pp. 7–12.

[167] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[168] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341–1360, 2020.

[169] J. Zhang, A. Chambers, S. Maeta, M. Bergerman, and S. Singh, "3d perception for accurate row following: Methodology and results," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 5306–5313.

[170] Z. Fei and S. Vougioukas, "Row-sensing templates: A generic 3d sensor-based approach to robot localization with respect to orchard row centerlines," *Journal of Field Robotics*, vol. 39, no. 6, pp. 712–738, 2022.

[171] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[172] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2997–3004.

[173] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[174] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Department, Iowa State University, Tech. Rep., 1998.

[175] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[176] N. Michel, P. Wei, Z. Kong, and X. Lin, "Energy-optimal unmanned aerial vehicles motion planning and control based on integrated system physical dynamics," *Journal of Dynamic Systems, Measurement, and Control*, vol. 145, no. 4, 2023.

[177] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, p. eabh1221, 2021.

[178] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.

[179] G. C. Goodwin, S. F. Graebe, M. E. Salgado *et al.*, *Control system design*. Prentice Hall Upper Saddle River, 2001, vol. 240.

[180] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[181] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE intelligent vehicles symposium (IV)*. IEEE, 2015, pp. 1094–1099.

[182] N. Stefas, H. Bayram, and V. Isler, "Vision-based monitoring of orchards with uavs," *Computers and Electronics in Agriculture*, vol. 163, p. 104814, 2019.

[183] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern recognition*, vol. 77, pp. 354–377, 2018.

[184] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 1–14, 2019.

[185] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4243–4250.

[186] Z. Murez, S. Kolouri, D. Kriegman, R. Ramamoorthi, and K. Kim, "Image to image translation for domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4500–4509.

[187] K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine, "Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6008–6014.

[188] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[189] A. Mehta, A. Subramanian, and A. Subramanian, "Learning end-to-end autonomous driving using guided auxiliary supervision," in *Proceedings of the 11th Indian Conference on Computer Vision, Graphics and Image Processing*, 2018, pp. 1–8.

[190] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3948–3955.

[191] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: Learning agile flight in dynamic environments," in *Conference on Robot Learning*. PMLR, 2018, pp. 133–145.

[192] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning," in *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, jun 2018.

[193] V. Vapnik and A. Vashist, "A new learning paradigm: Learning using privileged information," *Neural networks*, vol. 22, no. 5-6, pp. 544–557, 2009.

[194] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Conference on Robot Learning*. PMLR, 2020, pp. 66–75.

[195] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, 2021.

[196] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020.

[197] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “Carla: An open urban driving simulator,” in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[198] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, “Flightmare: A flexible quadrotor simulator,” in *Conference on Robot Learning*. PMLR, 2021, pp. 1147–1157.

[199] T. Qin, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual-inertial state estimator,” *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[200] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, “OpenVINS: A research platform for visual-inertial estimation,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2020.

[201] M. Labbé and F. Michaud, “Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation,” *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[202] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[203] D. Zhang, A. Loquercio, X. Wu, A. Kumar, J. Malik, and M. W. Mueller, “A zero-shot adaptive quadcopter controller,” *arXiv preprint arXiv:2209.09232*, 2022.

Appendix A

MRAC for MIMO System

Consider the linear system equation in the form of:

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B(\mathbf{u}(t) + f(\mathbf{x})) \tag{A.1}$$

where $f(\mathbf{x})$ is the matched uncertainty in the system and is a linear combination of N known locally as Lipschitz-continuous basis function, $\Phi(x)$, with unknown constant coefficient, Θ:

$$f(x) = \Theta^T \Phi(x) \tag{A.2}$$

Define the reference model as:

$$\dot{\mathbf{x}}_{ref}(t) = A_{ref}\mathbf{x}_{ref}(t) + B_{ref}\mathbf{r}(t) \tag{A.3}$$

which characterizes the desired behavior of the closed-loop system. A_{ref} is a Hurwitz matrix and $\mathbf{r}(t)$ is the external reference signal. The goal of MRAC is to let the closed-loop system states track the reference signals. Define the state tracking error as:

$$\mathbf{e}(t) = \mathbf{x}(t) - \mathbf{x}_{ref}(t) \tag{A.4}$$

and the error should asymptotically converge to zero. The controller can be designed as:

$$\mathbf{u}(t) = \left(K_x^T\mathbf{x}(t) + K_r^T\mathbf{r}(t)\right) - \hat{\Theta}^T\Phi(\mathbf{x}) \tag{A.5}$$

where $K_x^T\mathbf{x}(t) + K_r^T\mathbf{r}(t)$ corresponds to the feedback + feedforward part, and $-\hat{\Theta}^T\Phi(\mathbf{x})$ is the adaptive part compensating for the model uncertainty with estimated parameter $\hat{\Theta}$. The following *matching conditions* should be satisfied in order to achieve the desired reference behavior:

$$\begin{aligned} A + BK_x^T &= A_{ref} \\ BK_r^T &= B_{ref} \end{aligned} \tag{A.6}$$

