Unmanned Aircraft Systems Flight Planning: System Development and Feasibility Study

by

Zhilong Liu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Civil and Environmental Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Raja Sengupta, Chair
Professor Scott Moura
Professor Claire Tomlin

Fall 2017

Unmanned Aircraft Systems Flight Planning: System Development and Feasibility Study

Abstract

Unmanned Aircraft Systems Flight Planning: System Development and Feasibility Study

by

Zhilong Liu

Doctor of Philosophy in Civil and Environmental Engineering

University of California, Berkeley

Professor Raja Sengupta, Chair

Unmanned Aircraft Systems (UAS), commonly known as drones, are becoming increasingly popular in civilian applications. However, the infrastructure to ensure the safe operations are still largely missing. In this dissertation, I addressed two problems related to a flight mission. First, I propose a flight planning system specifically for single multirotor UAS operations. The system generates a 4D trajectory at the pre-flight stage before an actual flight taking place. Safety is achieved by ensuring enough energy to finish a trip under environmental influences such as wind, as well as avoiding static obstacles. Second, I proposed a hybrid controller capable of performing trajectory following as well as single-helicopter collision avoidance. The controller enables a multi-rotor to follow the planned trajectory.

In the first part of the dissertation, I first divide the flight planning system into two components: energy demand and energy supply. The demand side is determined by the performance of the UAS, the environment, and the flight trajectory, while the energy supply depends on the power source. Then, I focused on two specific problems on the energy demand side. First, I quantify the energy performance of a UAS by developing a theoretical power consumption model for multi-rotor Unmanned Aircraft Systems (UAS). The model is derived from the helicopter literature. The model parameters are identified and validated experimentally by flying an IRIS+ quadrotor UAS. Second, I solved the trajectory generation problem by formulating it as an optimal control problem. The cost to be optimized is the energy consumption or flight duration at the cruise phase of a flight mission, taking into account 3-dimensional wind field and terrain variation. The problem is solved numerically by the Ordered Upwind Method (OUM). By combining the optimized path with timestamps, we can generate 4D trajectories typically used in aviation flight planning, and thus maximize the software compatibility between manned and unmanned aviation. Simulation shows that the generated trajectories are able to avoid obstacles such as high-rise buildings and give energy consumption estimates along the entire trajectory.

In the second part of the dissertation, I addressed the trajectory following problem with collision avoidance capability given a planned 4D trajectory. I analyzed the tracking performance of an IRIS+ quadrotor for a challenging trajectory. In the scenario, wind played a

big role in the cross-track errors. The collision avoidance capability, or a safety controller, is integrated to the trajectory following controller by introducing a hybrid automaton, deciding when to start the avoidance action and what control actions to take. The hybrid controller is examined in a simple straight-line-following scenario to avoid a manned helicopter.

Contents

List of Figures

List of Tables

Acknowledgments

I first want to thank my advisor, Raja Sengupta, for his invaluable feedbacks through my PhD career at UC Berkeley. Raja's expertise in UASs and control was really helpful. He contributed a lot in the structural developments of this dissertation. When writing papers, Raja is supportive, although it would be better if the advice were given in a more timely matter. I also enjoyed the small group discussions in Friday lunches with lab mates or other researchers. I thank the other members of my committee for their contributions as well. Prof. Scott Moura provided inspiring feedbacks for the MPC simulation and the HJB equation derivations. Prof. Claire Tomlin helped me resolve the convexity concern in the numerical method. Prof. Karl Hedrick assisted me to formulate the sliding controllers with DSC filters for a quadrotor. Without him, I would have missed the most elegant solution to this problem. Without my committee members, I could not have delivered this fine piece of original work. Lastly, I want to thank Dounan Tang, my colleague who also works with Raja. He provided me with excellent suggestions on almost every part of the dissertation, including system design, modeling, optimization, and simulations.

I am grateful to all my family and friends for their constant support and encouragement. My parents are the most supportive. My father gave me advice on every important part of my life, and my mother prepares good food for me every week. My roommate Qiuchen Guo is my closest friend. Throughout my PhD career, she listened to my complaints in life and gave me encouragements. Without her, I could have given up a few years ago. I also want to thank Xin Qian and Qian Zhong for bringing me so much happiness in life. You two and Qiuchen are my energy source at UC Berkeley. In addition, I had a wonderful experience working with Professor Mark Mueller in ME233. He familiarized me with many fundamental concepts in control theory, which I would have no chance to grasp otherwise.

Financial support for this research was provided by the University Affiliated Research Center (UARC) at NASA Ames (Contract number: UCSCMCA-14-020) and the National Science Foundation (NSF) (Contract number: CNS-1136141). Both are gratefully acknowledged.

Chapter 1

Introduction

1.1 Overview

This dissertation is about Unmanned Aircraft Systems (UAS), or drones. UAS are exciting now and both government and industry is innovating rapidly. The Federal Aviation Administration (FAA) is currently revolutionizing the National Airspace System (NAS) by introducing the Next Generation Air Transportation System (NextGen). The robust integration of UAS into the NAS is in the NextGen road map. To achieve this goal, the National Aeronautics and Space Administration (NASA) is actively developing the core technology, leading to a new research field called UAS traffic management (UTM) [63]. UTM faces many challenges in sensing, control, and flight management. This dissertation helps resolve three of them, namely UAS flight planning, trajectory following, and collision avoidance with manned helicopters.

To deliver a safe flight, we need a pre-flight stage and an in-flight stage, thus the two-part structure of this dissertation. The first part advances the pre-flight task, or flight planning. We propose a flight planning system to generate a 4D trajectory defined by 3D waypoints with timestamps (or speedstamps). The second part focuses on in-flight tasks, including trajectory following and collision avoidance. We design a hybrid controller capable of performing trajectory following (speed-stamped) as well as collision avoidance with manned helicopters since they are the primary manned aircraft at low altitude. The subsequent paragraphs summarize the contribution of this dissertation to each problem.

1.1.1 Part I: Pre-flight

In Part I of the dissertation, we propose a UAS flight planning system to ensure that a UAS can fly from an origin to a destination with either minimum energy or time, avoiding static obstacles and under the influence of wind. To put the problem into context, we take a package delivery example similar to Amazon Prime Air (Figure 1.1) [28]. When the UAS flies from a distribution center to somebody's home, it has to resolve several fundamental problems. First, it has to determine a flight trajectory along with an energy estimate and

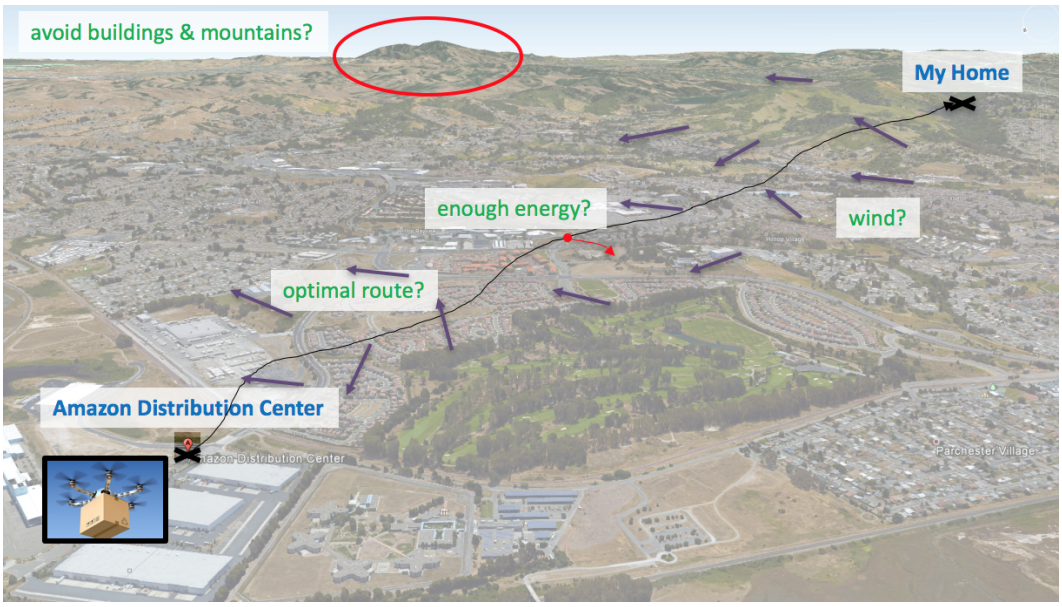


Figure 1.1: A motivating example for the proposed energy-based UAS flight planning system.

an arrival time. Energy shortage can cause mission failures. Second, the UAS has to avoid static obstacles such as buildings and mountains. Third, environmental factors such as wind should be taken into account. Chapters 2, 3, and 4 belongs to this part.

Flight planning (Part I), as established in manned aviation, proposes flight speeds, altitudes, and a horizontal path from an origin to a destination. The three together define a 4D trajectory. We adopt the same focus, and compute all three for a UAS mission (see Chapter 2). We aim to pick the three to optimize energy costs and flight time, like flight planning methods in the manned aviation literature. However, the current and future constraints of unmanned aviation induce some important differences, which motivate the innovations in this dissertation.

First, we develop a different algorithm to compute a minimum energy or trip time path for UAS, instead of using the literature (Chapter 4). The algorithmic flight planning literature can broadly be categorized into the graph-based [38] and continuum-based methods [115, 116, 31]. The graph methods use infrastructure like VOR stations and airways to generate the graph. Most of the airspace opened for unmanned aviation has no such infrastructure [92] and therefore no obvious graph. Hence, we seek a continuum approach to flight path computation. The PDE-based approaches classify broadly into the shooting and collocation methods. Both methods compute a 2D path in a rectangular grid defined at fixed altitude. The grid assumption works fine on a fixed altitude above mean sea level (MSL), but it cannot work on altitude above ground level (AGL) because such a surface undulates with the terrain. Our algorithmic contribution on path optimization leverages the fast marching and ordered upwind methods, which generalize to triangulated meshes. UAS flight planning should also account for wind fields, as is the practice in manned aviation. We show in

Chapter 4 that routes accounting for the winds can consume up to 40% less energy than the straight routes. The collocation method models wind fields as low-order polynomials, which works adequately for tropospheric winds. But terrain makes winds in the boundary layer of the atmosphere (first 1km) considerably more variable. Since this is the airspace envisaged for unmanned aviation, we turn to the ordered upwind method to handle wind fields with high variability.

Second, minimum-energy flight path computation methods assign energy consumption values to paths. Chapter 3 contributes a power consumption model for multirotor aircraft. Flight plans computed in Chapter 4 are based on this model. While there is an extensive literature on modeling power consumption for helicopters [57, 68] and fixed-wing aircraft [61], such a literature on multirotors is small even though they are currently the most popular UAS. Hence we focus on a power consumption model for multirotor UASs. The rotorcraft power consumption literature separates power into induced, profile, and parasite power. Flight planning requires all three, but Aleksandrov [6] and Roberts [105] only include induced power. Hoffmann [45] models both induced and profile power for a multirotor at low speed. Our model captures all three components at a range of speeds. We have evaluated our model by calibrating it to a 3DR IRIS+ quadrotor from hover to its maximum speed of $20m/s$ flown at the Richmond Field Station of the University of California, Berkeley [104]. Although this example focuses on an electric UAS, the consumption model is valid for any type of energy source.

Chapter 2 builds an overall UAS flight planning method out of the contributions in Chapters 3 and 4. We present the method in the form of a flight planning case study for an electric quadrotor UAS. The case study computes minimum time and minimum energy routes between multiple origins and a single destination in a region with significant wind and terrain variations. Chapter 2 also shows how to make the energy supply choice for an electric quadrotor. This choice is an analog of the fuel loading problem in manned aviation [88]. Essentially, one cannot estimate the energy demand without assuming a battery weight, and one cannot derive the required battery weight without first knowing the energy demand. Our methods resolve the two jointly in Chapter 2. The method presented is complete in the sense of the established practice of commercial flight planning, though much work remains to be done on computational efficiency. The case study shows how to choose battery weight, flight altitude, flight speed, and flight path cost by energy consumption and trip time to propose a complete unmanned analog of flight planning for manned aviation.

1.1.2 Part II: In-flight

In Part II of the dissertation, we illustrate that the trajectories generated from the FPS in Part I can be followed reasonably well while ensuring safety. Specifically, we design feedback controllers for two specific tasks, namely trajectory following and collision avoidance.

Chapter 5 handles the tracking of the trajectories generated in Chapter 4. We propose a two-level controller architecture to perform trajectory following. The high-level controller is a model predictive controller (MPC), while the low-level controller is a set of sliding con-

trollers. The tracking performance of these control architecture is studied for a challenging trajectory with sharp turns. The major contribution is in the low-level controllers. Multirotors are known to be underactuated. Their roll and pitch dynamics are coupled with horizontal motions. One approach to handle this problem is by adding outer-loop PID controllers [137], but the performance is in general worse than nonlinear approaches such as integrator backstepping [13, 74]. However, the backstepping controllers require rate terms of the synthetic controls, which leads to an explosion of terms [127]. We resolve this problem by augmenting the dynamics with two dynamic surface filters (DSC), to robustly track the 3D positions and Euler angles of a multirotor.

Chapter 6 presents a strategy to avoid collisions with manned aircraft at low altitude. In a dynamical flight environment, we also need to avoid moving objects. At low altitudes, the main moving obstacles are manned helicopters. In this chapter, we formulate a hybrid controller to switch between two modes, namely trajectory-following and collision-avoidance. The sensing technology is assumed to be automatic dependent surveillance-broadcast (ADS-B). Compared to existing techniques, the proposed hybrid safety controller has two advantages. First, it has a sense of optimality compared to [37, 94, 66, 60]. The controller is only enabled when necessary. Second, it is computationally feasible, which is hard to achieve with existing optimization techniques such as mixed-integer programming [78] and other numerical methods solving Hamilton-Jacobi equations [81].

Lastly, Chapter 7 summarizes the contributions of this work and provide recommendations for future study. The appendices include some details not presented in the main chapters, mainly simulation results and derivations of equations. Appropriate references to sections in the appendices are given throughout.

1.2 Summary of Contributions

In summary, the main contributions of this research are:

1. A flight planning system generates an optimal flight path, altitude, speed, and energy source weight.

2. A power consumption model to evaluate energy performance of a multirotor UAS.

3. An optimal routing engine that generate minimum energy or minimum time trajectories.

4. A set of low-level sliding controllers augmented with dynamic surface filters (DSC).

5. A hybrid controller switching between trajectory following and collision avoidance with ADS-B.

Part I

Pre-Flight

Chapter 2

The UAS Flight Planning System

In this chapter, we introduce the UAS flight planning system (FPS). We start by introducing the flight planning problem in aviation and the challenges in the context of UAS operations. Then, we describe the overall UAS flight planning system in Section 2.2. The system components are discussed in conjunction with the aviation literature in the subsequent sections (2.3, 2.4, and 2.5). Lastly, we give a comprehensive example illustrating our UAS flight planning process in Section 2.6.

2.1 Problem Overview

In manned aviation, a flight planning system takes environmental and aircraft performance information as inputs and specifies a 4D trajectory for the aircraft to follow. It is a safety critical system because it involves the fuel calculation to ensure that the aircraft can reach the destination. The amount of fuel is usually estimated conservatively, to ensure passenger safety and reduce energy-based contingency cost. For example, fuel deficiency can cause emergency landing in bad weather conditions. On the other hand, fuel accounts for a significant portion of the aircraft weight, and extra fuel weight causes unnecessary fuel burn. Therefore, flight planners also try to minimize flight cost through the appropriate choice of route, altitude, speed, and by loading the minimum necessary fuel on board. Significant research has been performed to evaluate the efficiency of current aviation practice [110, 109].

The fuel loading problem is a balance between energy demand and supply. To determine how much fuel to load, we first have to figure out the energy demand, which depends on aircraft self weight. But fuel accounts for a significant portion of the aircraft self weight, so we cannot determine the demand without assuming a certain amount of fuel supply. Therefore, fuel planners assume a certain amount of fuel and iteratively compute the energy demand. Similarly, in UAS flight planning, we need to assume an energy source to start with. Our UAS flight planning system focuses on electric multi-rotors with Lithium polymer batteries. In this dissertation, we assume that we have a small number of UASs and batteries to choose from, so that we can address the energy demand problem by exhaustively searching for every

possible vehicle-battery combinations (Section 2.2).

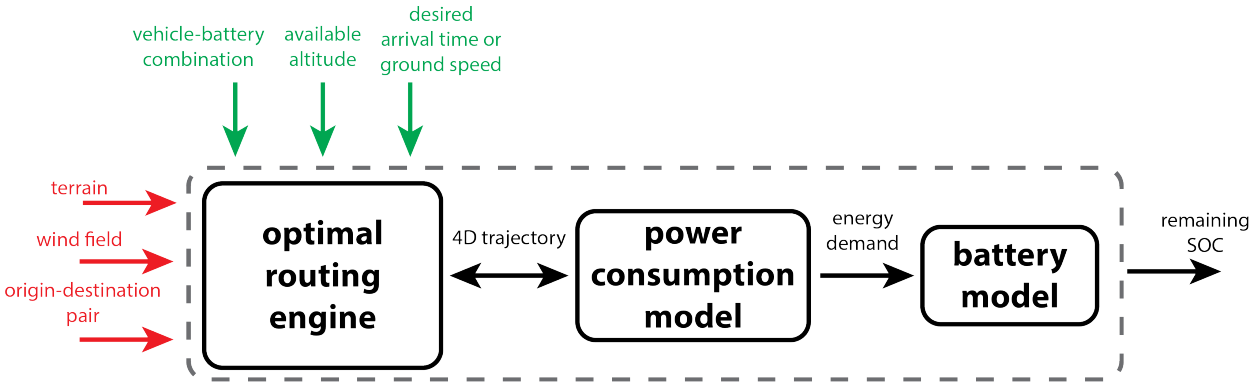
In manned aviation, fuel demand computation involves four steps: payload estimation, route and altitude selection, cruise speed selection, and contingency [29, 88]. Payload estimation includes the weight of transported people and goods. To manage the payload, airlines maintain a recent average history of bags and nominal weights for people and bags. In our FPS, we can safely assume that the payload is given. Contingency is considered by specifying alternate landing airports and contingency fuel. In this dissertation, we don't address the contingency problem in depth because the capacity estimation problem of Lithium Polymer (LiPo) batteries is very challenging. Instead, we handle contingency by specifying threshold on the remaining state-of-charge (SOC) on a battery in Section 2.2.3, assuming that the battery capacity is known and the SOC is accurately estimated by an equivalent circuit model (ECM). In reality, the battery capacity estimation problem is complicated and shall be studied in more detail in future research.

The problem we want to focus on is to select the route, altitude and cruise speed. To the best of the author's knowledge, researchers in aviation are not yet able to solve the join optimization problem to address all three objectives simultaneously. Instead, they usually optimize one by fixing the other two. The complication is that these three problems are coupled. We cannot optimize the route without assuming a cruise speed and altitude [124, 54]. Similarly, to optimize the cruise speed along the trajectory, we have to first pick a route and an altitude [93], and altitude optimization is done at a constant speed with a given route [26]. The challenge is to account for wind-induced cost, which in general is a function of the heading angle. Section 2.3 reviews the relevant routing algorithms in aviation. Similarly, in our algorithm (Chapter 4), we first consider the route optimization problem at a constant cruise speed and altitude. Then, we repeat this process exhaustively for a finite discrete set of cruise speeds and altitudes and pick the best option (Section 2.2).

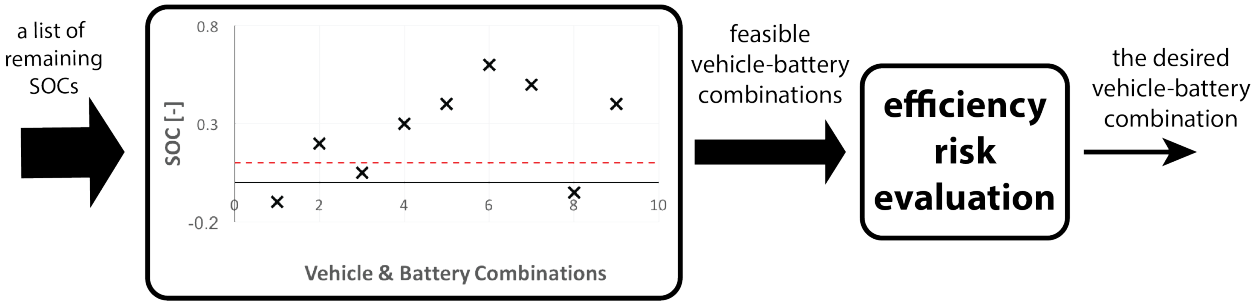
Unlike aviation, in which aircraft flies at constant altitude above mean sea level (AMSL) (See 14 CFR 91.179), we argue that flying at constant altitude above ground level (AGL) is a better approach for low-altitude UAS flights, because it provides an easy way to avoid static obstacles. But to the best of our knowledge, no algorithms in aviation can optimize flight routes at constant altitude AGL. Therefore, we develop a new optimal routing approach which takes into account not only wind-induced cost, but also terrain constraint at constant altitude AGL (Chapter 4). The drawback is that the computation time is slow. The algorithm speed-up problem is left as future work. In the following section, we start introducing our UAS flight planning system. The aviation literature is discussed as necessary in each component.

2.2 The Proposed Flight Planning System

The proposed flight planning system diagram is shown in Figure 2.1. Figure 2.1a shows how to compute the energy demand and supply for a single vehicle, while Figure 2.1b makes the final decision after evaluating all feasible options.



(a) The energy demand and supply computation flow for a *single* vehicle-battery combination.



(b) The decision diagram to select *one out of all* the available vehicle-battery combination..

Figure 2.1: The proposed system diagrams for UAS flight planning.

2.2.1 Inputs

In Figure 2.1a, the inputs in red are fixed for a given flight planning problem. They include terrain and wind data, as well as the origin and destination pair. The inputs in green are the ones we can make different choices on. They include vehicles, energy sources, fly altitudes, and travel speed.

2.2.2 Energy Demand and Supply

Figure 2.1a shows the energy demand and supply computation flow. Each block is described in the sections below. We briefly summarize the process here. First, we select a vehicle-battery combination, an altitude, and a desired speed, and feed all six inputs to the optimal routing engine (Section 2.3), which outputs a 4D trajectory. Second, we feed the 4D trajectory into a power consumption model (Section 2.4) to obtain the energy demand for the given set of green inputs. Finally, the energy demand is fed into a battery model (Section 2.5), to determine whether the battery is sufficient. The output is the remaining state-of-charge (SOC). This process is repeated for all vehicle-battery combinations at different altitude and cruise speed range.

2.2.3 Decision Making

With the list of SOCs generated from the process above for all the vehicle-battery combinations, we can now make a final choice. The decision process is specified in Figure 2.1b. The first decision criterion is the remaining state-of-charge (SOC_{remain}). We set a hard threshold on SOC_{remain} in equation (2.1). The threshold is indicated in a red dashed line in the Figure. A vehicle-battery combination with SOC_{remain} less than SOC_{min} is infeasible. Otherwise, it is feasible. This process filters out all the infeasible options.

$$SOC_{remain} \geq SOC_{min} \tag{2.1}$$

For all the feasible combinations, we then perform efficiency and risk evaluation. The evaluation prioritizes the combinations by imposing soft criteria such as energy efficiency and safety margins. If SOC_{remain} is slightly above SOC_{min}, the combination is energy efficient but risky. Some contingency capacity should be reserved for unexpected situations. On the other hand, if SOC_{remain} is close to 1.0, then the combination is safe but energy inefficient.

2.2.4 Assumptions

Before going into further discussion, we first state some basic assumptions on the choices of the green inputs.

1. We have a small number of multirotor UAS to choose from.
2. The energy source is Lithium Polymer (LiPo) batteries. For each UAS, we have a small number of batteries of different capacities to choose from for each UAS.
3. The UAS can only fly at a fixed altitude above ground level (AGL), and we have a small number of altitudes to choose from.
4. We have a rough idea of the desired trip time or travel speed (ground speed or airspeed). By assuming a simple straight line route, we can compute a first choice of travel speed to start the planning computation.
5. Along the trajectory, we use either a constant ground speed or a constant airspeed. Trajectory optimization with varying speed remains unresolved.

The assumptions are mostly inherited from the aviation literature discussed in Section 2.1, but they also takes into account the uniqueness of multi-rotor UASs. The first two assumptions imply that there are a finite number of UASs and batteries. In this dissertation, we use the term vehicle-battery combination, or simply combination, to refer to a specific choice of a vehicle and a battery. A small number of choices of flight altitude is likely for two reasons. First, UASs are likely to be in low altitude uncontrolled class G airspace [92]. Second, altitude separation is necessary to ensure safety. The constant speed assumption is typical in aviation to find an optimized route. It also enables us to find the optimal speed by iteration, which we show later in Section 2.6.5.

2.2.5 Objectives

Similar to manned aviation, the objectives of the flight planning system is to optimize the

- vehicle-battery combination
- 3D flight path
- flight altitude
- ground speed or airspeed

The 3D flight path and the ground speed together gives us a 4D trajectory with timestamps by applying equation (4.24) in Chapter 4. Further adjustment and control on the speed are handled during flight by spline curvature and mode predictive control (MPC), respectively, in Chapter 5.

With the assumptions and objectives in mind, we can now go through the individual blocks in Figure 2.1a.

2.3 Optimal Routing

The first block is an optimal routing engine which takes the red and green information as inputs and compute a 4D trajectory (3D waypoints with timestamps) as output. By assumption 3, 4, and 5, we optimize the route for a single vehicle-battery combination at a single altitude and a fixed speed. The trajectory could be optimized with respect to some cost and constraints. Currently, the engine can compute either minimum-energy or minimum-time trajectories at fixed ground speed or airspeed. In Chapter 4, we will go into details on this block. To solve the optimization problem, we need an efficient algorithm (Section 4.4). And to define the cost and constraints properly, we need good environment data, which we review in Section 4.5. For example, wind can affect the energy consumption of a UAS when it flies in different directions. Therefore, to optimize energy consumption, wind field data is desirable. For constraints, we are mostly interested in avoiding static obstructions such as buildings and mountains. One possible approach is to utilize terrain data. Dynamic obstacles are subjected to change and thus are not considered in the flight planning stage, but we will explore the collision avoidance with manned aircraft during the flight in Chapter 6.

In manned aviation, route selection is based on waypoints called Very High Frequency Omni-directional Range (VOR) and airways [88]. Airline pilots uses commercial software such as Jeppesen MyFlitePlan [55] to plan the flight routes, or 4D trajectories as defined by NextGen and SESAR [33]. In principle, airlines can fly wind-optimal routes, but in reality they prefer to fly a set of verified fixed routes. The main reason is that the airspace rules are complicated, and some of them are unpublished, so traditionally airlines would store routes they knew would be accepted.

Nonevertheless, there is a rich literature on aircraft trajectory optimization in wind. It is generally a two-point boundary-value problems (TPBVPs) [17]. One example is the classical Zermelo problem [138]. It is a minimum-time optimal control problem with a fixed origin and a destination. Other costs can be a combination of time, fuel consumption, and penalty regions [124, 122]. In most cases, the problem includes a dynamical equation relating ground velocity, air velocity, and wind velocity in the horizontal plane.

In general, there are two types of algorithms solving this problem: shooting method and collocation/transcription method [10]. A shooting method example is presented in [124]. The ground speed is assumed constant, and the control variable is the heading angle. By applying Pontryagin's minimum principle, a dynamical equation for the heading angle is derived. By iteratively changing the initial heading angle in a gridded horizontal plane, the optimal trajectory can be found. This process is repeated exhaustively for all available altitudes to determine the optimal flight altitude. The biggest drawback of this approach is that the convergence rate is highly dependent on the initial guess on heading. Secondly, this approach only applies to regular gridded planes, making it difficult to account for obstacles in low-altitude airspace where irregular terrain surfaces are present.

In [91] and [11], the authors use collocation method to solve the continuous-time optimal trajectory problem. In this method, the state is approximated using a polynomial to match the collocation points in a gridded plane. The solution is given by a polynomial expression as a function of time. The major drawback is that wind data has to be parametrized as analytic functions. But parameterizing wind with low-order polynomials cannot capture the spatial irregularity and turbulence nature of small-scale low-altitude wind (Section 1.1). Secondly, it cannot handle irregular terrain meshes. Third, the computation time explodes as the number of collocation points increases.

In addition to the two standard approaches, researchers in [54] tries another algorithm called neighboring optimal control (NOC) to numerically approximate the minimum-time Zermelo solution. The altitude and airspeed are assumed constant. The algorithm is a variation of linear quadratic regulator (LQR), and the solution is given as a time-to-go look-up table. But this approach models wind as augmented wind shear states, which again cannot capture wind field directly in the computation.

The proposed UAS flight planning system give a new approach to address the terrain geometry problem by using elevated digital elevation models (DEM) as feasible flight regions (Section 4.5.1), instead of an idealized 2D gridded plane. It provides a simple way to avoid mapped static obstacles without formulating complex optimization problems. But it also renders the traditional shooting method and collocation methods unusable. Therefore, we explore the Ordered Upwind Method (OUM), an algorithm with native support to triangulated mesh terrain data (Chapter 4). In addition, the FPS utilizes wind data at a fixed altitude AGL to address the spatial complexity of low-altitude wind. The temporal aspect is left as future work. The wind data is used to predict energy consumption (Section 4.5.2) and produce minimum energy and minimum time routes (Section 4.4). To optimize the cruise speed and altitude, we use an exhaustive search approach similar to manned aviation (Section 2.6. The entire routing process can be easily automated, and thus provides an efficient

and flexible approach to address the high traffic volume demand.

2.4 Power Consumption Model

After optimizing the route, we feed the 4D trajectory to a power consumption model (Chapter 3) to estimate the energy demand as a time history of power draw. This is shown in the second block in Figure 2.1a. This model is developed specifically for multi-rotor type of UASs. We find that the energy performance study on multi-rotors is small, but they are the most popular platforms in complex environments. Other types of UASs have more detailed studies. We only focus on the steady-state performance because the model is also used in the cost function inside the optimal routing engine. The model-specific literature review is provided in Chapter 3. In the following, we briefly explain how power model is used in aviation.

Our model is developed based on aviation practice. In manned aviation, the fuel consumption computation relies mostly on publicly available data. A majority of this data is a combination of Eurocontrol's Base of Aircraft Data (BADA), to calculate fuel consumed while airborne, and the International Civil Aviation Organization (ICAO) Engine Exhaust Emissions Data Bank, to calculate fuel consumed on the ground [52]. Aircraft manufacturers can often provide much more accurate performance data, which is utilized by pilots to perform speed optimization. Similarly, in the proposed FPS, we develop a power consumption model for multi-rotor UAS to predict the energy consumption in a given flight trajectory. This model also enables us to optimize altitude and cruise speed. In the sections below, we explain the optimization process briefly.

2.4.1 Altitude Optimization

With the power consumption model, we can further optimize the fly altitude based on wind information. In Chapter 3, we will show that the energy consumption of a UAS is a strong function of wind. In many cases, wind speed goes higher as the altitude increases [113]. Therefore, we want to fly low when against wind and fly high when along wind (Section 3.5.2). Since we assume that we only have a few choice on altitude, we can simply iterate all the available choices to find the optimal one. This approach is adapted directly from aviation [124, 54].

In aviation, safety and fuel consumption are the main concerns on altitude selection. It is selected based on flight levels separated by 1000 feet [124]. Even number of flight levels are assigned to one flight direction, while odd numbers are assigned to the other. Air traffic controllers assigns the cruise altitude to balance the congestion in each flight level. If congestion is not an issue, then altitude can be chosen to minimize fuel consumption [26]. In our work, we don't consider congestion, and thus altitude is chosen to solely minimize the energy consumption.

2.4.2 Speed Optimization

The power consumption model is also important in speed optimization. Given the flight path and altitude, there is an optimal speed minimizing the energy consumption. When flying too slow, the UAS spends more energy to maintain hovering. While flying too fast, the UAS can spend an excessive amount of energy on overcoming aerodynamic drag (Section 3.5.1). We optimize the travel speed by exhaustive search or iteration. Here we provide a high level description of the procedures. The example in Section 2.6 will make it more concrete. To do exhaustive search, we can feed a feasible range of ground speeds and airspeeds to the optimal routing engine to assess the resulting trajectories. To perform iteration, we start the computation in Figure 2.1a with a first speed and obtain a 4D trajectory. Then, we iteratively compute the energy consumption by applying different constant ground speeds along the same trajectory. It gives us an optimal ground speed with minimal energy (See Section 3.5.1). Then, we can use this ground speed to repeat the demand and supply computation in Figure 2.1a to obtain a slightly different 4D trajectory, and compute the optimal ground speed again for this new trajectory. We can repeat this process several times to obtain an almost optimal ground speed for our trip.

The optimal speed problem also exists in aviation. At low speed drag due to lift dominates, while at high speed parasite drag dominates. The fuel burn per unit distance is minimized at an optimal speed. In general, fuel burn is proportional to thrust for turbojet and turbofan engines. In unaccelerated level flights (constant ground speed), we can minimize the thrust T to maximize endurance, or minimize the $C_D/C_L^{1/2}$ ratio to maximize the cruise range. When the ground speed is allowed to change, more fuel saving can be achieved. In [93], the author uses singular optimal control to optimize the speed along a fixed route at constant altitude. For a Boeing 747-400, the fuel saving is about 7% compared to the constant cruise speed case. Currently, our FPS only focus on constant speed optimization. The varying speed case is left as future work.

2.5 Battery Model

Lastly, the energy supply of interest is Lithium polymer batteries, so we feed the power draw history to a battery model to discharge it, as is indicated in the third block of Figure 2.1a. The main difficulty is on estimating the battery capacity, which is a complicated function of current draw, temperature, and charging-discharging cycle [16, 3, 71]. This problem is not the main focus of this dissertation. Therefore, we assume that the battery capacity is accurately estimated. In this section, we provide a simple discharging method. Further development on the energy supply is out of the scope of this dissertation. The final output is the remaining state-of-charge, or the percentage of energy capacity left, inside the battery. We go through this process for all the available vehicle-battery combinations, or only for a subset of the combinations filtered by some heuristics, and we will end up with a list of remaining SOCs for each combination. They can be either positive or negative. If the

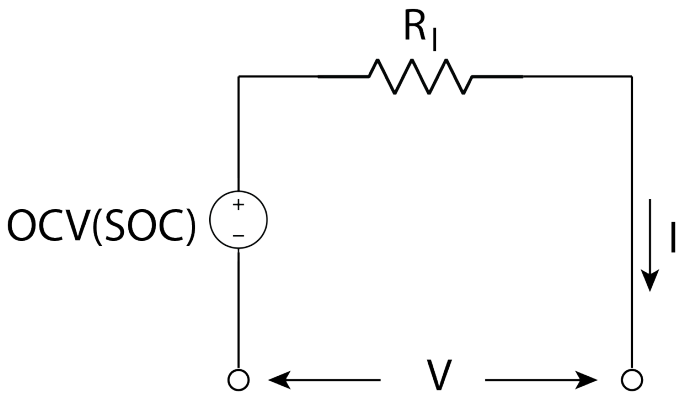


Figure 2.2: A simple equivalent circuit model of a LiPo battery.

remaining SOC is below a certain threshold, we say this combination is infeasible. The more positive the number, the better the contingency margin. In the remaining of this section, we will describe the battery discharging process with a equivalent circuit model (ECM).

Equivalent Circuit Model (ECM) is a simple approach to capture LiPo battery discharging behaviors [41]. Figure 2.2 is the model we use. The voltage source is the open circuit voltage (OCV), which is a function of SOC. The battery has an internal resistance of R_I. The battery output voltage and current are V and I, respectively. Equation (2.2) describes battery charging/discharging. Q is the nominal capacity of the battery. The input is current $I(t)$.

$$\dot{SOC}(t) = \frac{1}{Q} I(t), \quad SOC(0) = SOC_0 \tag{2.2}$$

To compute $I(t)$ at the current time, we can solve the following quadratic equation. The solution has two values, but only the proper value is chosen. It is obvious because we know that the voltage V is about $12V$.

$$P(t) = I(t)V(t) = I(t)\left(OCV(SOC(t)) - R_I I(t)\right)$$

The remaining state-of-charge SOC_{remain} is the final value of the solution in equation (2.2).

2.6 Flight Planning Example

In this Section, we illustrate how to use the UAS flight planning system with small examples. The intent is to give the readers an idea on the computation flow inside the FPS. We developed and host a FPS repository in [131]. Interested readers can follow the instructions therein to repeat the flight planning process in the examples.

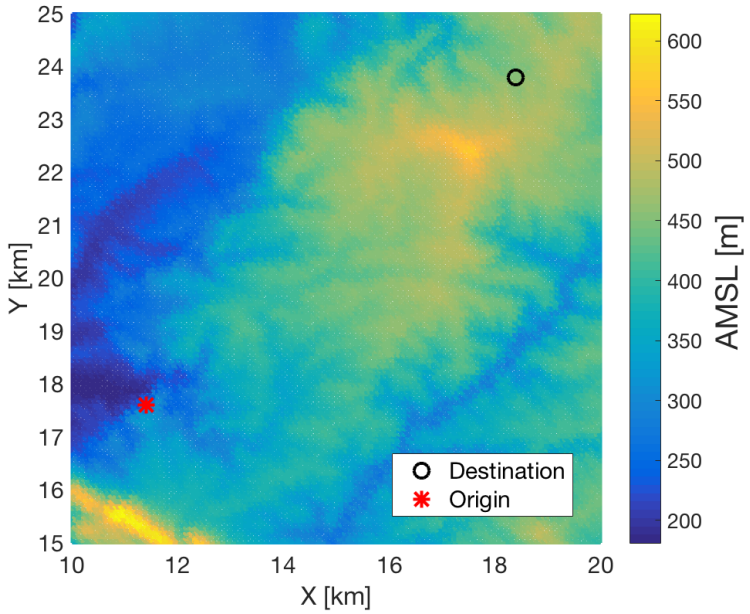


Figure 2.3: Terrain data at Perdigao in Portugal.

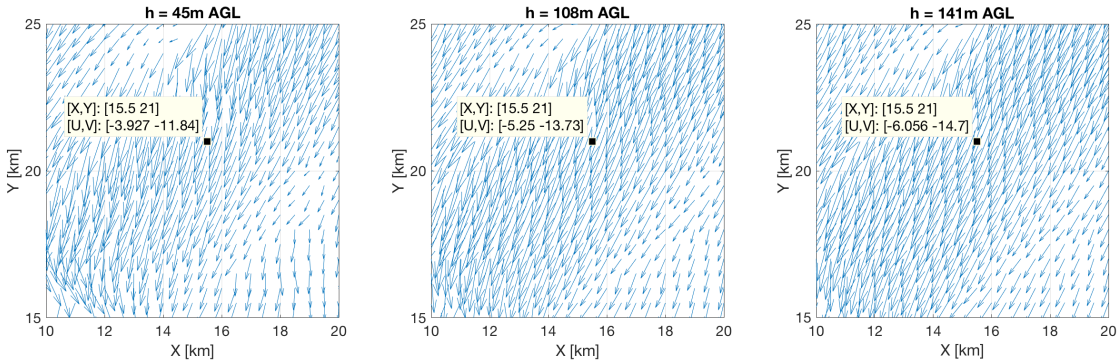


Figure 2.4: Wind data at Perdigao in Portugal at 3 different altitudes AGL at 00:00 UTC, May 19, 2017.

2.6.1 Specify the Fixed Inputs

As per Figure 2.1a, we start by specifying the fixed inputs. The terrain data is a portion of the dataset presented in Section 4.5.1. We visualize it in Figure 2.3. Each data point is specified by a horizontal coordinate (X, Y) in km and an altitude above mean sea level (AMSL) as indicated by the heat map color. The altitude varies from $180m$ to $620m$ AMSL. The example origin is shown in a red star, and the destination is a green circle.

The wind data at different altitudes is shown in Figure 2.4. We obtain the data from the Environmental Fluid Mechanics and Hydrology Group (EFMH) at UC Berkeley [32]. The datatips indicate the horizontal wind components. In general, the wind magnitude becomes larger as the altitude increases. At location $(15.5, 21.0)km$, the wind speeds are $12.4m/s$, $14.7m/s$ and $15.9m/s$ at $45m$, $108m$, and $140m$ AGL, respectively. For simplicity, we will assume the UAS can fly at $50m$, $100m$, and $150m$ AGL, and we use the closest corresponding wind field. Section 4.5 has a more detailed description of the terrain and wind data.

2.6.2 Specify the Variable Inputs

The UAS we use in this example is a 3DR IRIS+. It comes with a 3-cell $5000mAh$ LiPo battery. The vehicle frame without battery is $11N$, and one battery weighs $3N$. The internal resistance of the battery is unknown, but we assumed a reasonable value of $30m\Omega$ from [130]. The battery has a discharge curve shown in Figure 2.5 with the following parameters.

$$mg_{batt} = 3N \qquad R_I = 30m\Omega \qquad Q = 5000mAh \qquad SOC(0) = 1.00$$
$$mg_{UAS} = 11N \qquad SOC_{min} = 0.10$$

We can connect multiple batteries together in parallel to increase the capacity and obtain different vehicle-battery combinations. In this example, we try IRIS+ with 1, 2, 3, 4 and 6 batteries. Suppose we connect N batteries together, then the total vehicle weight and total battery internal resistance are

$$\begin{aligned} mg_{total} &= Nmg_{batt} + mg_{UAS} \\ R_{I,total} &= R_I/N \\ Q_{total} &= NQ \end{aligned}$$

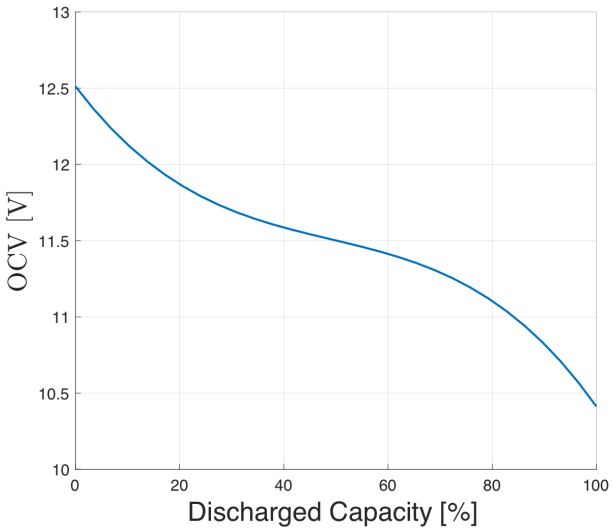


Figure 2.5: OCV as a function of the discharged capacity / SOC.

2.6.3 Path Optimization

To compute the energy demand, we first elevate the terrain surface to the altitude of choice. In this example, we start with $150m$ AGL. The terrain surface is smoothed by a Gaussian filter to avoid sharp changes in altitude. This is necessary to avoid destabilization of the vehicle. Further modifications to the terrain surface can be applied, but we will not go further in this dissertation.

The wind field we use here is the third one in Figure 2.4. The wind field has a maximum wind speed of $16m/s$. In Chapter 3, we show that the maximum airspeed of an IRIS+ is about $20m/s$. Since the vehicle is flying against wind, to make sure that we can fly to the destination, we choose a first ground speed of $5m/s$. Further discussion on optimizing the speed with constraints is presented in Sections 2.6.5 and 2.6.6.

Now that all six inputs in Figure 2.1a are specified, we can feed this information to the optimal routing engine developed in Chapter 4 to minimize either energy consumption or trip time. In this example, we choose to minimize the energy consumption at the fixed initial ground speed (Section 4.3.6). The first optimal path is the blue line shown in Figure 2.6.

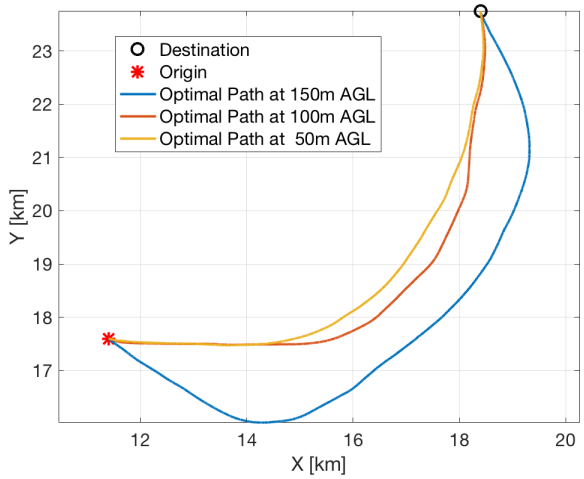
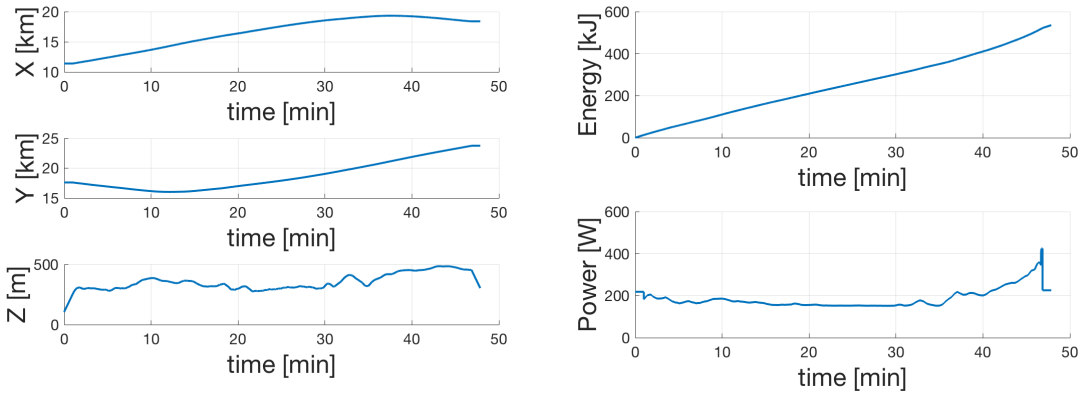


Figure 2.6: The minimum energy path generated from the optimal routing engine in Chapter 4.

Since we know the ground speed, we can convert the path to a 4D trajectory by attaching timestamps using equation 4.24. The minimum energy trajectory is shown in Figure 2.7a. The trajectory is then feed into the power consumption model to generate a time history of power draw in Figure 2.7b. The constant power at the beginning and the end are during take-off and landing, respectively. The optimal energy spent on this trip is $538.2kJ$ at the first ground speed.



(a) 4D trajectory. (b) power consumption time history.

Figure 2.7: The initial result from the energy-optimal routing engine and power consumption model.

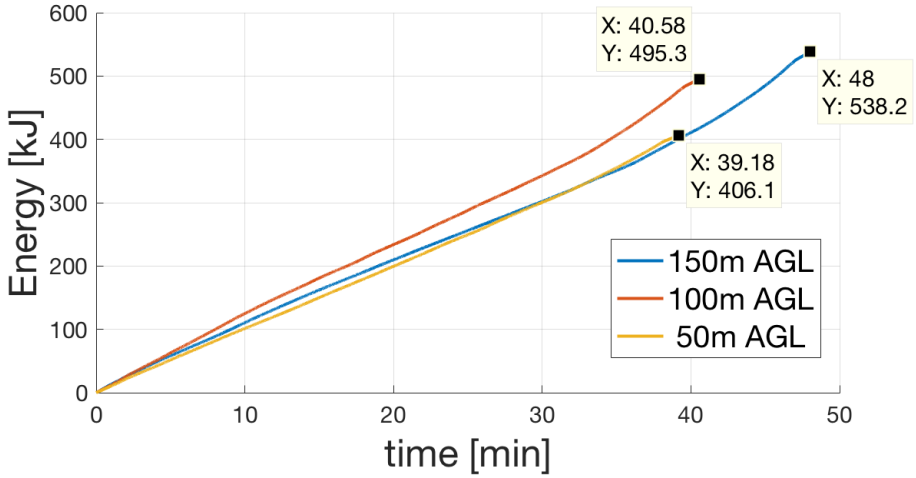


Figure 2.8: The energy consumption comparison at different altitudes.

2.6.4 Altitude Optimization

Now we can optimize the energy consumption further by repeating the same computation at different altitudes. We repeat the path optimization computation again for the other two altitudes. The optimal paths at the first ground speed $5m/s$ are indicated in Figure 2.6. Figure 2.8 shows that the energy consumption at $100m$ and $50m$ AGL are $495.3kJ$ and $406.1kJ$, respectively. The optimal altitude at the first ground speed is thus $50m$ AGL. For this particular wind field, as the altitude becomes lower, the energy consumption becomes smaller. It is because the IRIS+ is traveling against wind, and flying lower saves energy. The

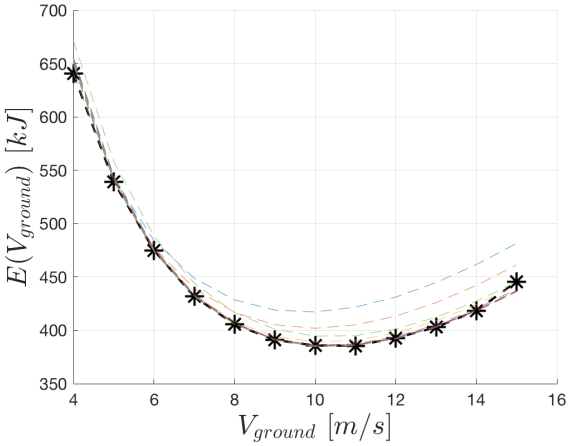
energy savings are quite significant because at this airspeed the aerodynamic drag reduces dramatically as the altitude gets lower (Chapter 3). If we were traveling in the opposite direction, flying higher would be preferable. But flying lower is more risky because the chance of hitting obstacles increases. We may further quantify the cost on energy and risk to select an altitude, but the choice is more of an art than science. We will not address this problem in this dissertation.

In this example, the result above suggests that we can pick an altitude and proceed to the next step: speed optimization. But in general, we don't know the optimal altitude for this speed is also optimal for a different speed. Instead of making assumptions on the wind field structure, in this dissertation, we address this issue by performing exhaustive search, meaning that for each available altitude, we search for an optimal speed over a reasonable range of ground speed and airspeed. In the next section, we will discuss two approaches to perform speed optimization.

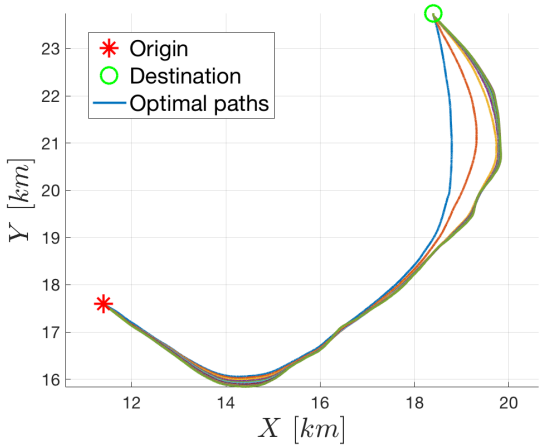
2.6.5 Speed Optimization

Next, we would like to optimize the ground speed or airspeed, and choose the one giving a lower energy consumption or shorter trip time. We have two approaches to do this optimization: exhaustive search or iteration. We start by focusing on optimizing ground speed at $150m$ AGL. The computation is repeated for different airspeeds and other altitudes.

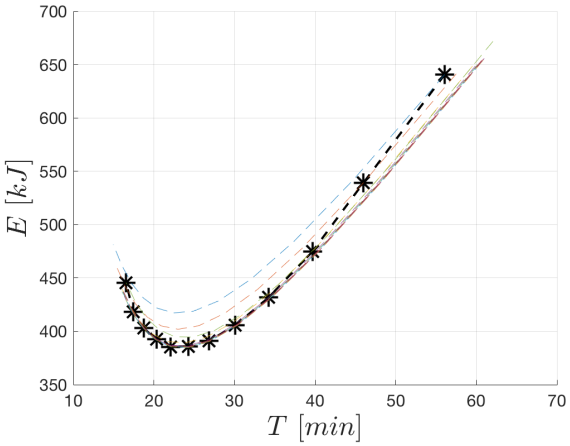
Approach 1 - Exhaustive Search We can optimize speed by exhaustive search (Figure A.1). Over a range of ground speeds, we compute the corresponding optimal trajectories. The black stars in Figure A.1a presents the minimum energy E in kJ for a discrete range of ground speeds from $4m/s$ to $15m/s$. The dark dashed line connects the optimal choices together. Figure A.1b shows the variations of optimal paths among the range of ground speeds. For each path, there is a E-V_{ground} curve in Figure 2.9a. The bottom of the parabola gives the optimal ground speed for the path that minimize energy. In this case, the path varies gradually. In general, there can be a sudden change in the paths after a certain ground speed threshold. This behavior is observed in other origin-destination pairs. See Appendix A. In Figure 2.9a, the thin colored dashed lines show the E-V_{ground} relationship for a given optimal path. If we pick a path not at the corresponding optimal ground speed, we could spend more energy. This is for $150m$ AGL and a $3N$ battery weight. Finally, each E-V_{ground} curve in Figure 2.9a becomes an energy versus trip time (E-T) curve in Figure 2.9. In this case, the trip time that minimizes the energy consumption is at $T = 22.1min$.