Neglect $\cdot(t)$ and $\cdot(x)$ for simplifications. The estimated parameters are generated online through the inverse Lyapunov analysis. Consider a globally radially unbounded quadratic Lyapunov function in the form of:

$$V(\mathbf{e}, \Delta\Theta) = \mathbf{e}^T P \mathbf{e} + trace\left(\Delta\Theta^T \Gamma_\Theta^{-1} \Delta\Theta\right) \tag{A.7}$$

where $\Delta\Theta = \hat{\Theta} - \Theta$ represents the parameter estimation error and Γ_Θ is the adaption rate. P satisfies the algebraic Lyapunov equation as follows:

$$A_{ref}^T P + P A_{ref} = -Q \tag{A.8}$$

where P and Q are symmetric and positive definite matrices. Since the derivative of the tracking error can be calculated as:

$$\begin{aligned} \dot{\mathbf{e}} &= \dot{\mathbf{x}} - \dot{\mathbf{x}}_{ref} \\ &= A\mathbf{x} + B\left(K_x^T \mathbf{x} + K_r^T \mathbf{r} - \hat{\Theta}^T \Phi + \Theta^T \Phi\right) - A_{ref}\mathbf{x}_{ref} - B_{ref}\mathbf{r} \\ &= (A + BK_x^T)\mathbf{x} - A_{ref}\mathbf{x}_{ref} + BK_r^T \mathbf{r} - B_{ref}\mathbf{r} - B\Delta\Theta^T \Phi \\ &= A_{ref}\mathbf{e} - B(\Delta\Theta^T \Phi) \end{aligned} \tag{A.9}$$

Then the lie derivative of V is equal to:

$$\begin{aligned} \dot{V} &= \dot{\mathbf{e}}^T P \mathbf{e} + \mathbf{e}^T P \dot{\mathbf{e}} + 2trace\left(\Delta\Theta^T \Gamma_\Theta^{-1} \dot{\hat{\Theta}}\right) \\ &= \mathbf{e}^T (A_{ref} P + P A_{ref}) \mathbf{e} - 2\mathbf{e}^T P B (\Delta\Theta^T \Phi) + 2trace(\Delta\Theta^T \Gamma_\Theta^{-1} \dot{\hat{\Theta}}) \\ &= -\mathbf{e}^T Q \mathbf{e} + 2[-\mathbf{e}^T P B (\Delta\Theta^T \Phi) + trace(\Delta\Theta^T \Gamma_\Theta^{-1} \dot{\hat{\Theta}})] \end{aligned} \tag{A.10}$$

Use the property of vector trace identity:

$$\left(\mathbf{e}^T P B\right)\left(\Delta\Theta^T \Phi\right) = trace\Big(\left(\Delta\Theta^T \Phi\right)\left(\mathbf{e}^T P B\right)\Big) \tag{A.11}$$

The equation can be rewritten as:

$$\dot{V} = -\mathbf{e}^T Q \mathbf{e} + 2trace(\Delta\Theta^T \left[\Gamma_\Theta^{-1} \dot{\hat{\Theta}} - \Phi \mathbf{e}^T P B\right]) \tag{A.12}$$

The *adaptive law* can be designed as:

$$\dot{\hat{\Theta}} = \Gamma_{\Theta} \Phi \mathbf{e}^T PB \tag{A.13}$$

and therefore the derivative of V is globally negative semi-definite:

$$\dot{V} = -\mathbf{e}^T Q \mathbf{e} \leq 0 \tag{A.14}$$

All signals in the closed-loop system remain uniformly bounded in time.