(a) Optimal energy consumptions.



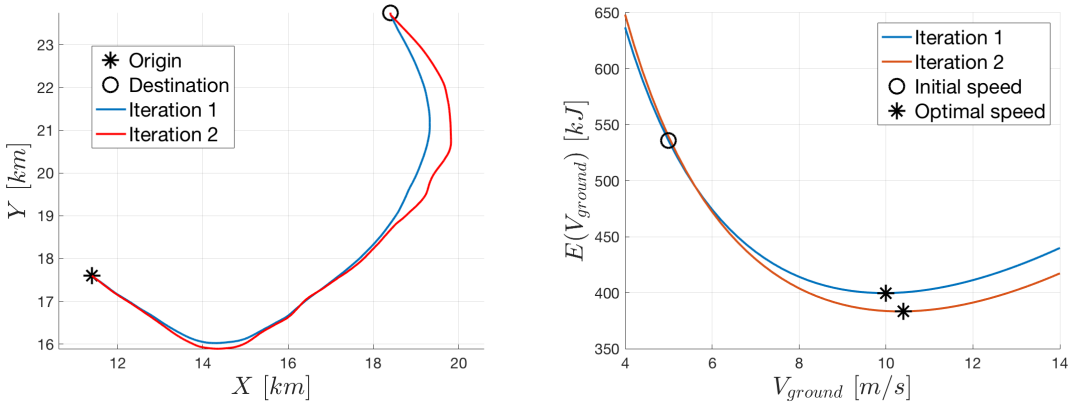
(b) The corresponding optimal paths.



(c) The corresponding trip times.

Figure 2.9: Repeat the iteration process for a set of ground speeds.

Approach 2 - Iteration: We can also optimize speed by iterating between ground speed and flight paths. Figure 2.10 shows the process. The black circle in Figure 2.10b shows the E-V_{ground} point of the first ground speed $5m/s$. We generated a minimum energy flight path, or the blue line in Figure 2.10a. Now we will fly this path at various ground speed and plot the energy consumptions. The result is shown in the blue line in Figure 2.10b. The black star indicates the optimal ground speed is $V_{ground,1} = 10.0m/s$ for this path. With this updated ground speed, we then compute the minimum energy path again (red line in Figure 2.10a), and re-optimize the ground speed (red line in Figure 2.10b). In this case, we get almost similar result in the second iteration, $V_{ground,2} = 10.4m/s$, so we can stop here. In general, we would perform multiple iterations until the speed converges.

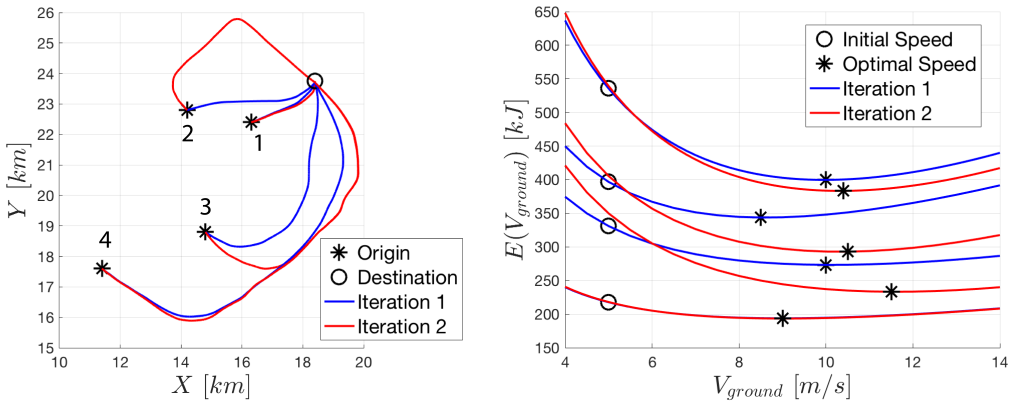


(a) Flight path iterations.

(b) Ground speed iterations.

Figure 2.10: Search for the optimal ground speed on the flight paths at $150m$ AGL.

We repeat the same iteration process for some other origins and observed different convergence behaviors. Figure 2.11 shows the iteration results. For paths at origins #1 and #4, two iterations are enough, but for paths at origins #2 and #3, the second iterations give quite different paths (Figure 2.11a). To obtain better results, more iterations are needed. In many cases, we expect convergence in a few iterations, but we are unable to prove the convergence proof for all cases in general. This is a future direction of research.



(a) Flight path iterations. (b) Ground speed iterations.

Figure 2.11: Similar convergence result for the other origins in Figure 2.3.

2.6.6 Speed and Arrival Time Constraint

There are times when we cannot achieve the optimal ground speed. In our example, imagine that the FAA regulates that the maximum ground speed a UAS can travel is $8m/s$, or somehow we know that under this wind condition, the maximum ground speed the UAS can reach is $8m/s$. In either case, we have to generate the minimum energy trajectory with a lower ground speed because the optimal speed does not comply with regulations or vehicle performance.

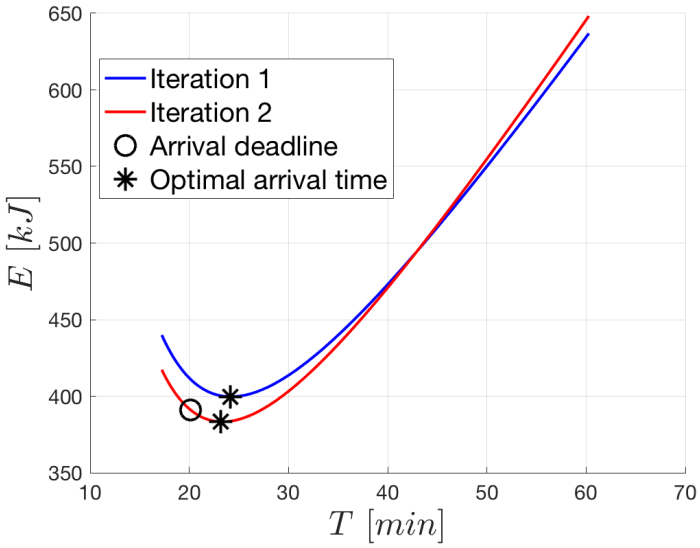


Figure 2.12: The energy consumption versus travel duration in two iterations.

There are also times when we have to fly faster than the optimal ground speed to meet certain deadlines. For example, if the package delivery time is no later than half an hour from now, but at the optimal ground speed, we will be late by 3 minutes. In this case, we can change the x-axis in Figure 2.10 to be the trip time T in minute. The updated plot is shown in Figure 2.12. If the arrival time had to be before 20 minutes, as is indicated by the black circle, then we have to sacrifice $8kJ$ more energy to avoid late arrival. In this case, even though the red path labeled "Iteration 2" is not the minimum energy trajectory for this arrival time, it is a good enough approximation without introducing extra complications.

In addition, the slope of this curve indicates a trade off between time and energy. At the black circle, the slope is $-5.76kJ/minute$, meaning that one minute delay saves $5.76kJ$ of energy. If we know that the monetary penalty on late arrival is $\$\beta/minute$ and the energy cost is $\$\alpha/kJ$, we can formulate a cost function similar to equation (2.3) to optimize the cost of this trip.

$$J = \alpha E + \beta t \tag{2.3}$$

This concludes the energy demand computation for $150m$ AGL and $3N$ battery example. In the next section, we will give a complete exhaustive search example.

2.6.7 Energy Demand: A Complete Exhaustive Search Example

As was mentioned at the beginning of Section 2.1, we focus on optimizing paths at both fixed ground speed and airspeed. In the flight planning example above, we computed the minimum energy trajectory at a fixed ground speed. We can also minimize trip time at a fixed ground speed. In addition, we can do minimum-energy or minimum-time trajectories at various given airspeeds. The two objectives with given airspeeds turn out to be equivalent. For more information on these four cases, refer to Chapter 4.

To explore all possible cases, we perform an exhaustive search for all available altitudes, ground speeds, and airspeeds, with trajectories minimize energy or time. For five different vehicle-battery combinations (IRIS+ with 1, 3, 3, 4, 6 batteries), we do an exhaustive search on all four formulations and three altitudes, and we end up with an E-T diagram in Figure 2.13. The legend in this figure does not distinguish the vehicle-battery combinations due to space limitation. To see the plot for each individual combination, refer to Appendix B.

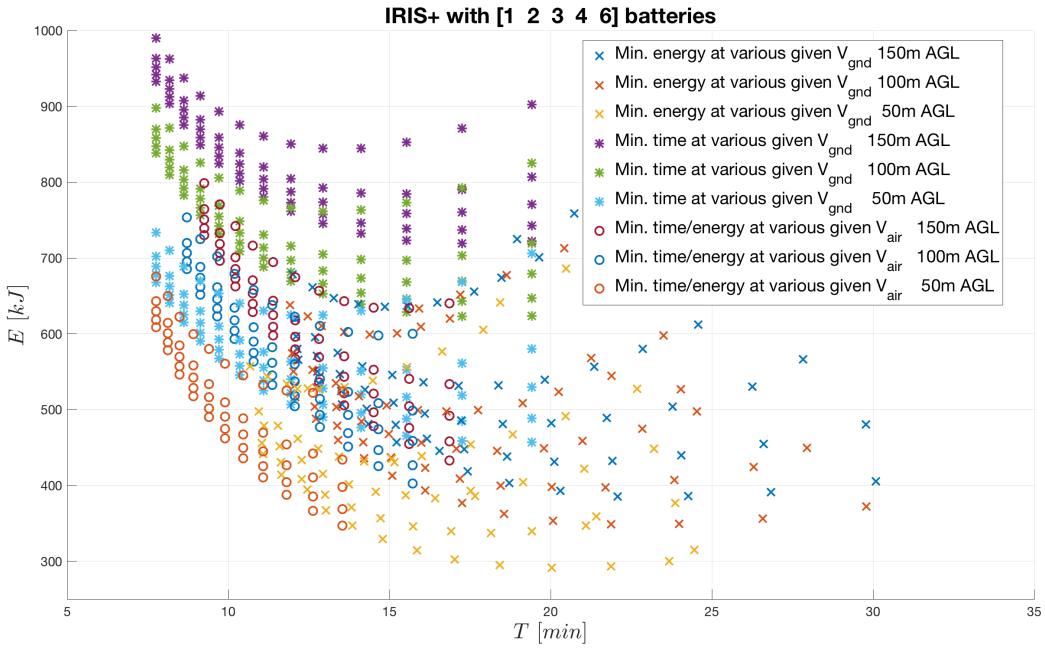


Figure 2.13: The exhaustive search result for a range of altitudes, ground speeds, and airspeeds for an IRIS+ with different number of batteries.

In general, we would like the energy to be as low as possible. The main constraint would be the arrival time deadline, which can push the optimal choice to the left. For our wind field, the energy consumption increases with altitude, as observed earlier. The minimum-energy-at-various-given-ground-speed case gives results very similar to the two cases at given airspeed, and they give the optimal energy-time trade off. On the other hand, the minimum-time-at-various-given-ground-speed case gives results far from optimal. Lastly, as the number of batteries increases, the energy consumption becomes larger, and this difference increases as the trip time becomes longer.

2.6.8 Energy Supply: Battery Discharging

To see whether the energy supply is sufficient, we discharge the battery by feeding the power draw time history to the battery model (Section 2.5). Without any speed and arrival time constraints, we can use Figure 2.13, or Appendix B, to select the most efficient vehicle-battery combination. We start with the 1-battery combination. If it is not feasible, we increase the number of batteries. At $150m$ AGL, the discharging time histories for an IRIS+ with different number of batteries at their optimal ground speeds are shown in Figure 2.14. In the 1-battery and 2-battery cases, the final SOC is less than the SOC_{min} threshold. Therefore, these two combinations are infeasible. The other combinations are feasible. The 3-battery combination has a SOC_{remain} of $0.215 > 0.10$ and therefore is the most energy-

efficient option. As the number of battery increases, the SOC margin increment becomes less and less, and the discharging time history between the 6-battery combination is almost identical to the 4-battery case. The final SOC in the former combination is larger only because the trip time is shorter. Therefore, after a certain threshold, adding more batteries does not increase the cruise range.

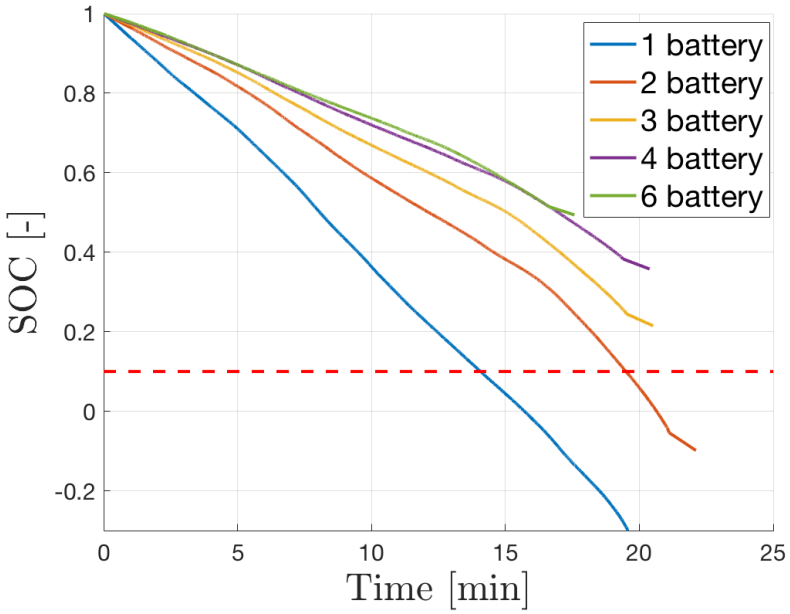


Figure 2.14: Battery discharging time histories at $150m$ AGL.

2.6.9 Arrival Time Constraint

To explore the optimal options under arrival time constraints, we assume three trip time limits, namely $20min$, $15min$, and $10min$. Again, by using Figure 2.13 or Appendix B, we obtain the results in Table 2.1. In the SOC_{remain} column, the red dash indicate an infeasible option, while the blue number is the first feasible option.

For an arrival time limit of 20 minutes, the trip results are the same as the unconstrained case above. It is because the optimal speeds all satisfies the time deadline. The optimal choice would be IRIS+ with 2 batteries at $50m$ AGL flying at a ground speed of $12m/s$.

If we reduce the trip time limit to 15 minutes, the optimal speeds are all shifted to $V_{ground} = 15m/s$. The remaining SOC for the 2-battery case 0.11, barely above the threshold. If we would like to consider contingency, then we may choose the 3-battery combination with $SOC_{remain} = 0.37$, instead. Sizing up the battery also reduce the actual trip time. The 6-battery combination gives an optimal trip time of 12.8 minutes, while all the other combinations are about 15 minutes. But without contingency, the optimal choice would be an IRIS+ with 2 batteries at $50m$ AGL flying at a ground speed of $15m/s$.

When the trip time limit is 10 minutes, the optimal speed becomes $V_{air} = 25m/s$ for all combinations. In this speed, the first feasible option is IRIS+ with 3 batteries, giving a remaining SOC of 0.18. In this case, sizing up the battery does not reduce the trip time because all unconstrained optimal speeds give longer trip times. Without considering contingency, the optimal choice would be an IRIS+ with 3 batteries at $50m$ AGL flying at an airspeed of $25m/s$.

Table 2.1: Optimal choice from FPS at different trip time limits.

Battery Capacity [mAh]	Total Vehicle Weight $[N]$	Altitude $[m$ AGL$]$	Optimal Speed $[m/s]$	Energy Demand $[kJ]$	Trip Time $[min]$	SOC_{remain} $[-]$
$20min$ Trip Time Limit						
$5,000$	14	50	11 (V_{ground})	291.7	20.0	–
$10,000$	17	50	12 (V_{ground})	337.4	18.2	0.173
$15,000$	20	50	13 (V_{ground})	383.3	16.4	0.385
$20,000$	23	50	14 (V_{ground})	430.4	15.15	0.487
$30,000$	29	50	15 (V_{ground})	526.5	12.8	0.584
$15min$ Trip Time Limit						
$5,000$	14	50	15 (V_{ground})	329.4	14.8	–
$10,000$	17	50	15 (V_{ground})	356.7	14.7	0.111
$15,000$	20	50	15 (V_{ground})	391.6	14.6	0.369
$20,000$	23	50	15 (V_{ground})	431.6	14.2	0.484
$30,000$	29	50	15 (V_{ground})	526.5	12.8	0.584
$10min$ Trip Time Limit						
$5,000$	14	50	25 (V_{air})	462.2	9.9	–
$10,000$	17	50	25 (V_{air})	474.7	9.9	–
$15,000$	20	50	25 (V_{air})	490.5	9.9	0.176
$20,000$	23	50	25 (V_{air})	509.9	9.9	0.375
$30,000$	29	50	25 (V_{air})	560.6	9.9	0.553

In the next two chapters, we will discuss the power consumption model and optimal routing engine used in the examples.

Chapter 3

Power Consumption Model

3.1 Introduction

In this chapter, we develop a power consumption model to estimate the energy demand in a UAS flight. We first review the relevant literature in Section 3.1.1 and summarize our contribution in Section 3.1.2. We present the model in Section 3.2, then we perform system identification to obtain parameters for a 3DR IRIS+ quadrotor UAS. The model is validated in Section 3.4. Finally, we simulate some insightful aviation scenarios with this model in Section 3.5.

3.1.1 Literature Review

In this section, we briefly review existing power consumption models for a multirotor UAS. In Chapter 2, we saw that energy demand in a commercial flight is computed using performance data such as the Eurocontrol's Base of Aircraft Data (BADA) and International Civil Aviation Organization (ICAO) Engine Exhaust Emissions Data Bank. Similar to airline fuel loading, we need to model the power consumption of UAS. We focus on multi-rotor type of UAS for two reasons. First, multi-rotor is the most popular type of UAS platform in complex environments. It is capable of vertical-take-off-and-landing, and hover to stop in emergency situations. Second, the literature on modeling multi-rotor energy consumption is small.

First, most of the current studies are empirical and data driven. In [6], the author studies the energy consumption of multi-rotors with different numbers of rotors. They study rotor force as a function of rotor speed, and measure the power consumption of a single rotor. In [105], the author decomposes the power consumption of a PCB quadrotor into avionic and payload power. Power is studied as a function of thrust only, and flight time is studied as a function of weight. In [83], the author has collected power consumption data for a variety of motors and measures the power draw as a function of thrust. This is also the approach taken in commercial aircraft because modeling the aircraft performance is difficult.

Second, multi-rotor-related power consumption models are either incomplete or only developed for highly-customized platforms. In [45], the power components of a quadrotor was studied in the context of quadrotor dynamics. But the power components presented, such as induced and profile power, are only valid at low airspeed ($< 5m/s$). The author in [96] develops a theoretical power consumption model for a customized research platform - called small convertible UAV. The aerodynamic forces were separated into thrust, in-plane force, and in-plane torque. An analytical model with six non-dimensional parameters is identified experimentally with UIUC wind tunnel measurements. The power model accounts for both thrust and drag in different vehicle configurations.

3.1.2 Contributions

We adopt a similar modeling approach but based on helicopter theory [57, 68]. In comparison to the empirical approach, the analytical approach provides more insight and is able to predict flight behaviors in untested situations. We will show this advantage in Section 3.4.2. For hovering, ascending/descending and low-speed flights, we find that helicopter models gives a good fit to our experimental data. However, at high speed flights, existing models such as the Glauerts induced velocity model [68] could not fit the data properly due to the 4-rotor configuration of a quadrotor. Therefore, we introduces some modifications to existing models to improve the model accuracy. The modification is discussed at the end of Section 3.2.1. A comparison on the accuracy is presented in Section 3.3.3. Lastly, we performed a statistical test to show that the identified model parameters are robust to re-sampling at the end of Section 3.3.3.

3.2 The Proposed Power Consumption Model

We focus on estimating the steady-state energy consumption of a multirotor UAS in long trips. The flight paths consists of a vertical-take-off segment, an arbitrary cruise segment, and a vertical-landing segment (Figure 3.1). The steady-state assumption is reasonable because accelerating and decelerating maneuvers only account for a small portion of missions. Under this assumption, the external forces are in static equilibrium, yielding zero net acceleration. This avoids a dynamic model and renders a static map from commanded speed and thrust to energy. The model can also capture the impact of wind on energy consumption, but it requires wind data as a model input. We measured airspeed experimentally with a pitot tube on board the UAS.

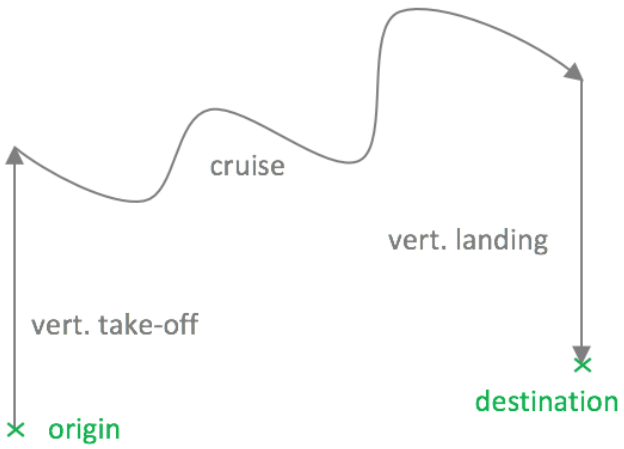


Figure 3.1: The definition of a trip.

We derive the power consumption model by treating the quadrotor as a single propeller. This modeling limitation is imposed by the available power sensors on board most quadrotors. Typically, a power sensor can only report a total current draw, or a total power, for all motors. Motor rotational speeds in revolutions-per-minute (RPM) may be available, but RPM readings can only be an indirect indication of the power distribution among motors. A more detailed multi-propeller model cannot be calibrated.

Our model decomposes the consumed power into three components, namely induced power (P_i), profile power (P_p), and parasite power (P_{par}) (Figure 3.2). The induced power (in blue) produces thrust by propelling air downward. The profile power (in yellow) overcomes the rotational drag encountered by rotating propeller blades. The parasite power (in red) resists body drag when there is relative translational motion between the vehicle and wind.

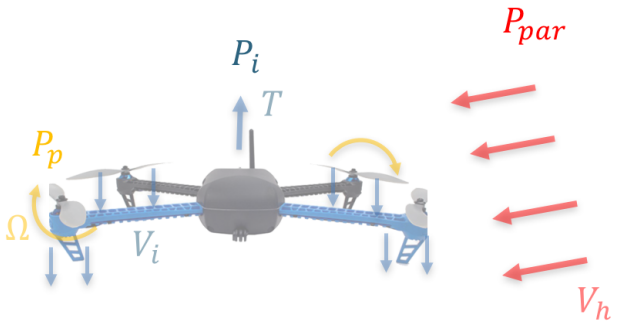


Figure 3.2: Power consumption components of a multi-rotor UAS.

In this section, we present the model from first principles. The model is similar to the one presented in [57] with minor modifications. The model is summarized at the end to list the parameters to be identified experimentally.

3.2.1 Induced Power

We base our modeling of induced power consumption on actuator disk theory [57]. Here we are only concerned with the vertical flow, which is assumed to be uniformly distributed over the propeller disks, inviscid, incompressible, and irrotational. Figure 3.3 shows the control volume (CV) of the air mass around the quadrotor during vertical flight. The quadrotor's propellers divide the air mass into two parts. The boundaries of interest are indicated by numbers $i = \{0, 1, 2, \infty\}$. Let

- p_i be the pressure at each boundary.
- V_i be the vertical speed at each boundary.
- T be the total thrust.
- A be the total propeller area.
- ρ be the density of air.
- $\dot{m}$ be the mass flow rate.

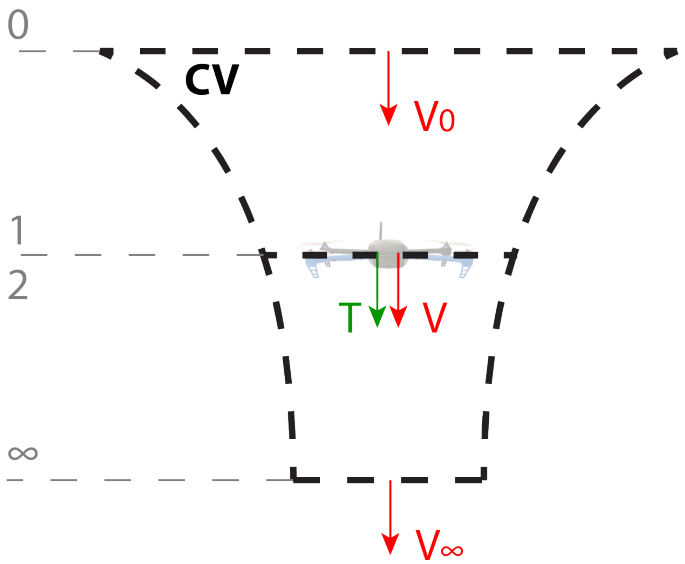


Figure 3.3: Control volume (CV) of the air mass of interest.

We also define $V := V_1 = V_2$ and $p := p_0 = p_\infty$.

After some simple derivations, we obtain the analytical expression for the idealized induced power ((3.1)). See Appendix C for details.

$$\begin{aligned} P_i = TV &= T\left(\frac{1}{2}\left(V_\infty + V_0\right)\right) \\ &= T\left(\sqrt{\frac{T}{2\rho A} + \left(\frac{V_0}{2}\right)^2} + \frac{V_0}{2}\right) \end{aligned} \tag{3.1}$$

Since the CV is moving together with the quadrotor, and air is at rest at boundary 0, the vertical speed of the vehicle V_z is the same as the speed at boundary 0, or $V_z := V_0$. In addition, to account for the deviation between the idealized uniform air flow and actual flow, as well as motor efficiency, we multiply the equation by a scaling factor $k_1 \in [0, 1]$ to the whole expression, which yields the first equation in (3.8), with k_2 defined in Table 3.1.

Lastly, note that rotor efficiency is generally improved for helicopters in horizontal flights. This effect is called translational lift [40]. In our experiments, we observed small but noticeable power reduction for such flights. However, as was mentioned before, the Glauerts induced velocity model [68] commonly used in helicopters could not fit the data properly due to the 4-rotor configuration of a quadrotor. Therefore, we modeled it by introducing a pseudo-lift force (L) term defined by equation (3.2), shown as a gray-dashed arrow in Figure 3.5. Equation (3.2) is derived by treating the whole UAS as an airfoil, similar to a fixed wing aircraft. We further assumed that L is only a function of airspeed (V_{air}), and set the coefficient c_6 to zero, to further simplify the thrust calculation described in Section 3.2.4.

$$\begin{aligned} &L = c_5(V_{air}cos\alpha)^2 + c_6 T \\ &c_5 := Ncc_l\rho R/4, \quad c_6 := k_3 Ncc_l\rho R^3/6 \approx 0 \end{aligned} \tag{3.2}$$

where c_l is the lift coefficient.

3.2.2 Profile Power

We derive the profile power consumed by a rotating rotor blade from blade element theory. Figure 3.4 shows an infinitely small section of a full rotor blade of radius R at a radial location $r < R$. Then the linear speed of the blade at location r for rotor $i = \{1, 2, ..., M\}$ is $V_i(r) = \Omega_i r$, with Ω_i and M being the angular speed of each rotor and the total number of rotors, respectively. Assuming small angles, we can obtain the profile power for the i^{th} rotor in hover mode by integrating the drag force $F_{p,i}$ along the blades.

$$\begin{aligned} P_{p,hover,i} &= N\int_0^R dD = N\int_0^R F_{p,i}V_i(r)dr \\ &= N\int_0^R \left(\frac{1}{2}c_d\rho\left(\Omega_i r\right)^2 c\right)\left(\Omega_i r\right)dr \\ &= \frac{Ncc_d\rho R^4}{8}\Omega_i^3 \end{aligned} \tag{3.3}$$

where

- N is the total number of blades in a single propeller.
- c_d is the drag coefficient of the blade.
- c is the blade chord width.

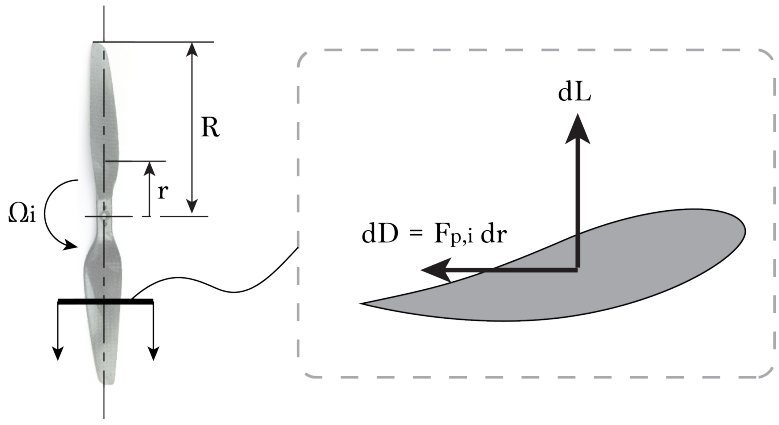


Figure 3.4: Diagram of a quadrotor propeller, with a section view of a blade showing the infinitesimal forces.

In horizontal flight, a similar procedure shows that the profile power becomes

$$\begin{aligned} P_{p,i} &= P_{p,hover,i}(1+\mu_i^2) \\ \mu_i &= \frac{V_{air} cos\alpha_i}{\Omega_i R} \end{aligned} \tag{3.4}$$

where

- μ_i is the advance ratio for propeller i.
- V_{air} is the horizontal airspeed.
- α_i is the angle of attack of the propeller disk i.

For a multi-rotor UAS, it is common to assume that the thrust generated is proportional to the angular speed squared, or $T_i = k_3\Omega_i^2$, where k_3 is a scaling factor converting from rotor angular speed Ω to thrust T. In addition, all angles of attack are identical, or $\alpha := \alpha_i$, for $i = \{1, ..., M\}$. Then the total profile power is

$$\begin{aligned} P_p &= \sum_{i=1}^{M} P_{p,i} = \sum_{i=1}^{M} \left(\frac{N c c_d \rho R^4}{8} \Omega_i^3 (1+\mu_i^2) \right) \\ &= \sum_{i=1}^{M} \left(\frac{N c c_d \rho R^4}{8} \left(\Omega_i^3 + (V_{air} cos\alpha_i / R)^2 \Omega_i \right) \right) \\ &\approx c_2 T^{3/2} + c_3 (V_{air} cos\alpha)^2 T^{1/2} \end{aligned} \tag{3.5}$$

The last approximation lumps all M rotors into a single rotor. Hence the second equation in (3.8), with c_2 and c_3 defined in Table 3.1. From [57], the second term in (3.5) contributes very little, so we assume $c_3 = 0$ empirically, to simplify the identification process.

3.2.3 Parasite Power

The parasite power is obtained by assuming that the body drag (D) is proportional to airspeed (V_{air}) squared.

$$P_{par} = DV_{air} = \frac{1}{2} C_d \rho A_{quad} V_{air}^3 \tag{3.6}$$

where

- $D = c_4 V_{air}^2$ is the drag force.
- C_d is the drag coefficient of the vehicle body.
- A_{quad} is the equivalent cross-sectional area of the vehicle when against wind. We assumed it to be a constant.

Equation (3.6) is re-written and summarized in the third equation in (3.8), with c_4 defined in Table 3.1. In general, C_d and A_{quad} are complex functions of the vehicle geometry and flight direction, making c_4 hard to identify.

3.2.4 External Forces at Steady State

In all the experiments, we only extract data when the UAS is at steady state (in force equilibrium). Figure 3.5 shows all the external forces, where

- L is a lift term introduced to model the translational lift effect.
- D is the parasite drag.
- α is the angle of attack.

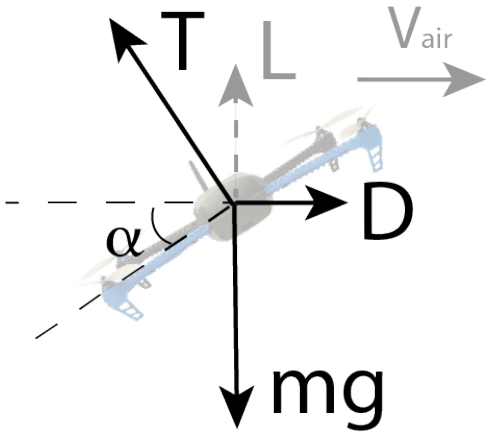


Figure 3.5: The UAS is under force equilibrium in horizontal and vertical directions.

Therefore, the thrust at steady state is computed from

$$T = \sqrt{(mg - L)^2 + D^2} \tag{3.7}$$

Since the lift L is a function of thrust T, as is shown in equation (3.2), we assumed c_6 to be zero (see Section 3.2.1), to avoid solving for thrust T with a complicated quadratic equation. When the UAS is hovering or ascending/descending slowly, the thrust reduces to gravity, or $T = mg$.

3.2.5 Model Summary

The power consumption model is summarized in equation (3.8). The analytical expressions of the parameters are shown in Table 3.1.

$$\begin{aligned} P_i(T, V_z) &= k_1 T \left[\frac{V_z}{2} + \sqrt{\left(\frac{V_z}{2}\right)^2 + \frac{T}{k_2^2}} \right] \\ P_p(T, V_{air}) &= c_2 T^{3/2} + c_3 (V_{air} cos\alpha)^2 T^{1/2} \\ P_{par}(V_{air}) &= c_4 V_{air}^3 \end{aligned} \tag{3.8}$$

$$\begin{aligned} V_{air} &= ||\boldsymbol{V}_{air}|| = ||\boldsymbol{V}_{ground} - \boldsymbol{V}_{wind}|| \\ T &= \sqrt{(mg - (c_5 (V_{air} cos\alpha)^2 + c_6 T))^2 + (c_4 V_{air}^2)^2} \end{aligned}$$

where

- $\boldsymbol{V}_{air}, \boldsymbol{V}_{ground}, \boldsymbol{V}_{wind}$ are the horizontal air velocity, ground velocity, and wind velocity, respectively.

- $k_1, k_2, c_1, c_2, c_4, c_5$ are parameters to be identified in experiments.

When hovering ($V_z = 0$), the induced power is reduced to

$$P_{i,hover}(T) = \frac{k_1}{k_2} T^{3/2} \triangleq c_1 T^{3/2} \tag{3.9}$$

Table 3.1: Analytical Expressions for the Parameters

parameter	analytical expression
k_1	constant $\in [0, 1]$
k_2	$\sqrt{2\rho A}$
c_1	k_1/k_2
c_2	$k_3^{1.5} N c c_d \rho R^4/8$
c_3	$k_3^{0.5} N c c_d \rho R^2/8 \approx 0$
c_4	$C_d \rho A_{quad}/2$
c_5	$\frac{N c c_l \rho R}{4}$
c_6	$\frac{k_3 N c c_l \rho R^3}{6} \approx 0$

For a typical helicopter, The three power components account for more than 95% of the total power consumed [57]. For a multirotor, we expect a similar rotor power contribution. The remaining would be consumed by the autopilot electronics. The rotor interference effect is not considered because there is no propeller disk overlapping for multi-rotor UAS [57]. The question in doubt is how much each individual power component contributes to the total power, which we address in Section 3.3.

3.3 Model Identification by Experiments

Figure 3.6: The test sites at the Richmond Field Station. Experiment 1 and 2 were performed at the red area, while experiment 3 was at the yellow area.

To identify the unknown coefficients, we perform three simple experiments, namely hover, steady-state ascend/descend, and cyclic straight-line missions, on an IRIS+ from 3D Robotics [1]. From equation (3.8) and (3.9), the power components are super-linear functions of payload and airspeed. Thus, we will focus on these two important factors. To minimize the effect of wind, the experiments were performed in a field partially surrounded by plants at the Richmond Field Station [104]. Figure 3.6 shows the site configuration and highlights the two test areas.

In each experiment, the total power draw is computed by multiplying voltage and current measurements from the onboard power module. The ground speed and Euler angles from GPS and IMU are computed by an extended Kalman filter (EKF) in the autopilot. The angle-of-attack α is computed from roll (ϕ) and pitch (θ) angles from equation (3.10).

$$\alpha = \cos^{-1}(\cos\phi\cos\theta) \tag{3.10}$$

The airspeed is directly measured from a pitot tube with a 4525DO digital differential pressure sensor [133]. To ensure the airspeed measurement is taken horizontally in our testings, the pitot tube is mounted on a 3DR0654 camera gimbal (Figure 3.7). The self weight of the IRIS+ with the gimbal and pitot tube assembly is $mg = 15.5N$. All data is collected at $10Hz$. For consistency, each experiment is repeated three times.

Figure 3.7: An IRIS+ with the airspeed sensor attached to a camera gimbal.

The data collected from the experiments and the scripts written to perform the system identification and validation are hosted in BitBucket repository [132]. Interested readers are encouraged to download the data for other research. In the following, we discuss the three experiments in details.

3.3.1 Experiment 1: Hover

In this experiment, the IRIS+ UAS is loaded with different payloads to hover (Figure 3.8). A total of 13 data points were collected from the vehicle. The total self weight with payload ranges from $14.3N$ to $24N$, close to the handling limit of the vehicle. The total power is given by equation (3.11). Figure 3.9 shows a least-square (LS) fit, giving $c_1 + c_2 = 2.84(m/kg)^{1/2}$.

$$P_{exp1} = P_{i,hover}(mg) + P_p(mg, 0) = (c_1 + c_2)(mg)^{3/2} \tag{3.11}$$

Figure 3.8: IRIS+ hovering with different payloads.

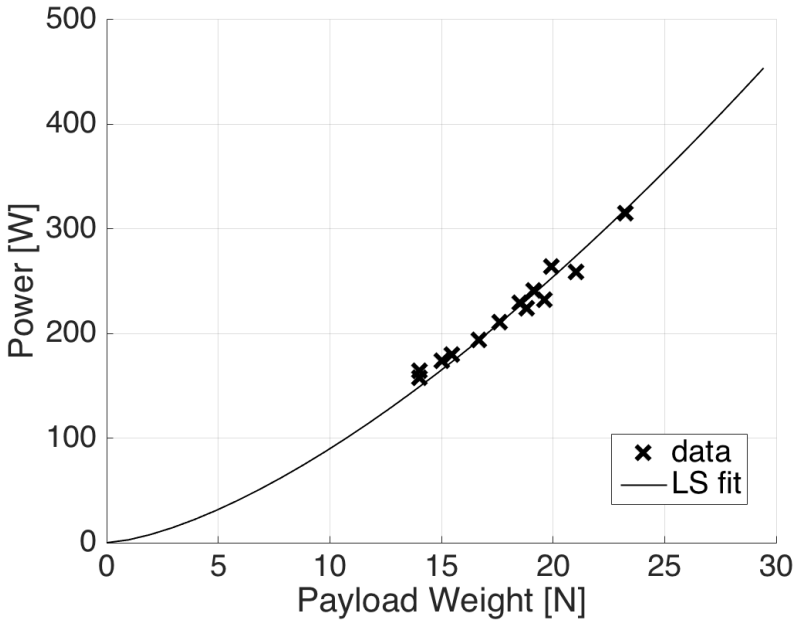


Figure 3.9: Least square fit of IRIS+ hovering with different payloads.

3.3.2 Experiment 2: Steady-state ascend/descend

In this experiment, the IRIS+ UAS is commanded to ascend and descend at constant vertical speed of $V_z = 2.5m/s$ between $0m$ and $100m$ altitude without payloads (Figure 3.10). This is the default maximum ascend and descend rate set by the ground station. Further variations on the vertical speed is unnecessary because the difference in profile power is too small to observe, and we already have enough data to uniquely solve the equations. The total power is given by equation (3.12). The measurement is shown in Table 3.2. The reported values are averages of the time-series data at steady state. The standard deviation (stdev) is also shown for comparison purpose. Together with Experiment 1, we have four equations (hover, ascend, descend, and $c_1 = k_1/k_2$), and four unknown parameters ($k_1, k_2, c_1,$ and c_2). The identified parameters are shown in Table 3.3. Note that the value of $c_1 + c_2 = 3.11$. It is slightly larger than the one reported from Experiment 1 but still acceptable ($\sim 10\%$ error), given the limitations on measurement accuracy and possibly other unknown disturbances.

$$P_{exp2} = P_i(mg, V_z) + P_p(mg, 0) \tag{3.12}$$

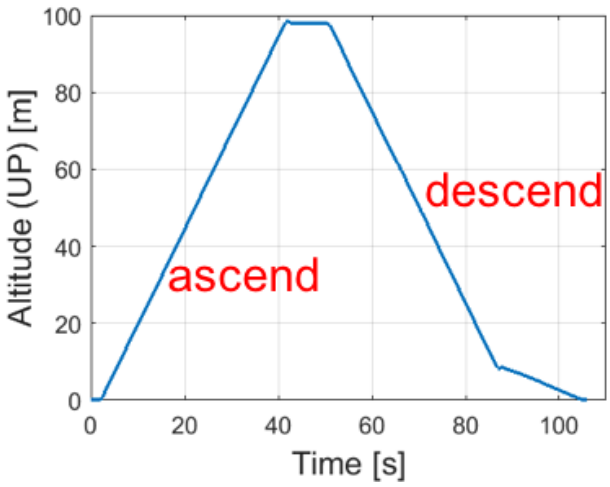


Figure 3.10: Steady-state ascend and descend altitude trajectory.

Table 3.2: Data from Experiment 2

scenario	vertical speed [m/s]	power (avg.) [W]	power (stdev) [W]
ascend	2.5	180	6.3
hover	0	164	1.7
descend	-2.5	150	4.9

3.3.3 Experiment 3: Cyclic straight line

The goal of this experiment is to quantify the effect of parasite drag. In this experiment, the vehicle self weight with gimbal and pitot tube is $mg = 15.5N$. The total power is given by equation (3.13), with the thrust T, lift L, and drag D defined by equation (3.7). The parameter c_3 is assumed to be 0 to simplify the identification process (Section 3.2.2).

$$\begin{aligned} P_{exp3} &= P_i(T,0) + P_p(T,V_{air}) + P_{par}(V_{air}) \\ &= (c_1+c_2)T^{3/2} + c_3(V_{air}cos\alpha)^2T^{1/2} + c_4V_{air}^3 \\ &\approx (c_1+c_2)T^{3/2} + c_4V_{air}^3 \end{aligned} \tag{3.13}$$

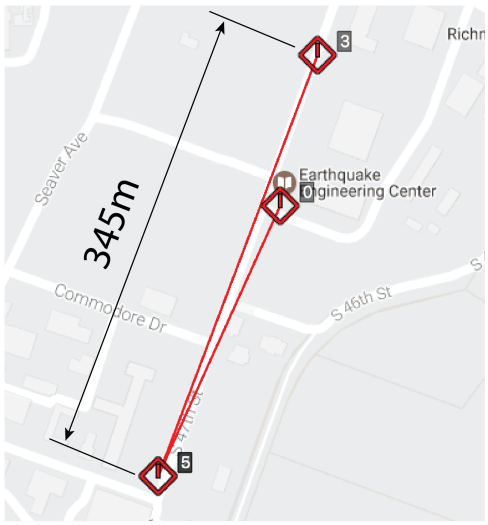


Figure 3.11: The cyclic straight line mission.

The mission was setup in Mission Planner [80] and performed by the autopilot autonomously. The IRIS+ was commanded to follow a 345-meter-long straight-line path back and forth between two waypoints for 3 to 5 times (Figure 3.11), with a 20-second hovering duration after reaching the waypoints. In each test, the UAS was commanded to follow a different desired horizontal ground speed V_{des}, ranging from $1m/s$ to $25m/s$ with $1m/s$ increments.

The test site is the highlighted road in yellow shown in Figure 3.6. During the tests, the wind was found blowing almost completely along the road. Therefore, we can safely ignore the power incurred due to cross wind (perpendicular to the flight directions). This treatment is necessary because pitot tubes can only measure airspeed in the longitudinal direction. The weak cross wind component makes the collected flight data suitable for system identification purpose. When using the model in flight planning, we use wind field data, so both the longitudinal and lateral components of wind are considered, and the power estimate is valid.

Figure 3.12 shows a high-speed test at $V_{ground} = 20m/s$. One forward and backward flight cycle is highlighted in red to illustrate the effect of wind. We focus our attention on the first half of the highlighted cycle, from $30s$ to $50s$. Note that although the IRIS+ was commanded to fly at $20m/s$, the actual ground speed could only reaches up to $10m/s$ in this half cycle, while the airspeed in the other half cycle reached $20m/s$. Therefore, the IRIS+ was at first flying against, and later along, very strong wind. For safety reasons, the autopilot limited the maximum tilting angle, α in our case, to be $45°$, which implicitly set a threshold on the maximum obtainable airspeed. The strong parasite drag exerted by wind cost the IRIS+ an additional $100W$ to overcome compared to hovering.

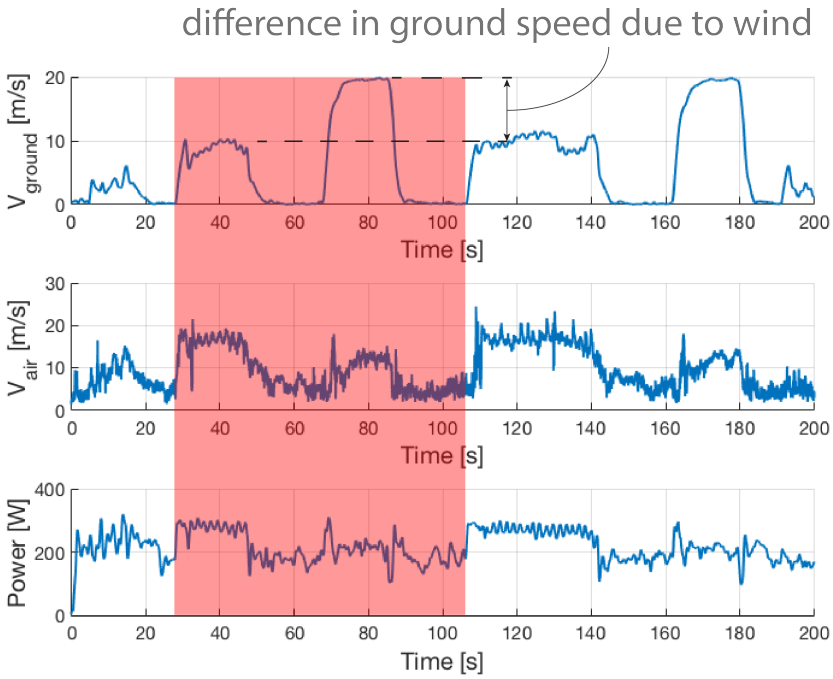


Figure 3.12: Time history of ground speed, airspeed, and power fora $20m/s$ high speed test.

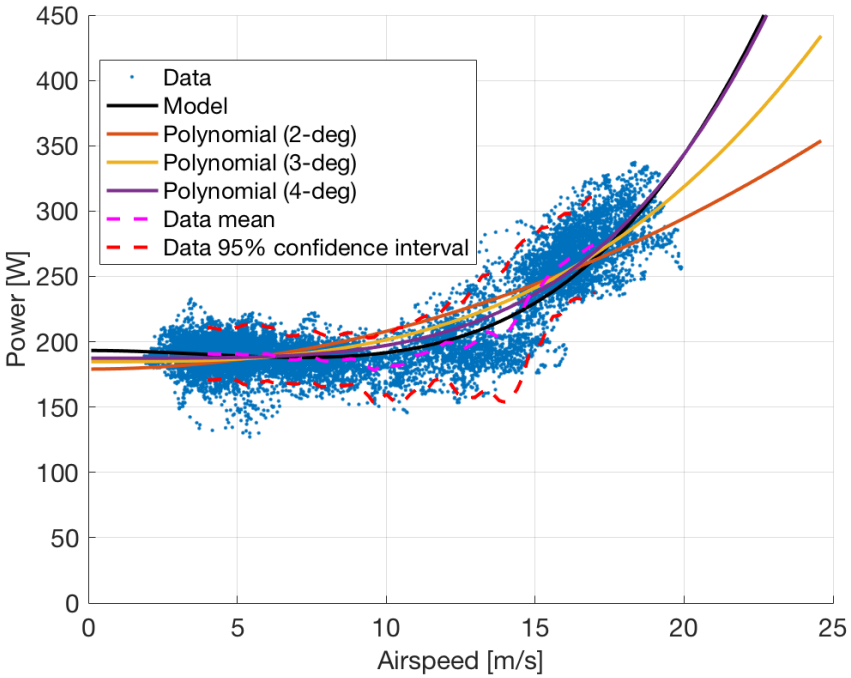


Figure 3.13: Power vs. airspeed data and LS fittings.

The test data and model identification results are summarized in Figure 3.13. The horizontal axis is the measured airspeed along the flight direction. The power consumption data was sampled at hovering and steady-state traveling. The mean and 95% confidence intervals of the sampled data points are shown in dashed lines. A least square fit is performed to the data using equation (3.13), which yields the value of c_4 and c_5 in Table 3.3. Note that the power consumption stays essentially constant for airspeed less than $10m/s$. The standard deviation of the data at different airspeed ranges from 10 to $20W$.

Table 3.3: Identified Parameters

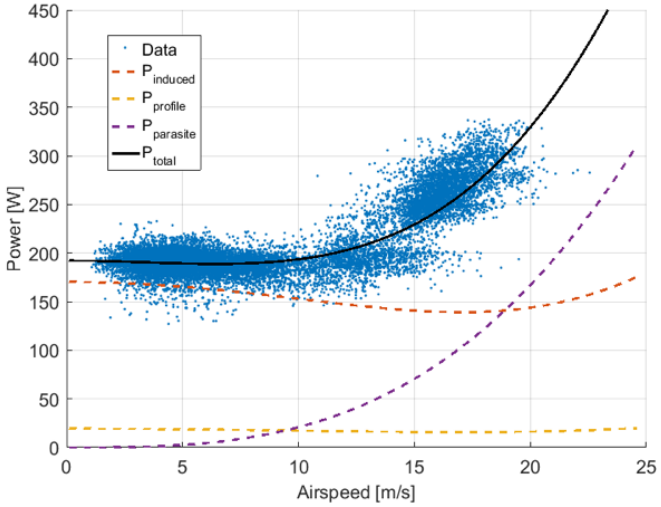
parameter	value
k_1	0.8554
k_2	$0.3051(kg/m)^{1/2}$
$c_1 = k_1/k_2$	$2.8037(m/kg)^{1/2}$
c_2	$0.3177(m/kg)^{1/2}$
c_3	~ 0
c_4	$0.0229kg/m$
c_5	$0.0154kg/s$
c_6	~ 0

When the airspeed is between 8 and $12m/s$, we observed slight power reduction due to increase in flow efficiency (Figure 3.5). We modeled this phenomenon by introducing a pseudo "translational lift" term L. With the identified parameter c_5, L is $2.5N$ at $10m/s$ airspeed.

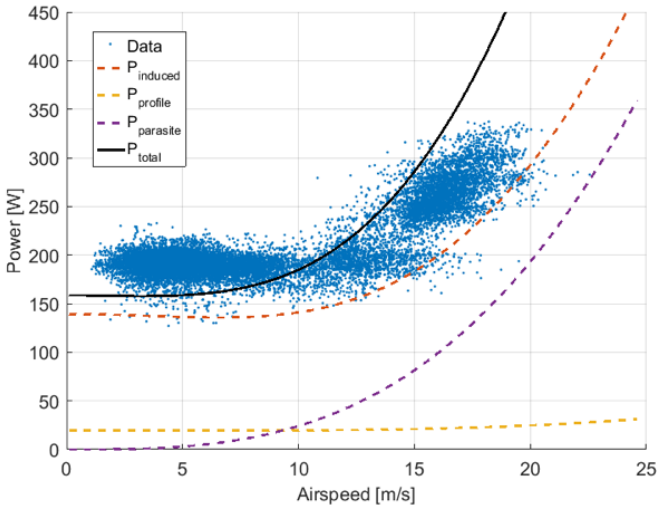
Our model has several advantages over the traditional polynomial fittings shown in Figure 3.13. First, our model is more accurate in the current scenario. Observe that polynomial fittings of degree 2 or 3 produce inadequate predictions at hovering and high speed. A 4-degree polynomial gives a much better fit, but the region with higher motor efficiency ($V_{air} > 8m/s$) is still deviated more from data mean compared to our model. Second, our model has less parameters. In this experiment, our model only needs 3 coefficients, including $(c_1 + c_2)$, c_4, and c_5, compared to 5 parameters for a 4-degree polynomial of similar accuracy. Lastly, our model is able to extrapolate. This point is elaborated in Section 3.4.2.