Appendix B

Equations of Motion for Multirotor UAV

Consider the Newton-Euler equations in body frame:

$$\mathscr{F} = m\left(\dot{\mathbf{V}} + \omega \times \mathbf{V}\right) \tag{B.1}$$

$$\mathscr{M} = \mathbf{I}\dot{\omega} + \omega \times (\mathbf{I}\omega) \tag{B.2}$$

where $\mathscr{F}$ and $\mathscr{M}$ are the external forces and moments acting on the vehicle. m is the mass weight, $\mathbf{V} = [u,\ v,\ w]^T$ is the body velocity vector, $\omega = [p,\ q,\ r]^T$ is the angular velocity vector, $\mathbf{I} = diag(I_{xx},\ I_{yy},\ I_{zz})$ is the moment of inertia matrix. The equations can be expanded as:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{B.3}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{xx}} qr \\ \frac{I_{zz}-I_{xx}}{I_{yy}} pr \\ \frac{I_{xx}-I_{yy}}{I_{zz}} pq \end{bmatrix} + \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \begin{bmatrix} L \\ M \\ N \end{bmatrix} \tag{B.4}$$

with external forces $\mathscr{F} = [X,\, Y,\, Z]^T$ and external moments $\mathscr{M} = [L,\, M,\, N]^T$. Define $[P_N,\, P_E,\, P_D]^T$ as the position vector in inertial frame and $[\phi,\ \theta,\ \psi]^T$ are the corresponding Euler angles. By slightly abusing the notations (e.g., $s_\phi = \sin\phi$, $c_\theta = \cos\theta$, $t_\psi = \tan\psi$), we have the relations:

$$\begin{bmatrix} \dot{P}_N \\ \dot{P}_E \\ \dot{P}_D \end{bmatrix} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi c_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{B.5}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & s_{\phi}t_{\theta} & c_{\phi}t_{\theta}\\ 0 & c_{\phi} & -s_{\phi}\\ 0 & s_{\phi}/c_{\theta} & c_{\phi}/c_{\theta}\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{B.6}$$

The equations of motion can be linearized at equilibrium conditions which will produce a 6-DOF model in the form of:

$$\begin{aligned} M\dot{\mathbf{x}} &= F\mathbf{x} + G\mathbf{u} \\ \mathbf{y} &= H_0\mathbf{x} + H_1\dot{\mathbf{x}} \end{aligned} \tag{B.7}$$

where $\mathbf{x} = [u,\ v,\ w,\ p,\ q,\ r,\ \phi,\ \theta,\ \psi]^T$ is the state vector, and $\mathbf{u} = [\delta_{lat},\ \delta_{lon},\ \delta_{yaw},\ \delta_{thr}]^T$ is the control input vector. M is an identify matrix, F is the stability derivative matrix, and G is the control derivative matrix. Denote the dimensional stability derivatives and dimensional control derivatives in subscript notations (e.g., $\frac{\partial X}{\partial v} = \frac{1}{m}X_v$ and $\frac{\partial X}{\partial \delta_{lon}} = X_{lon}$) and with $[U_0,\ V_0,\ W_0,\ \phi_0,\ \theta_0]^T$ being the trim conditions, the matrices can be written as:

$$F = \begin{bmatrix} X_u & X_v & X_w & X_p & X_q-W_0 & X_r+V_0 & 0 & -gc_{\theta_0} & 0\\ Y_u & Y_v & Y_w & Y_p+W_0 & Y_q & Y_r-U_0 & gc_{\phi_0}c_{\theta_0} & -gs_{\phi_0}s_{\theta_0} & 0\\ Z_u & Z_v & Z_w & Z_p-V_0 & Z_q+U_0 & Z_r & -gs_{\phi_0}c_{\theta_0} & -gc_{\phi_0}s_{\theta_0} & 0\\ L_u & L_v & L_w & L_p & L_q & L_r & 0 & 0 & 0\\ M_u & M_v & M_w & M_p & M_q & M_r & 0 & 0 & 0\\ N_u & N_v & N_w & N_p & N_q & N_r & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & s_{\phi_0}t_{\theta_0} & c_{\phi_0}t_{\theta_0} & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & c_{\phi_0} & -s_{\phi_0} & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & s_{\phi_0}/c_{\theta_0} & c_{\phi_0}/c_{\theta_0} & 0 & 0 & 0 \end{bmatrix},\quad G = \begin{bmatrix} X_{lat} & X_{lon} & X_{yaw} & X_{thr}\\ Y_{lat} & Y_{lon} & Y_{yaw} & Y_{thr}\\ Z_{lat} & Z_{lon} & Z_{yaw} & Z_{thr}\\ L_{lat} & L_{lon} & L_{yaw} & L_{thr}\\ M_{lat} & M_{lon} & M_{yaw} & M_{thr}\\ N_{lat} & N_{lon} & N_{yaw} & N_{thr}\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{B.8}$$

H_0 and H_1 are the measurement matrices, allowing the measurement vector $\mathbf{y}$ to be expressed in terms of the state vector $\mathbf{x}$ and its derivative $\dot{\mathbf{x}}$, and $\mathbf{y} = [u,\ v,\ w,\ p,\ q,\ r,\ a_x,\ a_y,\ a_z]^T$.