The major difference between our model and the helicopter literature is that we model the "translational lift" differently (Section 3.2.1). In the helicopter literature, the Glauerts induced velocity model is commonly used. However, it is only valid for a single propeller helicopter. The best model identification using this model is presented in Figure 3.14b. Observe that the total power was underestimated in low-speed regions and overestimated in high-speed regions. The reason is that the induced power component in the dashed red line is overestimated in the high speed region. Figure 3.14a shows the component break down

in our model. Note that the induced power was able to decrease to the proper value. The reason is that we model the lift drop explicitly as a pseudo-lift term. The parasite power and the profile power are very similar in both models.



(a) The proposed model.



(b) Glauerts induced velocity model.

Figure 3.14: Model accuracy comparison and power component breakdown.

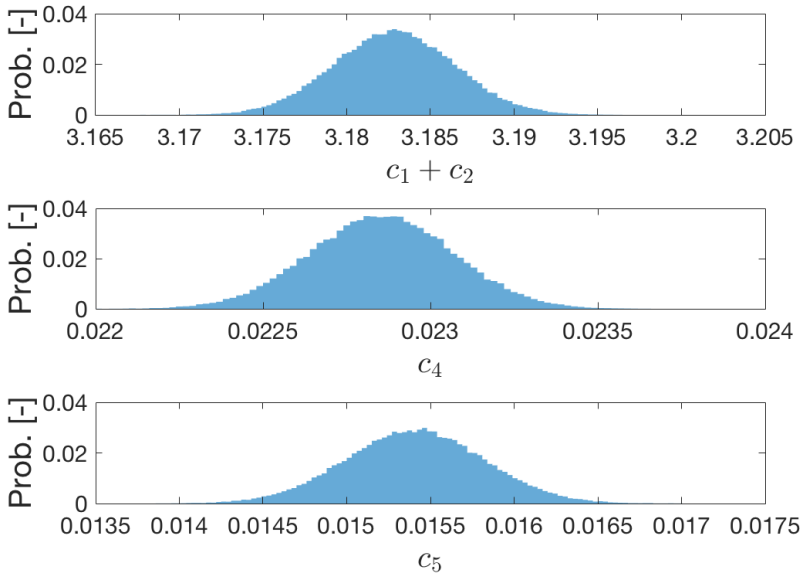


Figure 3.15: Histograms of the identified parameters in Experiment 3.

To verify the model robustness, we also performed bootstrapping on the identified parameters. Specifically, we repeatedly re-sampled the input and output data, including airspeed, angle-of-attack, and power, with replacement, and re-optimized the least-square errors. This re-sampling and re-optimization process is performed $100,000$ times, to generate probabilistic distributions of the identified parameters $(c_1 + c_2)$, c_4, and c_5 (Figure 3.15). In all three cases, the parameters looks Gaussian with a sharp mean and small variance, indicating that the parameters are robust to data re-sampling.

3.4 Model Validation

In this section, two experiments are performed to validate the model. First, we test the model accuracy against different airspeed in a general flight. Second, we test the model's extrapolation capability when flying at different payload.

3.4.1 Airspeed Validation

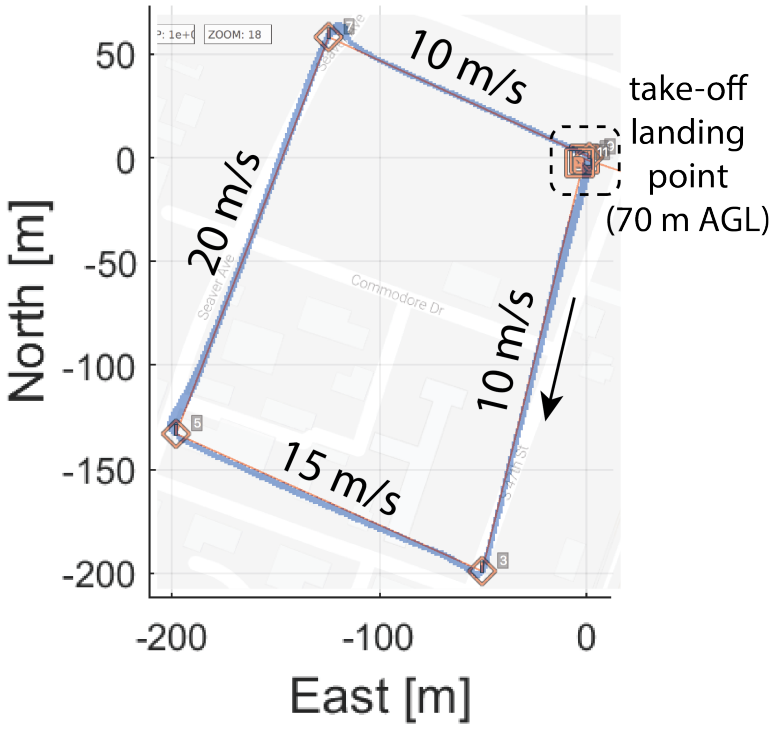


Figure 3.16: The validation experiment is a rectangular loop at 70m AGL. The waypoints are shown as red squares, and the recorded flight path in blue is overlaid with the desired path in black.

The proposed power consumption model at steady state was validated by a separate experiment (Figure 3.16). The UAS was first commanded to ascend to 70m above ground level (AGL), then complete a few rectangular loop at constant altitude, and finally land. At each edge of the loop, the UAS was commanded to travel at three different ground speeds, namely 10m/s, 15m/s and 20m/s. The altitude is significant higher than Experiment 3, but it is necessary to ensure obstacle-free flights. We only present validations for the horizontal loops, neglecting the take-off and landing phase because they were not completely autonomous.

The validation results from a 3-loop test are presented in Table 3.4. The first column gives the loop number in this flight. The second column lists the energy computed from voltage and current data from the power module. The third column shows the energy computed from the proposed power consumption model with measured airspeed and angle-of-attach as inputs. We do not use the desired trajectory for energy computation because the actual speed deviates from the desired speed quite a bit. One reason is that the wind speed at this altitude is much larger than close to the ground. The last column shows the percent differences between measurements and model simulations. Thanks to the direct airspeed measurements, percent errors in all three loops are within 5%.

A time history comparison between the data and model is presented in Figure 3.17. Two loops are highlighted in different colors. A major difference between the data and model is that the data contains a lot of spikes. This is because our model only captures the steady-state behavior, while the IRIS+ actually consumes more or less energy when accelerating or decelerating. But on average, the energy differences cancel out, resulting in the small percent error in Table 3.4. Other errors could be caused by unmeasured cross-wind gusts.

Table 3.4: Validation Results

Loop #	Energy from Measurement [kJ]	Energy from Model [kJ]	% Difference
1	14.6	15.2	4.5%
2	15.2	15.5	1.6%
3	15.0	15.6	4.0%

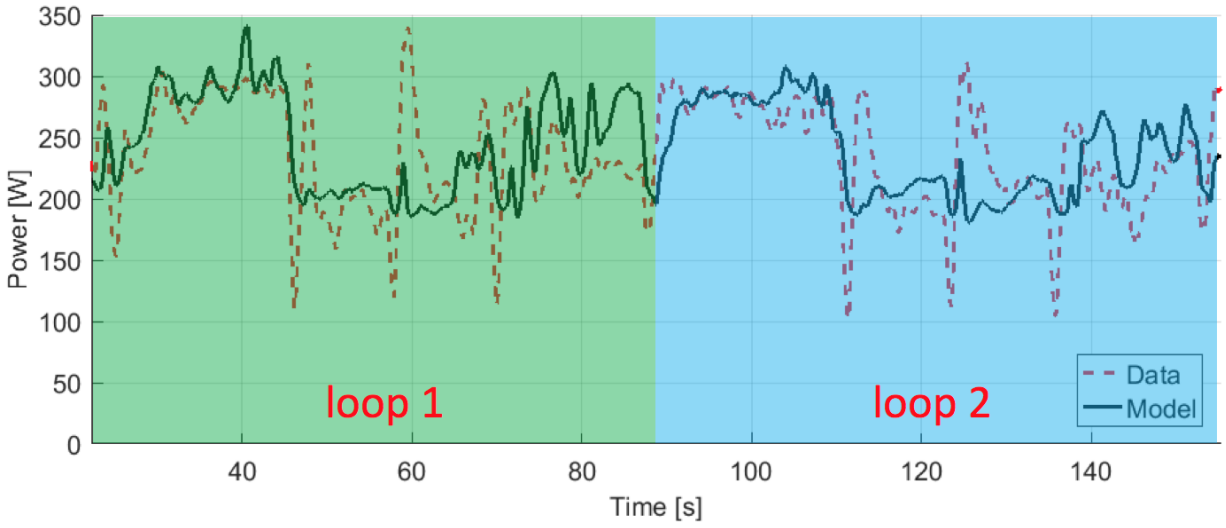


Figure 3.17: The validation experiment is a rectangular loop at $70m$ AGL. The waypoints are shown as red squares, and the actual path in blue is overlaid with the desired path in black.

3.4.2 Payload Validation

Our model also considered payload, or vehicle weight, as a model input. Thus, we performed Experiment 3 again for a heavier payload ($mg = 18.5N$), to observe the extrapolating

capability of our model. The power from the collected data and the model are shown in Figure 3.18 at various airspeed. Observe that at low airspeed ($\leq 10m/s$), the model over predicts the power consumption, but at higher airspeed ($> 10m/s$), the model matches the data mean really well.

The payload validation shows an additional advantage of our model over polynomial fittings. Our model is capable of predicting power at different payloads, a benefit inherited from first-principle modeling. However, similar capability in polynomial fittings can only be achieved by additional experiments, to generate a set of curves for different vehicle weights, and interpolate the results in between.

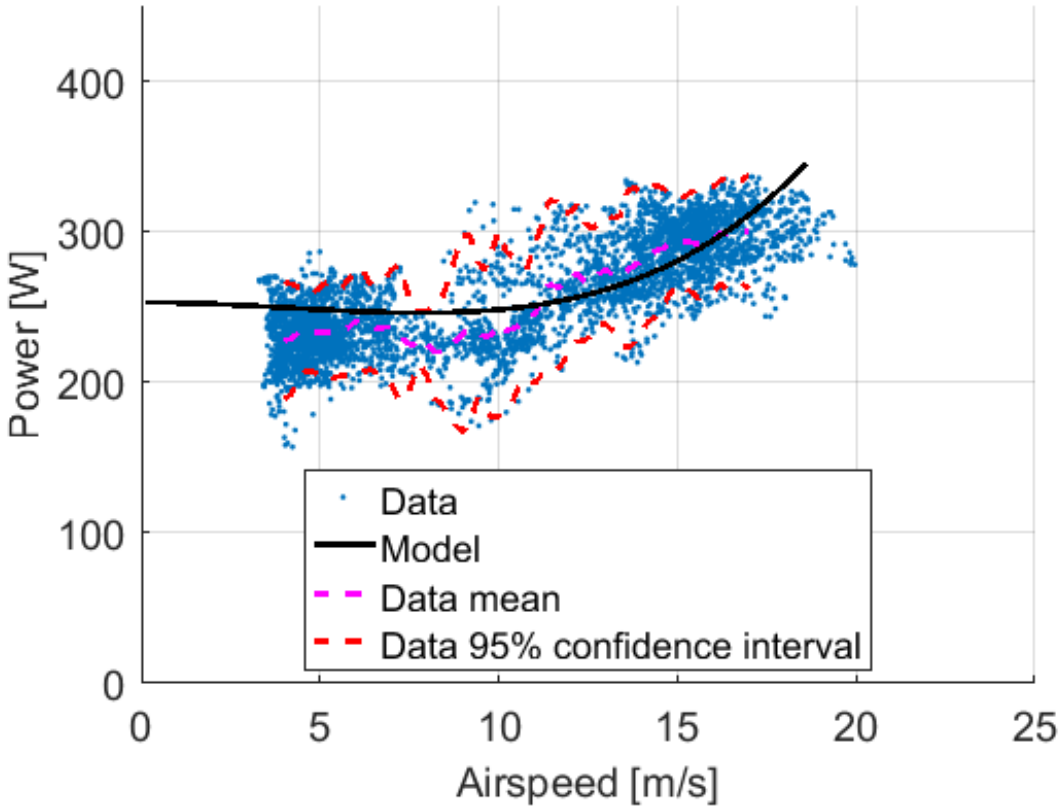


Figure 3.18: Results comparing the data and model prediction for a different payload in a range of airspeed.

3.5 Model Simulation

In this section, we illustrates three factors, namely vehicle speed, altitude, and regional variations, affecting the energy consumption in a straight line scenario (Figure 3.19). The UAS flies a one-way or round trip between two points A and B with a fixed distance d, with

constant wind speed V_{wind}. For simplicity, we only focus on the cruising stage of the trips with the following parameters.

$$m = 2kg \quad d = 10km$$

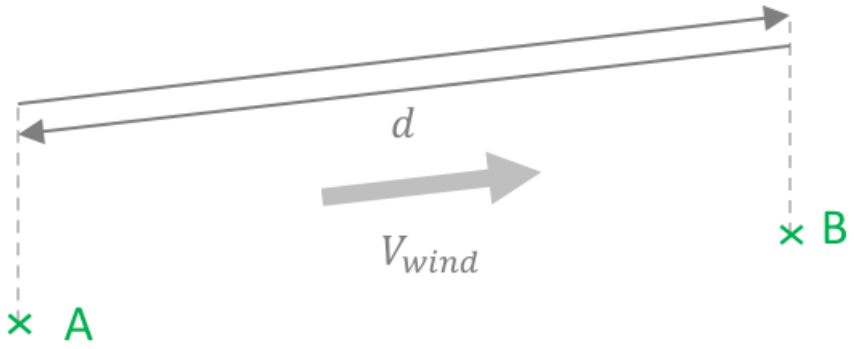


Figure 3.19: Setup of the simple straight line scenario.

3.5.1 Optimal Speed

In commercial aircraft, it is well-known that an airplane optimizes fly speed to minimize fuel burn. For example, the fuel burn per mile is minimized when the ratio $C_L^{1/2}/C_D$ is maximized, where C_L and C_D are the lift and drag coefficients, respectively. The first scenario explores the optimal speed of a multi-rotor UAS. In this case, the UAS is flying without wind ($V_{wind} = 0$). Therefore, ground speed is the same as airspeed (equation 3.14). For simplicity, we will only use ground speed V_{ground}, i.e.,

$$V_{ground} = V_{air} \tag{3.14}$$

The simulation result for a one-way trip without wind is summarized in Figure 3.20. The plot indicates that there is an optimal ground speed, $V^*_{ground} = 17.2m/s$, minimizing the total energy consumption at the cruise phase. When traveling below V^*_{ground}, the UAS spends more energy on hovering. In the extreme case when $V_{ground} = 0$, the vehicle consumes an infinite amount of energy. When traveling above V^*_{ground}, the UAS spends more energy on overcoming the parasite drag. In the extreme case when $V_{ground} = \infty$, it takes an infinite amount of energy to the destination. We use this insight to develop the speed optimization procedure in Chapter 2.

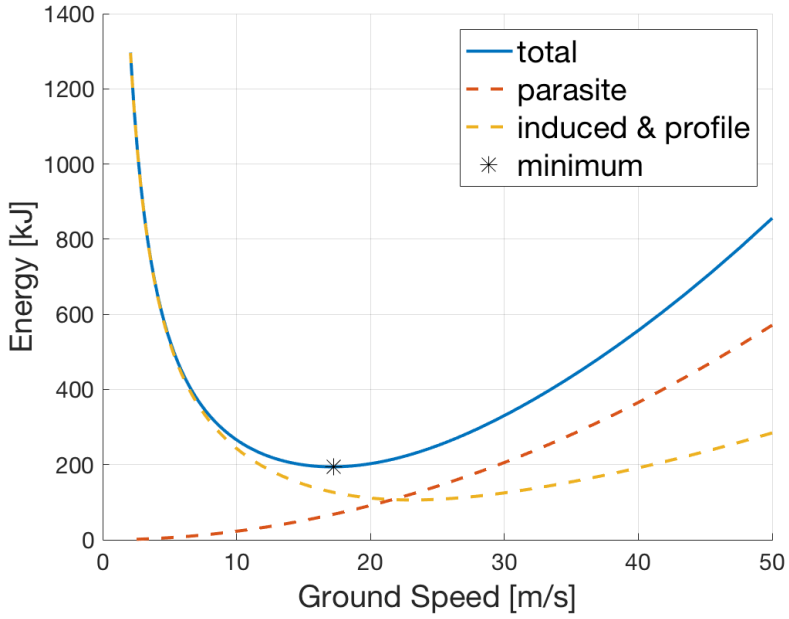


Figure 3.20: Energy vs. ground speed plot for a one-way trip without wind.

3.5.2 Altitude Variation

In commercial aircraft, engine efficiency becomes higher as altitude increases, because air density is lower. In the second scenario, we would like to investigate the energy variation due to altitude. Unlike commercial aircraft, UASs fly at low altitude, and air density does not change much with respect to altitude. But wind speed vary dramatically in this altitude range. Therefore, for low-altitude UAS flights, the determining factor in altitude variation is the vertical wind speed profile.

Figure 3.21 shows two empirical vertical wind profile fittings. The data is the averages measured from the Walsenburg South site by the Colorado Anemometer Loan Program at the Colorado State University from 2010 to 2012 [56]. One fit uses a log-law profile, and the other uses a power-law profile [89]. The deviation between the two curves is small. We simply use the average of them to perform the energy computation.

To assess the effect of vertical wind profile, we computed the round-trip energy consumption when the UAS is flying along or against wind at two representative altitudes, with the ground speed defined in equation (3.15). It is chosen such that the UAS is flying at airspeed $V_{air} \approx 13m/s$ and thus the power is similar in both cases.

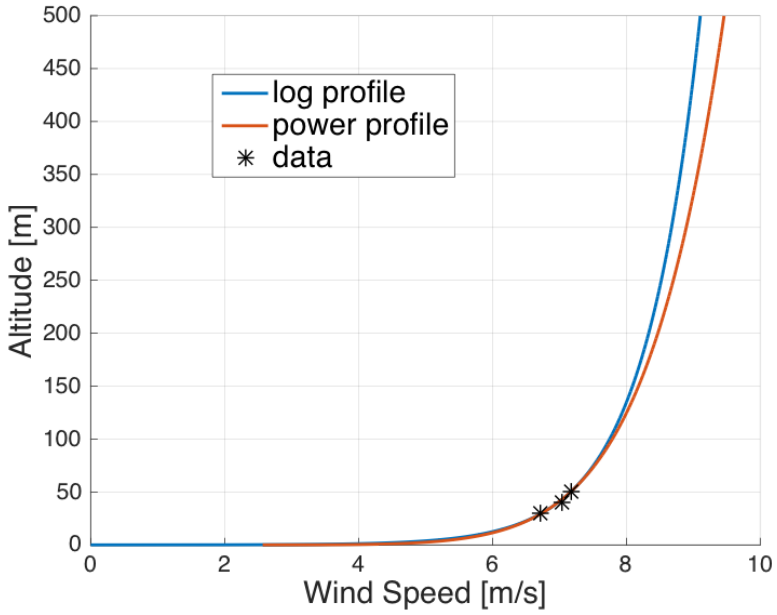


Figure 3.21: Empirical vertical wind speed profile from data.

$$V_{ground} = \begin{cases} 21m/s & \text{when along tailwind} \\ 5m/s & \text{when against headwind} \end{cases} \tag{3.15}$$

Table 3.5: Energy consumption of a straight-line round trip at different altitudes with the vertical wind profile defined in Figure 3.21.

Altitude [m]	Along wind [kJ]	Against wind [kJ]
200	124	538
100	127	526

The data reflects that the UAS consumes less energy when against headwind at lower altitude ($100m$), and when along tailwind at higher altitude ($200m$). Therefore, it is advantageous to fly low when against headwind and fly high when along tailwind. We use this observation in Chapter 2 to select the optimal flight altitude.

3.5.3 Weather and Regional Variation

The temperature and humidity difference due to weather or regional variation can affect airplane flights significantly. In most helicopters flying at low altitude, the difference in

performance on a cold, dry winter day is very noticeable and much better than on a humid, hot summer day. However, as the airspeed increases, aerodynamic drag becomes significant, and fly high is preferred to lower the air density and thus drag. Air density plays the key role in this trade-off.

In this scenario, we explore the energy variation due to regional difference. Specifically, the change of air density due to temperature and humidity. The parameters in equation 3.5 are functions of air density. The air density is low when it is hot and humid, and high when it is cold and dry.

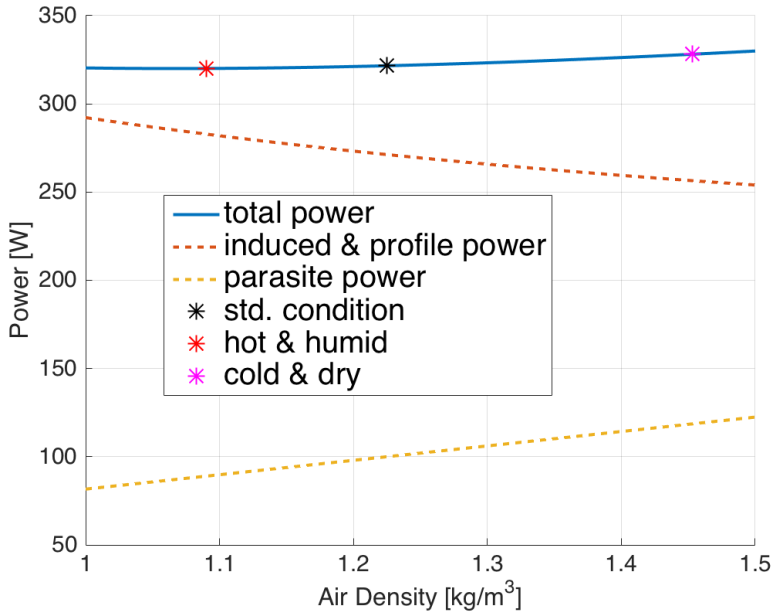


Figure 3.22: Power consumption curve of an IRIS+ at $V_{air} = 5m/s$ as a function of air density, or temperature and humidity.

At an airspeed $V_{air} = 5m/s$, we vary the air density to include a range of extreme weather conditions, and simulate the power consumption plot in Figure 3.22. The total power stays almost constant because the induced and profile decreases as air density increases, while the opposite relation holds in parasite power. Intuitively, as the air gets thinner, the propellers consume more energy to maintain hovering, but the aerodynamic drag also become smaller. The net effect is that they are canceling each other out, resulting in a constant power curve. At higher airspeed, e.g. faster than $15m/s$, drag will dominate, and thus hot and humid weather is preferred. Otherwise, at lower airspeeds or hovering, cold and dry weather is more desirable. Further experiments is necessary to validate this result. This is left as future work.

Chapter 4

Optimal Trajectory Generation

4.1 Introduction

In this chapter, we formulate the optimal control problems to generate 4D trajectories with either minimum energy or time under the influence of wind and static obstacles. To motivate the work, we first review the navigation system and algorithms in current UAS autopilots and the robotics literature in Section 4.1.1 and 4.1.2. We summarize our contribution at the end of the section. Then, we formulate the general optimal control problem in Section 4.2. The problem is simplified to a minimum energy problem and a minimum time problem with fixed ground speed or fixed airspeed in Section 4.3. Next, in Section 4.5, we discuss the environmental data necessary for the optimal routing computation. Finally, in Section 4.6,we simulate several scenarios to illustrate the obstacle avoidance capability as well as the energy/time optimality under wind.

4.1.1 Current Path Generation Process in UAS Autopilots

Similar to aviation, we need to generate 4D flight trajectories, including 3D waypoint position and the corresponding timestamps, to estimate the energy demand. Currently, the navigation system in commercial or open source UAS autopilot defines a sequence of waypoints for the UAS to follow. Figure 4.1 shows two ways to define waypoints in Mission Planner for ArduPilot [80]. The pilot can either define sequential waypoints on a map manually (Figure 4.1a) or draw a polygon to let the ground station compute a sweep pattern (Figure 4.1b). During the flight, the ground station smooths the waypoint sequences by fitting a spline. The UAS then flies the waypoints smoothly (Figure 4.2).

However, this mode of navigation is not desired in many challenging flights. First, this approach provides no obstacle avoidance guarantee when the UAS is flying at low altitude. It imposes a high level of autonomy requirement on the UAS to avoid obstacles such as buildings and towers in real-time. With the size and payload capacity of many UASs, this expectation is unrealistic. In addition, the flight altitude is usually referenced from a home waypoint, instead of altitude above ground level (AGL). This is because the map is usually

(a) Waypoint sequence. (b) Sweep pattern.

Figure 4.1: Path definitions in Mission Planner, a ground station software for ArduPilot.

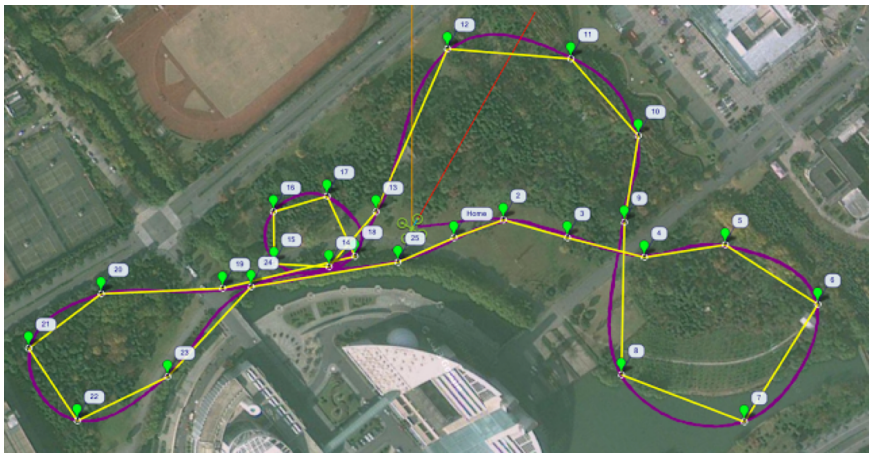


Figure 4.2: Waypoint path and actual flight path smoothed by spline.

2-dimensional and does not contain terrain altitude information. If there were a cliff in front, the UAS may crash. Our trajectory generation approach addresses this problem by computing routes on an elevated virtual terrain surface post-processed from digital elevation models (DEMs). The paths are guaranteed to consider terrain variation and other mapped static obstacles.

Second, straight line routes in general do not optimize energy consumption or travel duration under the influence of wind, which in certain cases could be very significant, as we shall see in Section 4.6.2. For example, if the UAS flies in a straight line oriented completely against very strong wind, its power draw can increase dramatically, and the flight duration can be reduced significantly. In addition, flying against high wind can reduce the ground speed significantly, and thus increases the flight duration. Our routing algorithm (Section 4.3.2) addresses this problem by considering the energy or time cost as a function of the navigation direction. The paths are guaranteed to consume the least amount of energy or

flight time.

4.1.2 Path Planning in Robotics

In robotics, there is a rich literature in path and trajectory generation. Many research efforts on UAS path planning are summarized in [38, 25]. In [78, 114, 134], the trajectory generation problem was formulated as Mixed-Integer programs. The quadrotor UAS is modeled as a rectangular prism, and it navigates through an environment with convex obstacles. The condition that the prism does not collide with an obstacle at a certain time is enforced by binary slack variables. Although the computation time to generate the minimum-time trajectory is long, the first feasible solution can be found very fast.

There are at least two reason why this approach is not a good idea for UAS navigation. First, the number of binary variables increases exponentially as the number of obstacles rises. Terrain can have very complex geometry, and if we model terrain as a fine mesh, the number of binary variables will explode, and the integer program becomes intractable. Second, there is no mechanism on handling wind. Under strong wind, the feasibility of these trajectories is questionable. In our approach, the routing algorithm is very efficient, with a computational complexity of $O(N \log(N))$ where N is the number of vertices in the terrain model. More importantly, it models the effect of wind in the cost function (Section 4.3.2). Strong wind above a certain threshold is considered infinitely costly and are avoided automatically (Section 4.3.5).

A second approach is potential field [51], which is computationally fast but non-optimal and incomplete, meaning that there is no guarantee to reach the destination. Therefore, this approach is not feasible in our application. Another type of approach is heuristic-based, such as A* and D* [34]. This type of algorithms is fast because they keep track of a heuristic cost guiding to the destination. However, the cost does not take into account fly directions and thus cannot account for wind.

Lastly, there are wave-front-propagation-based approaches. One example is the classical Dijkstra's algorithm. It produces shortest paths in graphs. However, the network solution from the algorithm cannot distinguish between the various Manhattan paths of equal costs. This inconsistency issue with the underlying continuous problem is explained in detail in [115]. The Fast Marching method (FMM) is introduced in [115] to address this problem. It directly approximates the solution of the underlying partial differential equation through consistent numerical approximations. However, FMM can only handle isotropic problems in which the cost is independent of propagation direction. In the presence of wind, the energy or time cost can become anisotropic, or they changes as a function of the navigation direction. The anisotropic property renders the FMM inappropriate for our problem.

To handle anisotropic costs, another numerical algorithm called the ordered upwind method (OUM) is developed [116]. Unlike the numerical schemes in Dijkstra's algorithm or the FMM, the OUM formulates a new numerical scheme which iteratively search for the optimal cost in different directions. But as sacrifice, the time it takes to update a single vertex is much longer. Nevertheless, the overall computational complexity is still $N \log(N)$.

In [31], the author presents an application of the OUM on guiding UAS through tornadoes. The cost in this work is travel time in wind. Precipitation regions are modeled as obstacles with infinite cost. However, they only apply OUM in an idealized flat surface without terrain information, and the cost considered is limited to time.

Our main contribution in this work is to formulate new energy and time costs relevant to UAS missions as functions of wind. The new costs are mostly anisotropic and have to be solved by the OUM. The major innovation is on the cost profile design, which we explain in Section 4.3.2.

4.2 The General Energy Optimal Control Problem

To compute the minimum cost trajectory in a trip, we formulate a fixed end-point free-end-time optimal control problem in equation (4.1).

The cost function J in equation (4.1a) is a time integral of infinitesimal costs $c(t)$ at the cruising stage. The decision variable is the time sequence of air velocity $\boldsymbol{V}_{air}(t)$. The problem includes several constraints. First, equation (4.1b) defines the dynamic constraint. It says that the ground velocity $\boldsymbol{V}_{ground}$, is the sum of the air velocity $\boldsymbol{V}_{air}$ and wind velocity $\boldsymbol{V}_{wind}$. The state $\boldsymbol{x}$ is the 3D position of the UAS. Equation (4.1c) defines the feasible region Ω to avoid static obstacles. We define it by digital elevation models (DEMs) in Section 4.5.1. In addition, we have an initial condition $\boldsymbol{x}_0$ (equations (4.1d)) and a terminal condition $\boldsymbol{x}_f$ (equation (4.1e)), representing the origin and destination, respectively. Lastly, there is a limit on maximum airspeed V_{max} set by vehicle performance (equation (4.1f)), and possibly a constraint on ground speed $V_{g,max}$ (equation (4.1g)) set by regulation agencies such as the Federal Aviation Administration (FAA).

$$\min_{\boldsymbol{V}_{air}} \; J = \left[\int_0^{T(\boldsymbol{x}_0, \boldsymbol{V}_{air})} c(t)dt \right] \tag{4.1a}$$

$$s.t: \quad \dot{\boldsymbol{x}}(t) = \boldsymbol{V}_{ground} = \boldsymbol{V}_{air} + \boldsymbol{V}_{wind} \tag{4.1b}$$

$$\boldsymbol{x}(t) \in \Omega \tag{4.1c}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \in \Omega \tag{4.1d}$$

$$\boldsymbol{x}(T) = \boldsymbol{x}_f \in \Omega, \qquad T \in (0, \infty) \tag{4.1e}$$

$$||\boldsymbol{V}_{air}|| \in [0, V_{max}] \tag{4.1f}$$

$$||\boldsymbol{V}_{ground}|| \in [0, V_{g,max}] \tag{4.1g}$$

where

- J is the cost to be minimized.
- $c(t)$ is the stage cost per unit time, which is a function of time.
- $\boldsymbol{V}_{ground}, \boldsymbol{V}_{wind}$ are the ground and wind velocity, respectively.

- Ω is the feasible region inside which the sUAS can fly.
- $\boldsymbol{V}_{air}$ is the horizontal air velocity, or our control variable.
- $\boldsymbol{x}_0, \boldsymbol{x}_f$ are the origin and destination of a trip, respectively.
- T is the terminal time, which is not fixed in our formulation.
- $V_{g,max}$ is the maximum ground speed.

This problem is in general difficult to solve because both the flight path and the vehicle speed are decision variables. In Section 4.3, we fix the airspeed or ground speed to simplify the problem, and solve for the optimal path via numerical algorithms. Solution to the general problem (4.1) is left as future work.

4.3 The Simplified Optimal Control Problems

As mentioned in Section 4.2, the optimal control problem (4.1) is hard to solve because we need to optimize both vehicle speed and flight path simultaneously. In this section, we fix the ground speed or airspeed and optimize the flight path. The method developed is used to jointly optimize the two in Chapter 2.

This section is organized as follows. First, we state our assumptions to mathematically formulate the problem. Then, introduce the concept of cost profile. Third, we introduce simplified minimum energy and minimum time cost function in equation (4.1). We show that the formulation reduces to a time-invariant Hamilton-Jacobi-Bellman (HJB) equation. Lastly, we solve the formulated problems with the Ordered Upwind Method (OUM).

4.3.1 Assumptions

The following assumptions are made to develop problem (4.1).

- A complete mission consists of three segments: a vertical take-off segment, a cruise segment at fixed altitude AGL, and a vertical landing segment. In the optimal control problem, we only compute optimal trajectories for the cruise segment inside an elevated 3D surface (Section 4.5.1), which is not necessarily horizontal. The take-off and landing segments are computed separately and not considered in the optimization.
- The vehicle is mostly at steady state, enabling the application of the power consumption model developed in Chapter 3 to formulate the energy cost function. This assumption is reasonable because accelerating maneuvers usually only occupy a small portion of the entire trip.

- The trip duration is short enough for the wind field $\boldsymbol{V}_{wind}(\boldsymbol{x})$ to be time invariant. We also assume it is deterministic. For multi-rotor UAS, this assumption is reasonably valid for flight durations up to 30 minutes.

Lastly, to make the problem solvable, we need certain assumptions on ground speed or airspeed, which we discuss in the following section.

4.3.2 Cost Profile Design

We start by re-writing the cost function as a line integral parametrized by s, so that the cost propagates in space instead of time.

$$J = \int_0^T c(t)dt = \int_{s_0\ C}^{s_f} c(t)\left(\frac{dt}{ds}\right)ds = \int_{s_0\ C}^{s_f} \frac{c(s)}{V_{ground}(s)}ds \triangleq \int_{s_0\ C}^{s_f} g(\boldsymbol{x}(s), \boldsymbol{u}(s))ds \tag{4.2}$$

The cost differential g can be a function of both the state $\boldsymbol{x}(s)$ and control $\boldsymbol{u}(s)$. We define the control variable $\boldsymbol{u}$ to be a unit vector tangential to the curve C at s along the navigation direction. This is also the direction of ground speed $\boldsymbol{V}_{ground}$. The feasible control set $\mathcal{U}$ is

$$\mathcal{U} = \{\boldsymbol{u} \in \mathbb{R}^n \,|\, ||\boldsymbol{u}|| = 1\} \tag{4.3}$$

We define the cost profile set $CP(\boldsymbol{x}, \boldsymbol{u})$ in equation (4.4). It is a set of vectors with magnitudes $g(\boldsymbol{x}, \boldsymbol{u})$ in direction $\boldsymbol{u}$ at position $\boldsymbol{x}$.

$$CP(\boldsymbol{x}, \boldsymbol{u}) = \{t\boldsymbol{u}g(\boldsymbol{x}, \boldsymbol{u}) \,|\, \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{u} \in \mathcal{U}, t \in [0, 1]\} \tag{4.4}$$

Problem (4.1) is a standard optimal control problem. In [117], this problem is reduced to a time invariant Hamilton-Jacobi-Bellman (HJB) equation (4.5), by applying Bellman's optimality principle and dynamic programming. The derivation is presented in Appendix D.

$$\begin{aligned} \min_{\boldsymbol{u}\in\mathcal{U}} [(\nabla_{\boldsymbol{x}} V(\boldsymbol{x}) \cdot \boldsymbol{u}) + g(\boldsymbol{x}, \boldsymbol{u})] = 0, \quad & \boldsymbol{x} \in \Omega \\ V(\boldsymbol{x}_f) = 0, \quad & \boldsymbol{x}_f \in \Omega \end{aligned} \tag{4.5}$$

where the value function $V(\boldsymbol{x})$ at s is

$$V(\boldsymbol{x}) = \min_{\boldsymbol{u}\in\mathcal{U}} \ J(\boldsymbol{x}, \boldsymbol{u}) \tag{4.6}$$

If the cost differential g is not a function of direction $\boldsymbol{u}$, then the cost profile is bounded by a circle centered at $\boldsymbol{x}$ and thus isotropic. This problem can be solved by the Fast Marching Method (FMM) [115]. Otherwise, the cost is anisotropic. If an anisotropic cost profile set (4.4) is convex, then we can solve for a unique viscosity solution [117] using the Ordered Upwind Method (OUM). Therefore, our remaining task is to design the cost profile to be convex. The cost differential for energy and time are denoted as g_energy and g_time, respectively.

4.3.3 Energy Cost Profile

To design the energy cost profile, we first re-write the energy differential

$$dE = Pdt = P\left(\frac{dt}{ds}\right)ds = \frac{P}{V_{ground}}ds \tag{4.7}$$

Therefore, if we set the temporal cost $c(t)$ to be the power consumption $P(t)$ of the UAS, the total cost becomes the energy consumed in the cruise phase of the trip. And the spatial energy cost g_{energy} becomes

$$g_{energy}(\boldsymbol{x}, \boldsymbol{u}) = \frac{P(\boldsymbol{x}, \boldsymbol{u})}{V_{ground}(\boldsymbol{x}, \boldsymbol{u})} \tag{4.8}$$

The numerator term $P(\boldsymbol{x}, \boldsymbol{u})$ (Section 3.2.5) and denominator term $V_{ground}(\boldsymbol{x}, \boldsymbol{u})$ for multi-rotor UASs are in general functions of both position and navigation direction because of wind. To ensure the convexity of $g_{energy}(\bar{\boldsymbol{x}}, \boldsymbol{u})$ at a fixed point $\bar{\boldsymbol{x}}$, we can either make power P or ground speed V_{ground} constant and check the resulting cost profiles.

4.3.3.1 Constant Ground Speed

First, we can make the ground speed independent of navigation direction $\boldsymbol{u}$. Without loss of generality, we assume that wind is blowing along the positive x axis at point $\bar{\boldsymbol{x}}$, so that $\boldsymbol{V}_{wind} = [V_{wind},\ 0]^T$, and the direction $\boldsymbol{u}$ makes an angle θ to wind. Therefore,

$$\boldsymbol{V}_{air}(\theta) = \begin{bmatrix} V_{ground}\cos\theta - V_{wind} \\ V_{ground}\sin\theta \end{bmatrix} \tag{4.9}$$

To ensure that the cost profile set is convex, we need to satisfy inequality (4.10) [36].

$$g_{energy}(\theta)^2 + 2g'_{energy}(\theta)^2 - g_{energy}(\theta)g''_{energy}(\theta) \geq 0 \tag{4.10}$$

Figure 4.3 shows the normalized value of the left-hand side of inequality (4.10) as a function of θ, computed for the power consumption model with parameters in Section 3.2.5 and for a range of ground speeds and wind speeds. Observe that the values are mostly greater than or equal to zeros, except for some values near 0 degree and 360 degrees. Regardless, we can safely call the CP set defined by g_{energy} convex. Figure 4.4a shows a numerical example of the power $P(\boldsymbol{x}, \boldsymbol{u})$ affected by wind at a fixed point $\bar{\boldsymbol{x}}$ in every horizontal direction $\boldsymbol{u}$ in polar coordinates. The resulting shape of CP_{energy} is shown in Figure 4.4b. Both shapes looks convex.

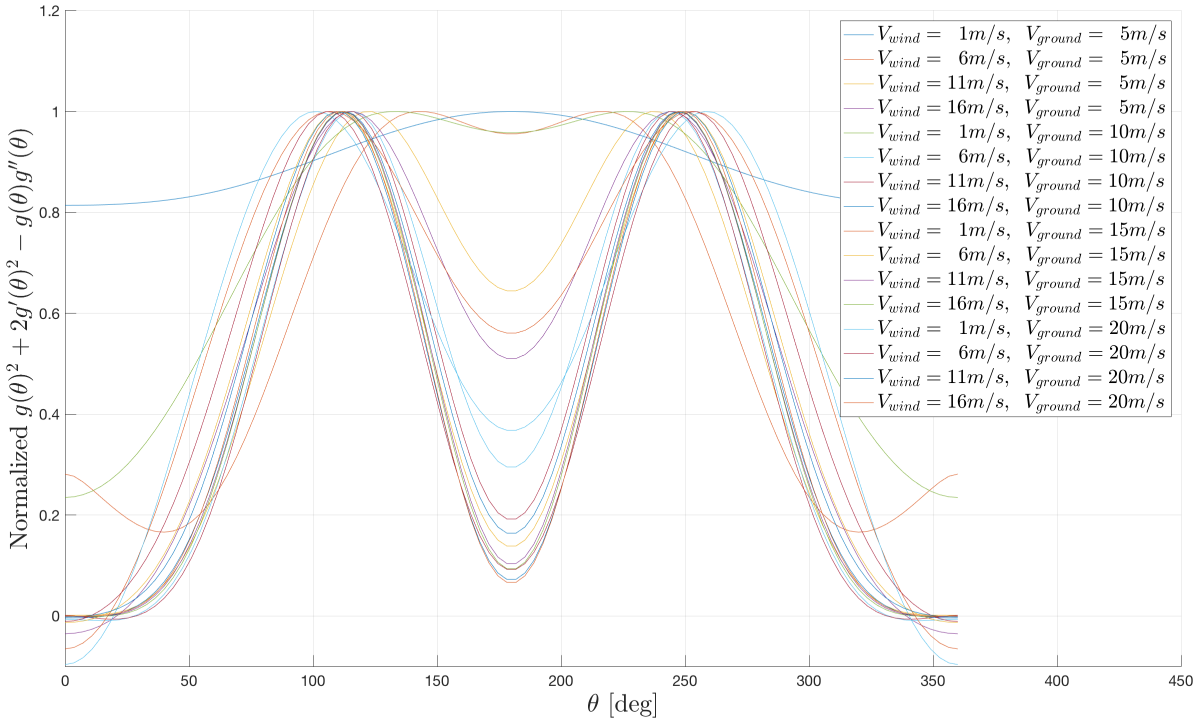
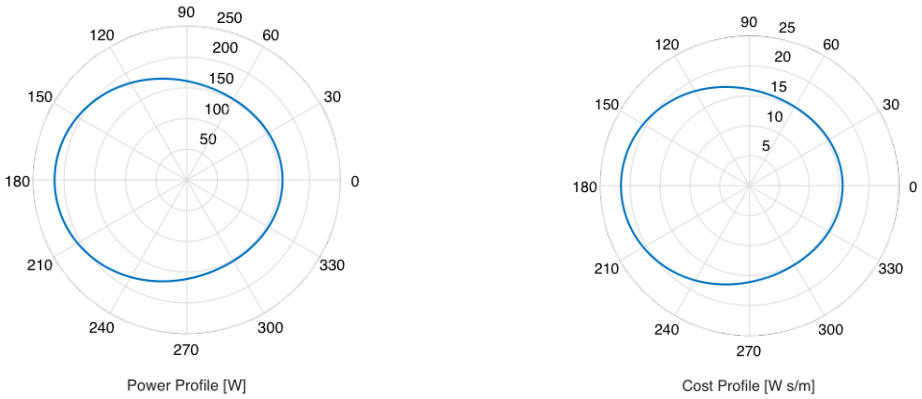


Figure 4.3: Convexity of energy cost profiles for a range of V_{ground} and V_{wind}.



(a) $P(\bar{\boldsymbol{x}}, \boldsymbol{u})$. (b) Energy cost $g_{energy}(\bar{\boldsymbol{x}}, \boldsymbol{u})$.

Figure 4.4: 2D visualization of the energy cost at constant ground speed in polar coordinate at $V_{ground} = 10m/s$ and $V_{wind} = 5m/s$ with wind blowing along the $\theta = 0°$ outward direction.

To reduce confusion and keep things simple, in the simulated examples in Chapter 2 and

Section 4.6, we further restrict the ground speed to be independent of position, so that it is a constant given by equation (4.11) for any flight planning task. This simplification enables us to optimize the ground speed by iteration (Section 2.6.5). Nevertheless, more advanced examples can actually specify a ground speed map as a function of $\boldsymbol{x}$, but designing an optimal ground speed map is a challenging task and out of the scope of this dissertation.

$$V_{ground} = const \tag{4.11}$$

4.3.3.2 Constant Airspeed

Alternatively, we can also make the airspeed independent of direction $\boldsymbol{u}$, so that the power consumption becomes constant at any given point $\bar{\boldsymbol{x}}$. Then, the energy cost element $g_{energy}(\boldsymbol{x}, \boldsymbol{u})$ becomes

$$g_{energy}(\boldsymbol{x}, \boldsymbol{u}) = \frac{P}{V_{ground}(\boldsymbol{x}, \boldsymbol{u})} = \frac{P}{||\boldsymbol{V}_{air}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{V}_{wind}(\boldsymbol{x})||} \tag{4.12}$$

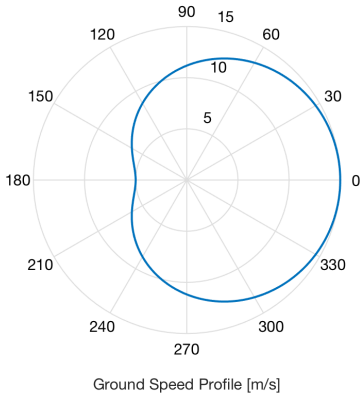
The wind velocity term $\boldsymbol{V}_{wind}(\boldsymbol{x})$ is assumed time invariant (Section 4.3.1). To ensure that the cost profile defined by g_{energy} is convex, we need to again satisfy inequality 4.10. After some simplifications, it becomes

$$|V_{air}^2 + V_{wind}^2 + 2V_{air}V_{wind}\cos\theta| \geq \frac{1}{2} \tag{4.13}$$

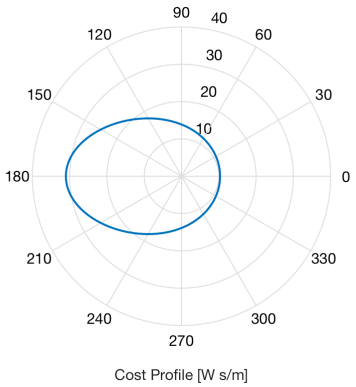
In addition, to make sure that we can fly against wind, we need

$$V_{air} > V_{wind} \tag{4.14}$$

As long as we restrict V_{air} to be slightly larger than V_{wind}, these two condition are satisfied. If we plot the ground speed $V_{ground}(\boldsymbol{x}, \boldsymbol{u})$ at a fixed point $\bar{\boldsymbol{x}}$ in wind, we obtain a polar plot similar to Figure 4.5a. The set inside the blue boundary does not look convex. But if we compute its inverse and multiply by the constant power term, we obtain a convex set shown in Figure 4.5b. Therefore, fixing airspeed is feasible.



(a) $V_{ground}(\bar{\boldsymbol{x}}, \boldsymbol{u})$.



(b) Energy cost $g_{energy}(\bar{\boldsymbol{x}}, \boldsymbol{u})$.

Figure 4.5: 2D visualization of the energy cost at constant airspeed in polar coordinate at $V_{air} = 10m/s$ and $V_{wind} = 5m/s$ with wind blowing along the $\theta = 0°$ outward direction.

Again, to keep things simple, in the simulated examples in Section 4.6, we further restrict the airspeed to be independent of position, so that it is a constant given by equation (4.15) for any flight planning task. This simplification enables us to optimize the airspeed by iteration (Section 2.6.5). Nevertheless, more advanced examples can actually specify an airspeed map as a function of $\boldsymbol{x}$, but designing an optimal airspeed map is a challenging task and out of the scope of this dissertation.

$$\begin{aligned} V_{air} &= const \\ V_{air} &> V_{wind,max} \end{aligned} \tag{4.15}$$

There are two good reasons why we want to set the airspeed constant. First, it makes the energy consumption constant given a vehicle-battery combination (Chapter 3). Therefore, we know that the UAS will not have power failures due to excessive current draw in high wind areas. Second, it works well with the steady-state assumption in our power consumption model.

4.3.4 Time Cost Profile

To design the time cost profile, we first re-write the time differential

$$dt = \left(\frac{dt}{ds}\right) ds = \frac{1}{V_{ground}} ds \tag{4.16}$$

Therefore, if we set the temporal cost $c(t)$ to be 1, the total cost becomes the time spent in the cruise phase of the trip. And the spatial time cost g_{time} becomes

$$g_{time}(\boldsymbol{x}, \boldsymbol{u}) = \frac{1}{V_{ground}(\boldsymbol{x}, \boldsymbol{u})} = \frac{1}{||\boldsymbol{V}_{air}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{V}_{wind}(\boldsymbol{x})||} \tag{4.17}$$

The wind velocity term $\boldsymbol{V}_{wind}(\boldsymbol{x})$ is assumed time invariant (Section 4.3.1), so that it is only a function of position. To ensure the convexity of CP_{time}, we can again either fix the ground speed or airspeed.

4.3.4.1 Constant Ground Speed

If we make the ground speed independent of navigation direction $\boldsymbol{u}$, then $g_{energy}(\boldsymbol{x}, \boldsymbol{u})$ at a fixed point $\bar{\boldsymbol{x}}$ is a constant in equation (4.18) at any given point $\bar{\boldsymbol{x}}$, and the cost profile becomes isotropic (Figure 4.18). This problem can readily be solved by the Fast Marching Method [115]. If we further assume that the ground speed is independent of position $\boldsymbol{x}$, the solution becomes trivial.

$$g_{energy}(\boldsymbol{x}, \boldsymbol{u}) = \frac{1}{V_{ground}(\boldsymbol{x})} \tag{4.18}$$

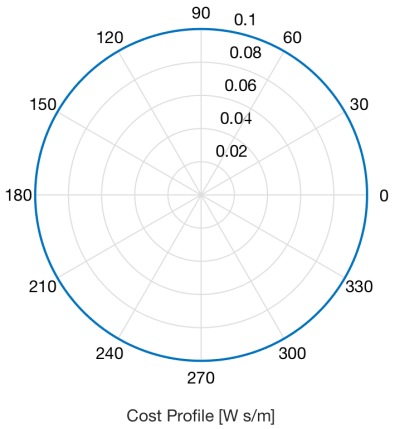


Figure 4.6: 2D visualization of the time cost at constant ground speed in polar coordinate at $V_{ground} = 10m/s$.

4.3.4.2 Constant Airspeed

If we make the airspeed independent of navigation direction $\boldsymbol{u}$, then the power term P becomes a constant at a fixed point $\bar{\boldsymbol{x}}$. Therefore, the cost profile becomes a scaled version of (4.12). The optimal paths of the two fixed-airspeed formulations are therefore equivalent. The minimum costs are only scales by a conversion factor between energy and time. A visualization of the cost profile is provided in Figure 4.7. It is simply a scaled version of Figure 4.5b.