$$H_0 = \begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0&0&0\\ 0&0&0&0&W_0&-V_0&0&gc_{\theta_0}&0\\ 0&0&0&-W_0&0&U_0&-gc_{\theta_0}&0&0\\ 0&0&0&V_0&-U0&0&0&gs_{\theta_0}&0 \end{bmatrix},\quad H_1 = \begin{bmatrix} 1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0 \end{bmatrix} \tag{B.9}$$

By augmenting the model with the extra first-order dynamic terms, the new state vector and control vector become:

$$\begin{aligned} \mathbf{x} &= [u,\ v,\ w,\ p,\ q,\ r,\ \phi,\ \theta,\ \psi,\ \delta'_{lat},\ \delta'_{lon},\ \delta'_{thr}]^T \\ \mathbf{u} &= [\delta_{lat},\ \delta_{lon},\ \delta_{yaw},\ \delta_{thr}]^T \\ \mathbf{u_w} &= [\delta^w_{lat},\ \delta^w_{lon},\ \delta^w_{thr}]^T \end{aligned} \tag{B.10}$$

where $\mathbf{u}$ is the control input from flight control system, and $\mathbf{u_w}$ is the contribution from wind, modeled as an additional input. The first-order dynamics has the form of:

$$\frac{\delta'(s)}{\delta(s)} = \frac{1}{T_a s + 1} \tag{B.11}$$

where T_a is the identified time constant. Moreover, the time delay in each control axis is explicitly included in the state-space model to account for unmodeled high-frequency dynamics. The final state-space model has the formula of:

$$\begin{aligned} M\dot{\mathbf{x}} &= F\mathbf{x} + G_u\mathbf{u}(t-\tau_\mathbf{u}) + G_w\mathbf{u_w}(t-\tau_\mathbf{w}) \\ \mathbf{y} &= H_0\mathbf{x} + H_1\dot{\mathbf{x}} \end{aligned} \tag{B.12}$$

$$F = \begin{bmatrix} X_u & X_v & X_w & X_p & X_q - W_0 & X_r + V_0 & 0 & -gc_{\theta_0} & 0 & X_{lat} & X_{lon} & X_{thr} \\ Y_u & Y_v & Y_w & Y_p + W_0 & Y_q & Y_r - U_0 & gc_{\phi_0}c_{\theta_0} & -gs_{\phi_0}s_{\theta_0} & 0 & Y_{lat} & Y_{lon} & Y_{thr} \\ Z_u & Z_v & Z_w & Z_p - V_0 & Z_q + U_0 & Z_r & -gs_{\phi_0}c_{\theta_0} & -gc_{\phi_0}s_{\theta_0} & 0 & Z_{lat} & Z_{lon} & Z_{thr} \\ L_u & L_v & L_w & L_p & L_q & L_r & 0 & 0 & 0 & L_{lat} & L_{lon} & L_{thr} \\ M_u & M_v & M_w & M_p & M_q & M_r & 0 & 0 & 0 & M_{lat} & M_{lon} & M_{thr} \\ N_u & N_v & N_w & N_p & N_q & N_r & 0 & 0 & 0 & N_{lat} & N_{lon} & N_{thr} \\ 0 & 0 & 0 & 1 & s_{\phi_0}t_{\theta_0} & c_{\phi_0}t_{\theta_0} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{\phi_0} & -s_{\phi_0} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & s_{\phi_0}/c_{\theta_0} & c_{\phi_0}/c_{\theta_0} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1/T_a & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1/T_a & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1/T_a \end{bmatrix} \tag{B.13}$$

$$G_u = \begin{bmatrix} 0 & 0 & X_{yaw} & 0 \\ 0 & 0 & Y_{yaw} & 0 \\ 0 & 0 & Z_{yaw} & 0 \\ 0 & 0 & L_{yaw} & 0 \\ 0 & 0 & M_{yaw} & 0 \\ 0 & 0 & N_{yaw} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1/T_a & 0 & 0 & 0 \\ 0 & 1/T_a & 0 & 0 \\ 0 & 0 & 0 & 1/T_a \end{bmatrix}, \quad G_w = \begin{bmatrix} X_{w,lat} & X_{w,lon} & X_{w,thr} \\ Y_{w,lat} & Y_{w,lon} & Y_{w,thr} \\ Z_{w,lat} & Z_{w,lon} & Z_{w,thr} \\ L_{w,lat} & L_{w,lon} & L_{w,thr} \\ M_{w,lat} & M_{w,lon} & M_{w,thr} \\ N_{w,lat} & N_{w,lon} & N_{w,thr} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{B.14}$$

where G_u and G_w are the control derivative matrices with respect to the controller and the wind, respectively.