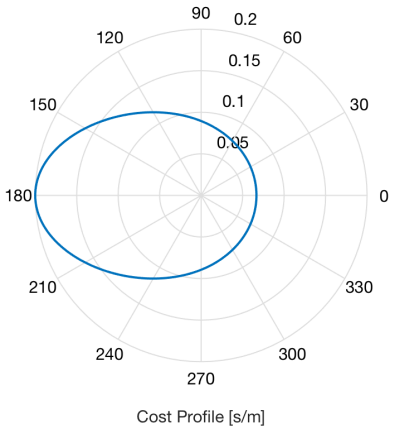


Figure 4.7: 2D visualization of the time cost at constant airspeed in polar coordinate at $V_{air} = 10m/s$ and $V_{wind} = 5m/s$ with wind blowing along the $\theta = 0°$ outward direction.

4.3.5 Safety Consideration

For safety considerations, we impose infinite costs on top of the nominal cost $g_{nominal}$ in regions where the wind speed exceeds a maximum threshold $V_{wind,max}$, so that the optimal trajectory does not enter these regions.

$$g(\boldsymbol{x}, \boldsymbol{u}) = g_{nominal}(\boldsymbol{x}, \boldsymbol{u}) + I_{inf}(||\boldsymbol{V}_{wind}(\boldsymbol{x})|| > V_{wind,max}) \tag{4.19}$$

where the function $I_{inf}(\cdot)$ is defined as

$$I_{inf}(\text{condition}) = \begin{cases} \infty, & \text{if condition is true} \\ 0, & \text{otherwise} \end{cases} \tag{4.20}$$

4.3.6 The Simplified Optimal Control Problem

With the cost profiles defined above, we can simplify problem (4.1) to problem (4.21) by re-writing the cost function as a line integral in terms of the cost profile element $g(\boldsymbol{x}, \boldsymbol{u})$. The flight path is parametrized by variable s.

$$\min_{\boldsymbol{u}(s)} \quad \left[J = \int_{s_0}^{s_f} g(\boldsymbol{x}(s), \boldsymbol{u}(s)) ds \right] \tag{4.21a}$$

$$s.t: \quad \frac{d}{ds}\boldsymbol{x}(s) = \boldsymbol{u}(s) := \frac{\boldsymbol{V}_{ground}}{V_{ground}} \tag{4.21b}$$

$$\boldsymbol{x}(s) \in \Omega \tag{4.21c}$$

$$\boldsymbol{x}(s_0) = \boldsymbol{x}_0 \in \Omega \tag{4.21d}$$

$$\boldsymbol{x}(s_f) = \boldsymbol{x}_f \in \partial\Omega \tag{4.21e}$$

$$\boldsymbol{u} \in \mathcal{U} = \{\boldsymbol{u} \in \mathbb{R}^n \,|\, ||\boldsymbol{u}|| = 1\} \tag{4.21f}$$

Depending on the choice of the cost $g(\boldsymbol{x}(s), \boldsymbol{u}(s))$ (Section 4.3.2) in equation (4.21a), the cost can be either energy consumption or flight duration at the cruise phase. In both cases, the cost function depends on not only the position $\boldsymbol{x}(s)V_{ground}$ but also the navigation direction $\boldsymbol{u}(s)$. The control variable is the sequence of the navigation direction $\boldsymbol{u}$ in different parametrized positions s. They are unit vectors of length 1 as is shown in equation (4.21f). The dynamic constraint in equation (4.21b) says that the navigation direction $d\boldsymbol{x}(s)/ds$ is the same as $\boldsymbol{u}(s)$. Lastly, we still have the feasible region (equation (4.21c)), initial condition (equation (4.21d)), and terminal condition (equation (4.21e)).

The optimal path can be found in two steps. First, we solve the HJB equation numerically and build an optimal cost-to-go map $J^*(\boldsymbol{x})$ associated with a specific destination, as well as the corresponding optimal control map $\boldsymbol{u}^*(\boldsymbol{x})$. The computation is performed backward from the destination point to all the other points. This problem is addressed in Section 4.4. Second, we generate the optimal paths by looking up and apply the optimal control actions step-by-step from the $\boldsymbol{u}^*(\boldsymbol{x})$ map, from some origins to the specified destination [116].

4.4 Numerical Algorithm: Ordered Upwind Method (OUM)

The Ordered Upwind Method (OUM) [116, 117] is an efficient algorithm to find a numeric approximation, or an optimal cost map $\tilde{J}(\boldsymbol{x})$, to the viscosity solution of equation (4.5), over a discretized configuration space. In this section, we review this method.

The discretized space is divided to three types of vertices: *Far*, *Considered*, and *Accepted* [117]. Figure 4.8 shows the vertex labels during the computation.

- ***Far*** - Computation of $\tilde{J}$ has not yet started.
- ***Considered*** - Tentative values of $\tilde{J}$ has assigned.
- ***Accepted*** - Finalized values of $\tilde{J}$ has assigned.

Then we define the following terms to introduce the algorithm.

- ***Accepted Front*** - *Accepted* vertices that has a *Considered* neighbor.
- ***AF*** - Edges made of neighboring vertices pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ where $\boldsymbol{x}_i \in$ *Accepted Front* and $\boldsymbol{x}_j \in$ *Considered*.
- ***Near Front*** - Let $\boldsymbol{x}_i \in$ *Considered* and $\boldsymbol{x}_j, \boldsymbol{x}_k \in$ *Accepted Front*, then the near front $NF(\boldsymbol{x}_i)$ is defined as

$$\{(\boldsymbol{x}_j, \boldsymbol{x}_i) \in AF \,|\, \exists \boldsymbol{x} \text{ on } (\boldsymbol{x}_j, \boldsymbol{x}_k) \text{ s.t. } ||\boldsymbol{x} - \boldsymbol{x}_i|| \leq \Gamma h_{max}\}$$

with Γ being the anisotropic ratio [117].

In the triangle $(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k)$, where $\boldsymbol{x}_i \in \textit{Considered}$ and $\boldsymbol{x}_j, \boldsymbol{x}_k \in \textit{Accepted Front}$, define the minimum cost approximation $\hat{J}_{\boldsymbol{x}_j,\boldsymbol{x}_k}(\boldsymbol{x}_i)$ to be

$$\begin{aligned} \hat{J}_{\boldsymbol{x}_j,\boldsymbol{x}_k}(\boldsymbol{x}_i) &= \min_{\lambda} \left[\hat{J} + \Delta \boldsymbol{x}_i \; g(\boldsymbol{x}_i, \boldsymbol{u}_i) \right] \\ \hat{J} &:= \lambda \hat{J}(\boldsymbol{x}_j) + (1-\lambda)\hat{J}(\boldsymbol{x}_k) \\ \Delta \boldsymbol{x}_i &:= [\lambda \boldsymbol{x}_j + (1-\lambda)\boldsymbol{x}_k] - \boldsymbol{x}_i \end{aligned} \tag{4.22}$$

The high-lighted triangle in Figure 4.8 illustrates this optimization. The minimum costs of accepted vertices $\boldsymbol{x}_j$ and $\boldsymbol{x}_k$ are known, and the cost of vertex $\boldsymbol{x}$ inside edge $\boldsymbol{x}_j\boldsymbol{x}_k$ is obtained from linear interpolation. The cost of vertex $\boldsymbol{x}_i \in \textit{Considered}$ is the sum of the cost of vertex $\boldsymbol{x}$ and $\Delta \boldsymbol{x}_i \; g(\boldsymbol{x}_i, \boldsymbol{u}_i)$. And the optimal control $\boldsymbol{u}$ is given by the direction from $\boldsymbol{x}_i$ to $\boldsymbol{x}$.

Then the tentative cost update for the vertex $\boldsymbol{x}_i \in \textit{Considered}$ is

$$\hat{J}(\boldsymbol{x}_i) = \min_{(\boldsymbol{x}_j,\boldsymbol{x}_k) \in NF(\boldsymbol{x}_i)} \hat{J}_{\boldsymbol{x}_j,\boldsymbol{x}_k}(\boldsymbol{x}_i) \tag{4.23}$$

With the definitions above, the Ordered Upwind Method is summarized as follows:

1. Start with all vertices in *Far*, $\hat{J}(\boldsymbol{x}_i) = \infty, \; \forall \boldsymbol{x}_i \in \Omega$.
2. Move the destination vertex $\boldsymbol{x}_f$ to *Accepted* with $\hat{J}(\boldsymbol{x}_f) = 0$.
3. Move all vertices $\boldsymbol{x}_i$ adjacent to $\boldsymbol{x}_f$ into *Considered* and evaluate the tentative cost $\hat{J}(\boldsymbol{x}_i)$ from equation (4.22) and (4.23).
4. Find the vertex $\boldsymbol{x}_i^* = argmin_{\boldsymbol{x}_i \in Considered}[\hat{J}(\boldsymbol{x}_i)]$, move it to *Accepted*, and update the *Accepted Front*.
5. Move all vertices in Far adjacent to $\boldsymbol{x}_i^*$ into *Considered*.
6. Recompute the cost for all vertices $\boldsymbol{x}_i$ within Γh_{max} from $\boldsymbol{x}_i^*$. If the newly-computed cost is less than the previous tentative cost, then update the cost.
7. If $\textit{Considered} \neq \emptyset$, then go to step 4.

Lastly, we can use equation (4.24) to attach timestamps to the generated flight path to make it a 4D trajectory defined in NextGen.

$$t(s) = \int_{s_0}^{s} \frac{d\bar{s}}{V_{ground}} \tag{4.24}$$

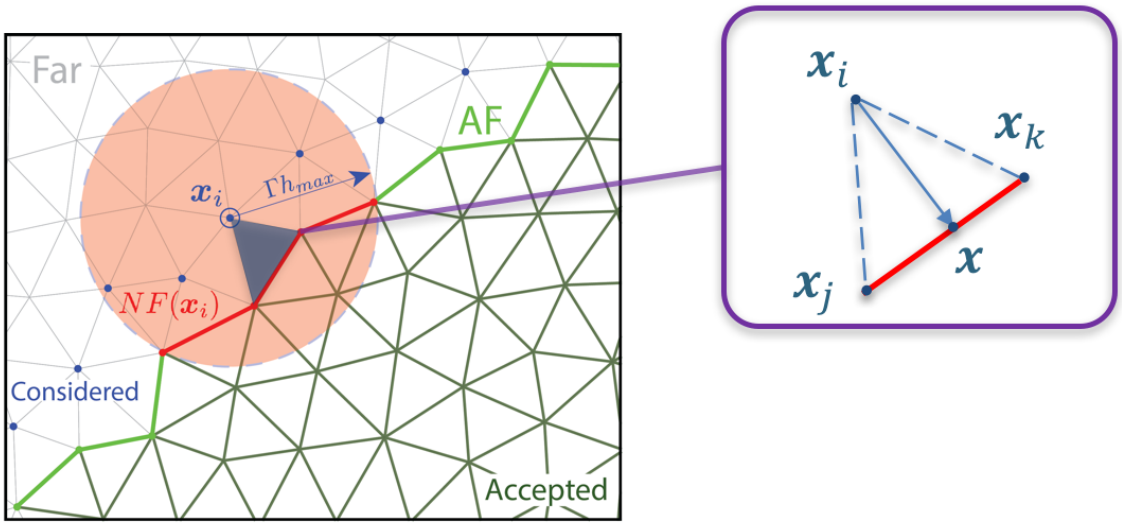


Figure 4.8: Labels introduced for a general triangulated mesh to describe the OUM algorithm. The color of each term matches the graphical representation.

4.5 Environmental Data

We are interested in environment data that has significant impact on our FPS. To avoid static obstacles, we need terrain data so that the UAS knows where to fly. To estimate energy demand, we need wind information in the flight region. In this section, we review the current development on these environment data and compare to the requirement for UAS flight planning.

4.5.1 Terrain Data

To avoid static obstructions such as terrain and buildings, we run the algorithms on digital elevation models (DEM) [77]. DEMs are simply images with altitude information. After some post-processing, they can be stored as triangulated meshes with lists of vertices, edges, and faces. Large scale terrain DEM can be obtained from NASA ASTER GDEM2 [85]. The horizontal resolution of the dataset is $30m$ horizontally, which suffices for most of the flight planning tasks.

For UAS routing, we elevate the DEM model upward to the desired flight altitude. In our case, this altitude is set to $150m$ ($500ft$) AGL due to the current low-altitude requirement imposed by the FAA (Section **??**). Other choices of altitude are also possible depending on future regulation requirements on altitude limit and vertical separation. The vertical resolution of wind data an also play a big role. Then, we smooth the surface to avoid sharp change in altitude. It is necessary because disrupt changes in altitude may destabilize the UAS and render it unsafe. The resulting mesh is a 3D surface on which we perform the

routing computation. The optimal paths live inside this elevated surface, and thus will not hit mapped static obstacles.

One draw back of this approach is that the DEM maps are not updated in real-time. If there were unmapped new buildings, then the UASs are still prone to collisions. Therefore, before planning, it is important to ensure to flight regions has up-to-date terrain information. The mapping problem is not the focus of this dissertation. We only address the routing problem assuming the terrain data is accurate.

4.5.2 Wind Data

One of the biggest challenges in UAS flight planning is the lack of wind and weather data [64] in the low-altitude Class G airspace. Wind data at the boundary layer, or low-altitude airspace, is commonly collected with radar or sodar-based wind profilers [30, 100]. Currently, the best available wind forecast data is from the High-Resolution Rapid Refresh (HRRR) radar assimilation [121]. The finest horizontal resolution is about $3km$. This resolution is too coarse because the total travel distance is small in most UAS missions, typically less than $5km$. It implies that we only have one or two wind data points to plan the routes. Regardless of the routing methods, such low-resolution wind data fundamentally limits the quality of the trajectory output.

Due to terrain geometry, the wind behavior at low altitude is quite complicated. Studies have conducted on verifying the accuracy of HRRR [135], but the result does not seem good. In addition, the infrastructure required for data collection is cumbersome and expensive. Although the focus of this dissertation is not on improving the wind data resolution, we still need to find a better alternative of wind data to test the routing algorithm.

In the current implementation of our flight planning system, we use the realistically-simulated high-resolution 3D wind field from the Environmental Fluid Mechanics and Hydrology (EFMH) group at UC Berkeley [32]. The horizontal, vertical, and temporal resolution are $100m$, $50m$ and $10min$, respectively.

4.6 Simulation

In this section, we simulate three flight planning scenarios: one shows the obstacle avoidance capability, the other two presents the energy and time optimality under the influence of wind. The UAS is a 3DR IRIS+ with a $5100mAh$ LiPo battery. The total self weight with gimbal and battery is $14N$. In the end, we comment briefly on the implementation details and bottlenecks.

4.6.1 Obstacle Avoidance Scenario

Figure 4.9 shows a simple minimum-energy flight planning example without wind on a $600m \times 600m$ space with two tall buildings as obstacles and a constant slope on the ground. The

origin is at the upper-right corner from the top view, while the destination is the lower-left corner. The minimum-energy cost map is propagated backward from the destination to the origin. We restrict the UAS to fly on the elevated 2D surface at $150m$ AGL. The computed paths avoid the building obstacles successfully.

The UAS travels at a fixed ground speed of $20m/s$. The optimal cost, or minimum energy, to destination is about $20.8kJ$ at the cruise phase, or $482mAh$ with a $12V$ 3-cell LiPo battery. The energy spent during take-off and landing can be added to obtain the total energy consumption. In this case, assume that the UAS is flying at $150m$ above ground level (AGL) at $2.5m/s$ vertical speed, then the energy spend on taking-off and landing is $19.8kJ$, which yields a total of $40.6kJ$, or $940mAh$, for the entire trip.

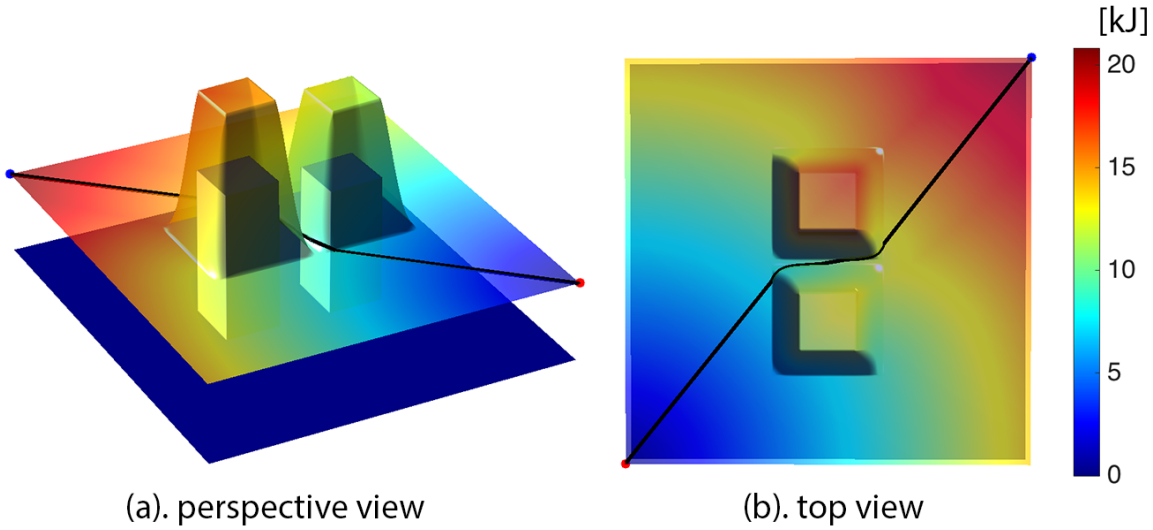
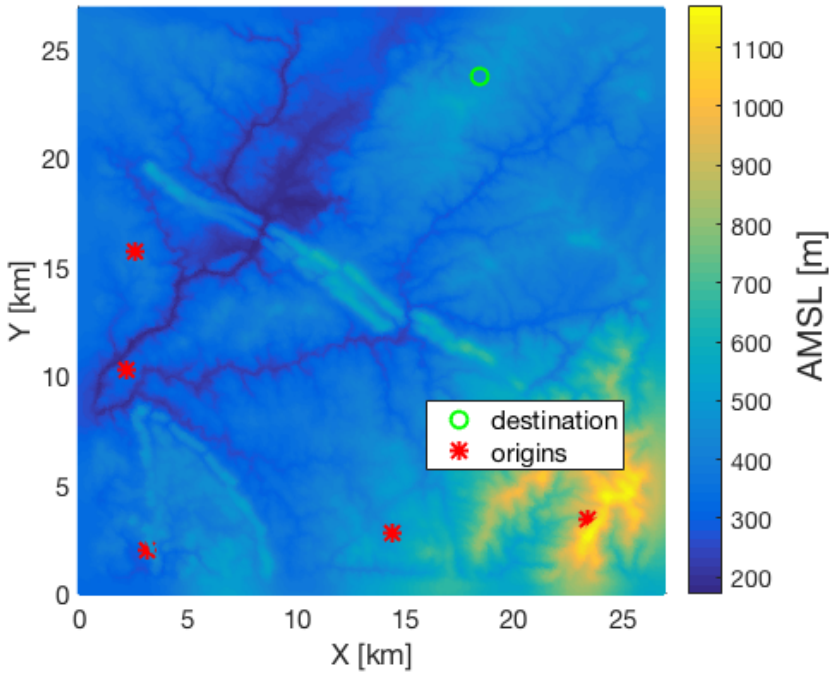
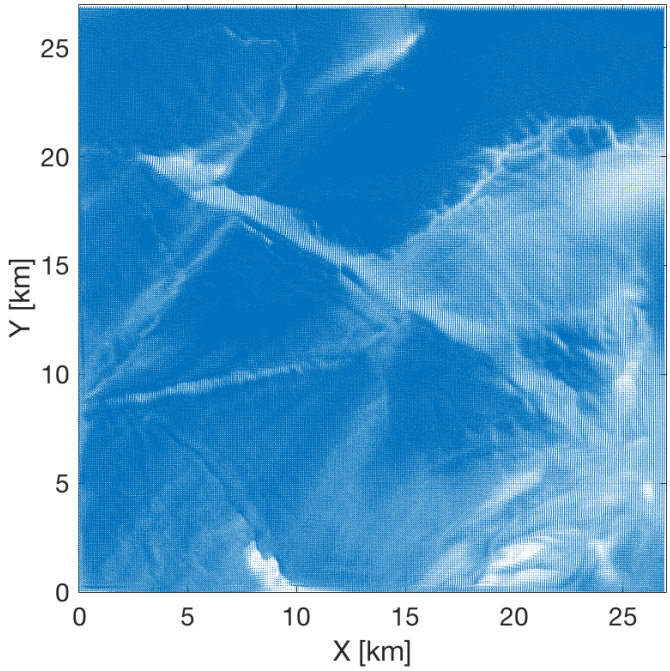


Figure 4.9: Optimal energy cost map and the optimal path without wind.



(a) Digital elevation model (DEM). The color represents altitude above mean sea level (AMSL). In our simulations, we use the green circle as the destination, and the red stars as the origins.



(b) 3D wind field data at $141m$ AGL.

Figure 4.10: Terrain and wind data from the the EFMH group.

4.6.2 Minimum Energy Scenario in Wind

As is mentioned earlier, real-time wind data collection infrastructure is largely missing, therefore an experimental testing is not yet possible. Nevertheless, we manage to obtain high fidelity wind data for simulation. In the following, we first describe the data briefly, then present the simulation results.

4.6.2.1 Terrain and Wind Data

For short trips (say $< 20km$), high-resolution wind field is desired to obtain good results because the trip length is small (Section 4.5.2). But the best publicly available high-resolution rapid refresh (HRRR) wind database is still too coarse ($3km$ horizontally) for our application [120]. Instead, we use realistically-simulated 3D wind field ($27km \times 27km$) from the Environmental Fluid Mechanics and Hydrology (EFMH) group at UC Berkeley [32]. The wind field is simulated from the current stare-of-the-art large-eddy simulation (LES) model for the atmospheric boundary layer [21]. The terrain and wind data for our simulation is a cropped-subregion taken from 23 : 00 to 00 : 00 UTC on May 19, 2017 (Figure 4.10) at Perdigao in Portugal. It gives horizontal, vertical, and temporal resolution of $100m$, $50m$ and $10min$, respectively. We chose wind data at about $141m$ AGL. Observe that the wind field has an irregular structure inherited directly from the terrain profile. A simple polynomial parametrization is hard to capture the irregularities.

4.6.2.2 Scenario Setup

We choose one destination and five different origins to perform the OUM computation (Figure 4.10a). We observe to choose our ground speed. The maximum and median wind speeds are $18.2m/s$ and $9.7m/s$, respectively. The IRIS+ can fly at $20m/s$ based on the performance data in Chapter 3. Therefore, we picked a conservative ground speed of $5m/s$.

4.6.2.3 Result and Analysis

Figure 4.11 compares several 3D paths at $150m$ AGL on the XY plane. The flight distances range from $16km$ to $27km$. The paths in black are the minimum energy paths taken wind into account, while the paths in gray are without considering wind. The non-wind-optimal paths are very close to straight line paths because the vertical terrain variation ($\sim 100m$) is small compared to the horizontal spatial scale ($\sim 5km$). But the wind-optimal paths, such as paths #1 to #3, can be quite different from straight lines.

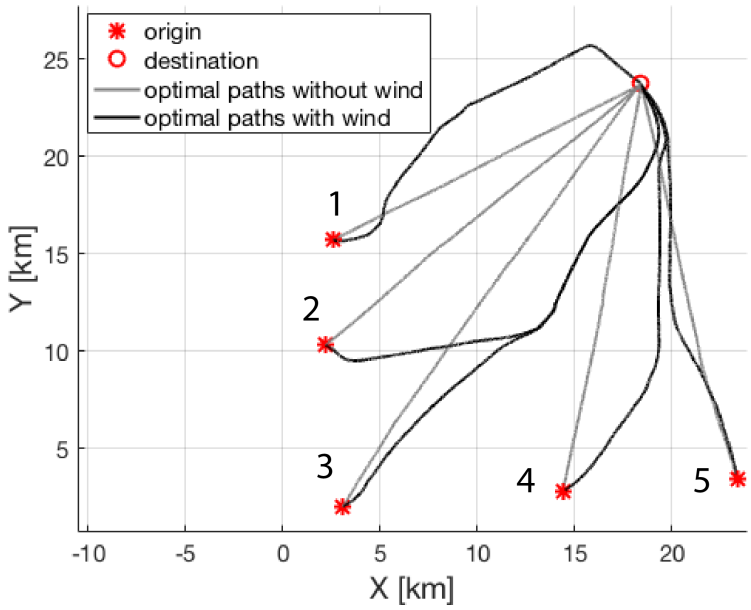


Figure 4.11: Paths comparison between wind-optimal and non-wind-optimal paths.

Similar to the paths, the energy savings vary quite a bit, ranging from 2% to 40% (Table 4.1). The largest energy saving is achieved in path #2. Without considering wind, the energy spent is $1611.7kJ$, while the wind-optimal path only spends $984kJ$, giving an overall saving of 39%. In the following, we take Path #2 as an example to illustrate reasons contributing to the large energy savings.

The time history of the energy and power consumption of path #2 is plotted in Figure 4.12. Observe that the wind-optimal path in red yields significantly lower power in several time durations by taking advantage of wind, resulting in less energy consumption overall.

Table 4.1: Energy saving comparison between optimal paths with and without considering wind.

Path #	Consider wind [kJ]	Not consider wind [kJ]	% saving
1	845.0	1259.3	32.9%
2	984.0	1611.7	39.0%
3	1224.4	1664.7	26.4%
4	867.9	975.2	11.0%
5	768.0	750.7	2.2%

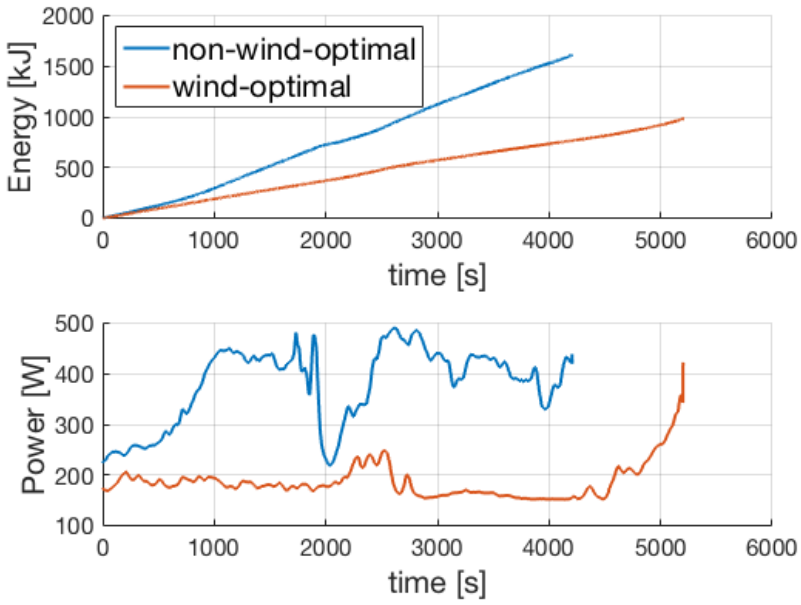


Figure 4.12: Energy consumption comparison between wind-optimal and non-wind-optimal paths.

Figure 4.13 explains why the wind-optimal path gives significant energy savings. First, the algorithm tries to avoid high wind regions. In the non-wind-optimal case, the path is almost completely against high wind, from 11 to $16.5m/s$, in the entire trip, while in the wind-optimal case, the path is mostly located in low-wind areas, from 5 to $11m/s$. Second, the UAS tries not to fly against the wind when high wind is unavoidable. At the beginning and near the end of the trip, where strong wind is present, the wind-optimal path flies almost perpendicular to the high wind, giving significantly smaller airspeed and power.

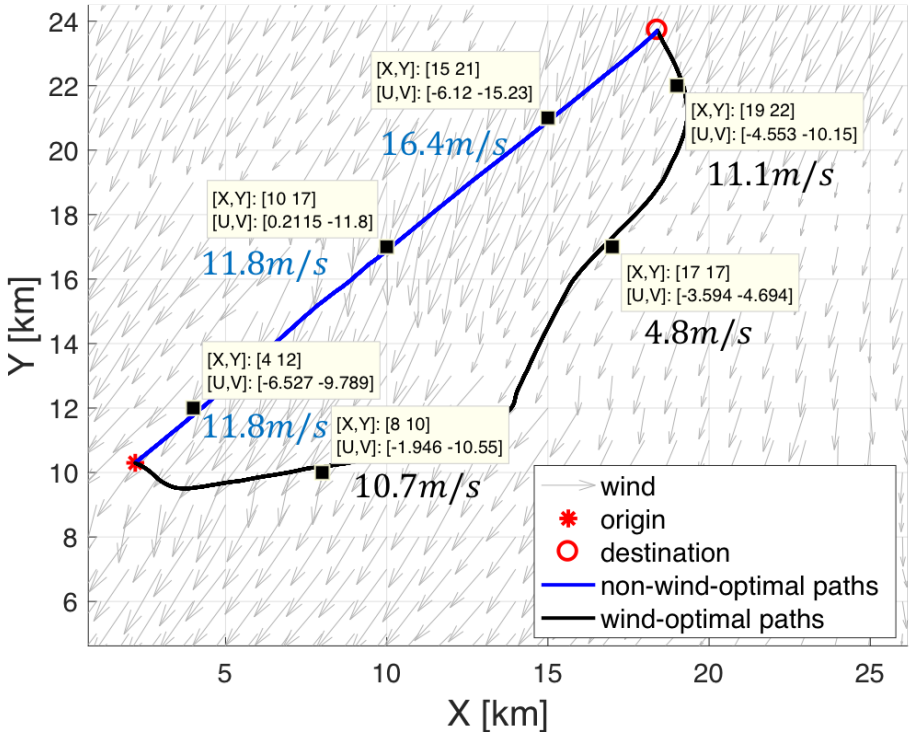


Figure 4.13: The minimum energy path #2 with and without considering wind. The wind field in gray is taken at 23 : 00 UTC on May 19, 2017 at Perdigao.

Table 4.2 shows a snapshot of the 4D trajectory with timestamps after applying equation (4.24) to flight path #2, together with the expected energy consumption.

Table 4.2: A snapshot of the flight trajectory together with energy estimate at the cruise phase of Trip #2.

Timestamp $[hr : min : sec]$	Way point [m]	Energy consumed [kJ]
13 : 00 : 00.0	(2200.0, 10300.0, 250.4)	0.00
13 : 00 : 03.5	(2245.8, 10278.3, 257.7)	1.74
13 : 00 : 05.5	(2272.7, 10265.3, 261.1)	2.76
...	...	...

4.6.3 Minimum Time Scenario in Wind

We simulate the same scenarios to obtain minimum time trajectories. Figure 4.14 compares the 3D paths at $150m$ AGL on the XY plane. The IRIS+ can fly at a constant airspeed

of $20m/s$. The paths in black are the minimum time paths taken wind into account, while the paths in gray are without considering wind. The non-wind-optimal paths are very close to straight line paths. But the wind-optimal paths have shapes that are very similar to the minimum energy paths in Figure 4.11.

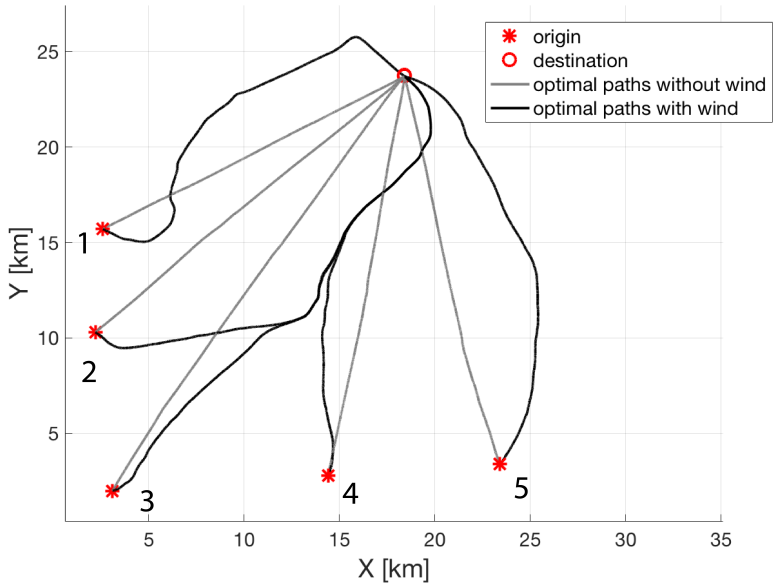


Figure 4.14: Paths comparison between wind-optimal and non-wind-optimal paths.

The time savings is significant in general (Table 4.3). The largest time saving is achieved in path #2. Without considering wind, the duration is 68 minutes, while the wind-optimal minimum-time path only spends 34 minutes, giving a 51% shorter duration. Path #1 also shows a 47% time savings. The other three paths also have time savings ranging from 9% to 30%.

Table 4.3: Time saving comparison between optimal paths with and without considering wind.

Path #	Consider wind [min]	Not Consider wind [min]	% saving
1	28.06	52.96	47.0%
2	33.72	68.20	50.6%
3	44.93	65.49	31.4%
4	31.80	36.98	14.0%
5	25.45	27.86	8.6%

The time history of the ground speed and power consumption of path #2 is plotted in Figure 4.15. The power stays constant as expected because the airspeed is a constant. The ground speed in the wind-optimal path is in general faster than the non-wind-optimal path. In particular, between time $0min$ and $32min$ on the time axis, we see consistent boosts in ground speed, which results in the time savings even when the distance is slightly longer in the wind-optimal path.

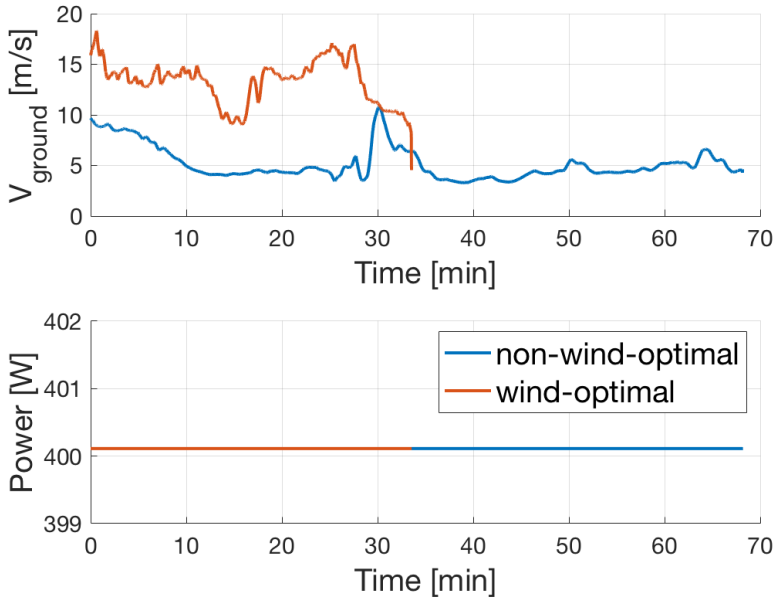


Figure 4.15: Energy consumption comparison between wind-optimal and non-wind-optimal paths.

Figure 4.16 shows paths comparison for origin #1 and #2 in the wind field. And again, the time saving is attributed to avoiding and flying perpendicular to high wind regions, and the strategy is very similar to the minimum energy case. Note that although the minimum-energy and minimum-time paths for origin #2 are very similar, the flight times are drastically different. This is because we are fixing the ground speed when minimizing energy, while fixing the airspeed when minimizing time.

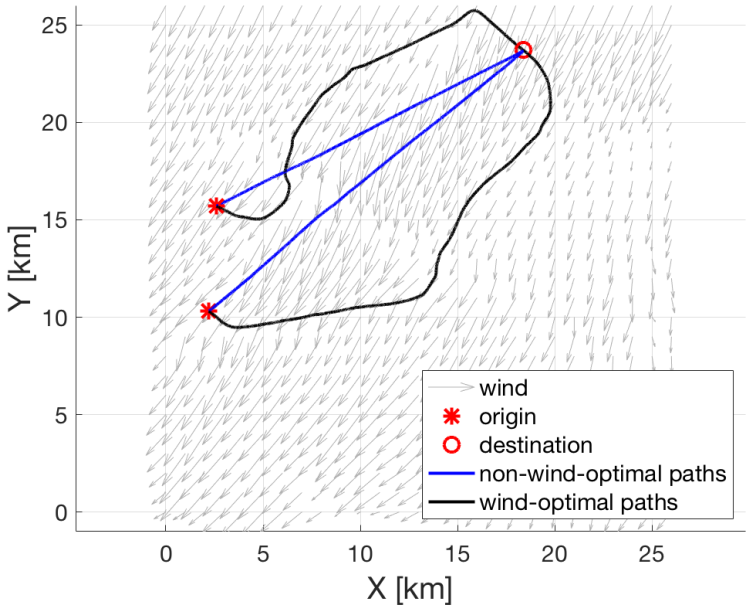


Figure 4.16: The minimum time path #3 with and without considering wind in the wind field. The wind field in gray is taken at 23 : 00 UTC on May 19, 2017 at Perdigao.

4.6.4 Implementation Details and Bottlenecks

Since there is no existing libraries available, we wrote our own implementation of the OUM algorithm in Julia language [58]. Julia is known to be fast in scientific computing but easier to code than C/C++. Therefore, it is ideal for prototyping our algorithm. To efficiently store the *Considered* vertices, we applied a minimum heap data structure. The computation of equation (4.22) is performed using the golden-section search algorithm [99] as recommended in [116].

The most prominent bottleneck is in the computation time, specifically the time spent on solving equation (4.22). Take the scenarios in Section 4.6.2 as examples. The terrain data has a range of $27km \times 27km$ and a 100-meter horizontal resolution, giving $(270)^2 = 72900$ vertices in the Delaunay-triangulated mesh. The total computation time on generating the minimum energy and time cost map are 9.7 minutes and 2.0 hours, respectively. After some profiling, we saw that a large fraction of the time, about $\sim 50\%$ in the energy formulation, was spent on computing the vector norms inside the golden-section search iterations. In the minimum time formulation, there are more iterations because the ground speed is not a constant. To reduce the computation time, we can possibly adopt sweeping algorithms that are parallelizable [129]. Another bottleneck is the necessity to pre-specify the ground speed or airspeed before running the algorithm. Unless there is a better formulation of the optimal control problem, this issue is difficult to overcome.

Part II

In-Flight

Chapter 5

4D Trajectory Following

In this chapter, we would like to design controllers capable of following the trajectories defined in Chapter 4. We first review the trajectory following literature in manned aviation (Section 5.1). Then we give an overview of the approach in Section 5.2.1. Then, we present the controller designs in Section 5.3 and 5.4. Lastly, we present the simulation results in Section 5.5.

5.1 Trajectory Following in Manned Aviation

After planning a trajectory, we want to make sure that the aircraft follows the trajectory as closely as possible. Figure 5.1 shows an example of a flight plan from Delta airline. The example plans a trip from the Seattle-Tacoma International Airport (SEA) to the Amsterdam Airport (AMS). There are three critical information highlighted in red: waypoints (VOR) in latitude and longitude, elapsed/remaining trip time, and consumed/remaining fuel.

Aircraft trajectory following is performed manually by pilots in manned aviation [88]. Pilots flies the aircraft in straight lines from one VOR to another, at specified ground speeds. The pilot in command of the mission compares the time and fuel readings to the flight plan while reaching each VOR. The ground speed and tentative schedule gets updated accordingly to fulfill the remaining flight plan.

Unlike ground transportation, the challenge on airplane trajectory following is on positioning, or sensing, the aircraft rather than controlling it. To position the aircraft within good accuracy, the U.S. Federal Aviation Administration's (FAA) defines visual flight rules (VFR) and instrument flight rules (IFR). At low altitude and under clear weather conditions, we can fly an airplane solely by reference to outside visual cues, such as the horizon to maintain orientation, nearby buildings and terrain features for navigation, and other aircraft to maintain separation. On the other hand, without clear visual cues, commercial airliners and their pilots in Class A airspace are required to operate under IFR. In NextGen, the FAA mandates GPS-based navigation via Automatic Dependent Surveillance-Broadcast (ADS-B), specifically ADS-B Out, by year 2020 [92]. ADS-B Out broadcast the aircraft's

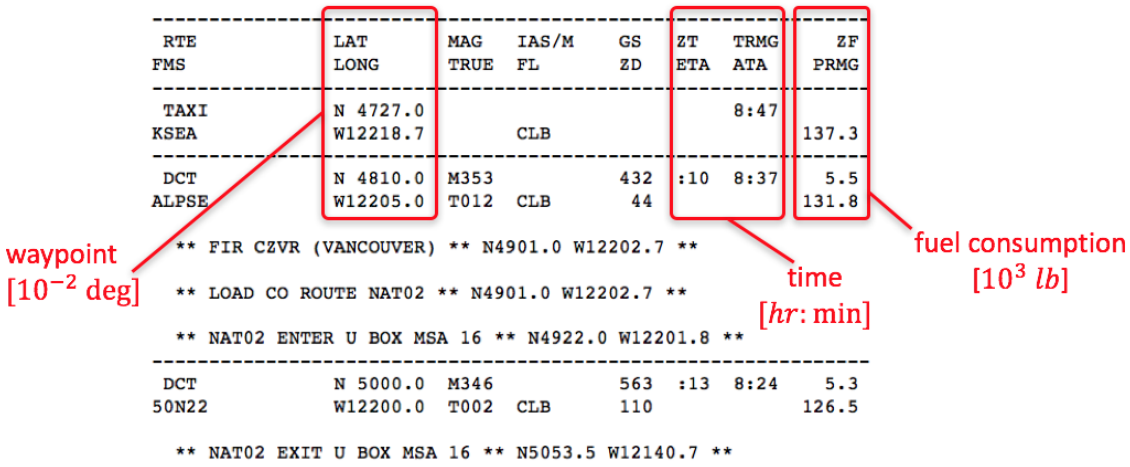


Figure 5.1: A flight plan example from Delta airline.

GPS information, while ADS-B In subscribes this information from nearby aircraft. The introduction of ADS-B improves safety and efficiency in the air and on runways and reduces costs.

However, for UASs, trajectory following has to be fully autonomous without human interferences. Therefore, there are difficult challenges in both sensing and control. The sensing problem in open areas is mostly resolved by the introduction of GPS and IMU. But at low-altitudes, there are complex static and dynamic obstacles. Small UASs with limited sensors generally cannot handle such environments trivially. We propose to avoid static obstacles in the pre-flight stage with high-definition terrain models. Therefore, unlike manned aviation, the trajectories will not be simple straight line segments connecting a sparse set of waypoints. Instead, they can have curly segments with dramatic altitude variations, which imposes great challenges in control. The UASs have to follow the planned trajectory closely even under the uncertainty of wind and weather while maintaining safety. In this chapter, we address the trajectory following problem of a quadrotor.

5.2 Overview

5.2.1 Definition of 4D Trajectory Following

To track the 4D trajectories generated in Chapter 4, we need to design appropriate feedback controllers. In general, we have two strategies, namely trajectory following and path following. A 4D trajectory is a path with timestamped 3D waypoints. In trajectory following, we can track the time evolution of a path directly. However, for long distance travel, the planned trajectory may not be a good representation of the true trajectory. For example, if the UAS tried to avoid other vehicles along the flight path, then the timestamps in the

pre-planned trajectory would be outdated. Continue to follow the outdated trajectory after collision avoidance can be problematic since the current vehicle position can be far away from the planned position. On the other hand, path following tracks a 3D path without time constraints. Therefore, at each time step, the UAS re-plans a short-term 4D trajectory along the path to follow. This approach can overcome the disadvantage of trajectory following mentioned above.

In this dissertation, we would like to re-define the technical term trajectory following in the vehicle control literature slightly to make it compatible with manned aviation. NextGen introduces 4D trajectory, or a time-stamped 3D path, as a unifying term for both manned and unmanned aviation. A pilot is essentially performing path following in manned aviation because there is trajectory re-planning taking place constantly in the pilot's mind. But there is also a sense of time in the path because the pilot also adjusts the flying speed so that the actual arrival time is close to the planned arrival time. Therefore, the pilot is also following a trajectory in this sense. We would like to adopt a looser definition of a 4D trajectory as a sequence of 3D waypoints stamped with desired speeds. This speed-stamped trajectory definition is similar to the timestamped one if there were not significant deviation between the actual and planned flight trajectory. In this dissertation, we use the term 4D trajectory following to mean tracking 3D waypoints with speed commands.

5.2.2 The Proposed Control Strategy

In this Chapter, we break the 4D trajectory following problem into two levels of feedback control (Figure 5.2). First, a high-level controller takes the cross-track and heading error of the UAS with respect to the desired path, and generates an acceleration and a heading command. Since the speed-stamped reference trajectories contains future information, we can take advantage this knowledge by applying model predictive control (MPC) in the high-level controller. It is particularly effective in reducing over-shoot when we are transitioning to high-curvature segments. But to solve the receding-horizon optimal control problem at a reasonable speed, we have to keep the model relatively simple. We discuss the high-level MPC controller design in Section 5.4. By assuming the UAS is under vertical equilibrium, we can convert the acceleration command to Euler angle commands.

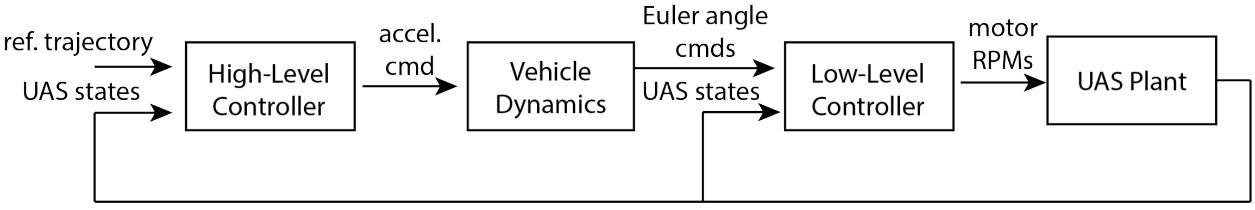


Figure 5.2: The feedback controller design block diagram for 4D trajectory following.

Multi-rotor dynamics are known to be nonlinear and underactuated. The excessive computation burden on handling such complications in optimization determines that we could

not solve the full control problem solely based on MPC. Therefore, we introduce a low-level controller to track the desired Euler angles commands by producing forces and moments, or motor RPMs. Sliding mode control is a well-known robust nonlinear control approach. It provides a good robustness margins to handle various uncertainties. The control laws are analytical and thus computationally very fast. Therefore, it is the ideal low-level controller candidate. The disadvantage is that it still requires a reasonably accurate dynamic model and the associated model parameters of the UAS. In Section 5.3, we present a general set of robust sliding controllers to achieve both Euler angles and translational position (XYZ) tracking.

Lastly, upon reaching the destination, it is more effective to track a static position than speed. Therefore, we introduced a transition to position tracking when reach the final waypoint (Figure 5.3), to provide faster and smoother stabilization.

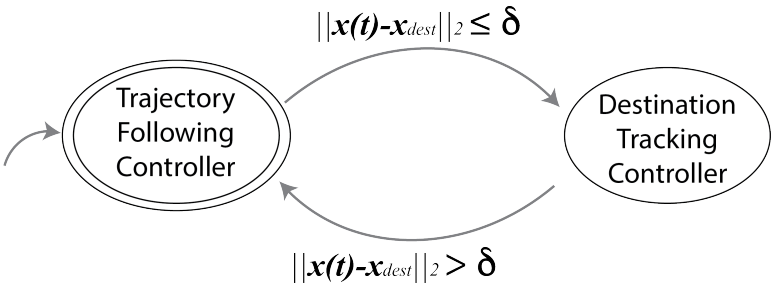


Figure 5.3: The finite state machine, switching between trajectory following and destination tracking.

5.3 Low-Level Sliding Controller

In this section, we aim to design a robust low-level controller (Section 5.2.2) for a quadrotor UAS. We first review the existing literature (Section 5.3.1) and the dynamic model (Section 5.3.2), then goes into the controller design (Section 5.3.3).

5.3.1 Literature Review

In this section, we review the quadrotor control literature. It can be divided into two classes. The first class focuses on attitude and altitude stabilization [12, 70, 75, 13, 14], while the second class controls the horizontal position as well [67, 137, 127]. Controlling the first class is straight forward because attitude and altitude are directly controlled by four independent control inputs. However, when horizontal position tracking is required, the problem becomes more difficult due to the underactuated nature of quadrotors. In this section, we review both classes of control methods and lead to our contribution.

Attitude and altitude stabilization of a quadrotor could be achieved by many approaches. Classical methods were examined in several papers. In [12], a PID controller and a LQ controller were compared. The PID controller neglected gyroscopic effects but were able to stabilize the system with minor disturbances. The LQ controller gave average performance because the linearized reference model was imperfect. In [70], Li presented another experiment to confirm the feasibility of PID designs. Nonlinear control methods were also widely experimented. In [13], a backstepping and a sliding mode controllers were compared. The backstepping controller gives more robust performance. In [14], an adaptive sliding mode controller was presented. Simulation results indicated that the controller was robust to model uncertainty in mass and rotational inertia.

However, the feasible control methods reduce dramatically when horizontal position tracking is required. The challenge is on resolving the underactuation issue, or using only four independent actuation inputs to handle all six degrees of freedom. Specifically, the tilting in roll and pitch are coupled with horizontal motions. To derive feasible horizontal control laws, the dynamic model is usually greatly simplified [67, 137] to avoid the explosion of terms when taking time derivatives.

In [137], Xu divided the model into two subsystems. The fully actuated system were handled by PID controllers. The underactuated system with horizontal motions was stabilized to a fixed point by a sliding mode controller. However, it was unable to track a moving trajectory because it lacked the time derivatives of the desired roll and pitch. In [67], Lee tried both feedback linearization and adaptive sliding mode to fully control a quadrotor. In feedback linearization, the system was reduced to include only thrust forces and gravity, and yet the derived control laws were quite lengthy. In addition, it augmented the state-space model with control inputs as additional states, yielding complicated dynamical input control laws. In the sliding mode controller, the horizontal position were controlled by two PD controllers, instead. In [127], another similar feedback linearization method with dynamic inputs was presented. Compared to [67], the model in [127] was more complete, and the term explosion problem became huge! In summary, it is practically impossible to derive concise control laws from a full quadrotor model by directly applying nonlinear control methods.

In this Section, we design the low-level controllers capable of tracking both horizontal positions and Euler angles. As we will see later, horizontal (XY) position tracking of a quadrotor is quite difficult. Our approach is to augment the quadrotor model with two first-order filters from the dynamic surface control (DSC) literature [123]. With the DSC filters, we were able to access lagged version of the required time derivatives and indirectly formulate concise control laws, to fully control a quadrotor, including horizontal positions.

5.3.2 Dynamic Model

The dynamic model of a quadrotor is similar to a typical airplane with a different set of forces. Here we use Euler angles instead of quaternion, to express the control laws derived later. One limitation of using Euler angles is that there is a singularity at $\theta = 90°$. Define a fixed north-east-down (NED) inertial world frame $\mathcal{W}$ and a non-inertial body frame $\mathcal{B}$

attached to the center of gravity of the quadrotor (Figure 5.4). The following is a list of variables used to describe the dynamics of a quadrotor.

- $\boldsymbol{X} = [X\ Y\ Z]^T$: quadrotor position in $\mathcal{W}$;
- $\boldsymbol{x} = [x\ y\ z]^T$: quadrotor position in $\mathcal{B}$;
- $\boldsymbol{V} = [V_X\ V_Y\ V_Z]^T$: quadrotor velocity in $\mathcal{W}$;
- $\boldsymbol{\Theta} = [\phi\ \theta\ \psi]^T$: Euler angles roll, pitch, and yaw in $\mathcal{B}$, respectively;
- $\boldsymbol{\omega} = [\omega_x\ \omega_y\ \omega_z]^T$: quadrotor angular velocity in $\mathcal{B}$;
- m: quadrotor mass;
- $\boldsymbol{I} = diag(I_x, I_y, I_z)$: mass moment of inertia in $\mathcal{B}$;
- ω_{r_i}: motor speeds, $i = 1, 2, 3, 4$;
- $\Omega = -\omega_{r_1} + \omega_{r_2} - \omega_{r_3} + \omega_{r_4}$: sum of motor speeds;
- k_f, k_m: motor thrust and torque coefficients, respectively;
- c_t, c_r: translational and rotational drag coefficients, respectively;
- l: moment arm from the origin of $\mathcal{B}$ to each motor.
- $\boldsymbol{g}$: gravitational acceleration, $[0\ 0\ g]^T$ in NED frame with $g = 9.81 m/s^2$.

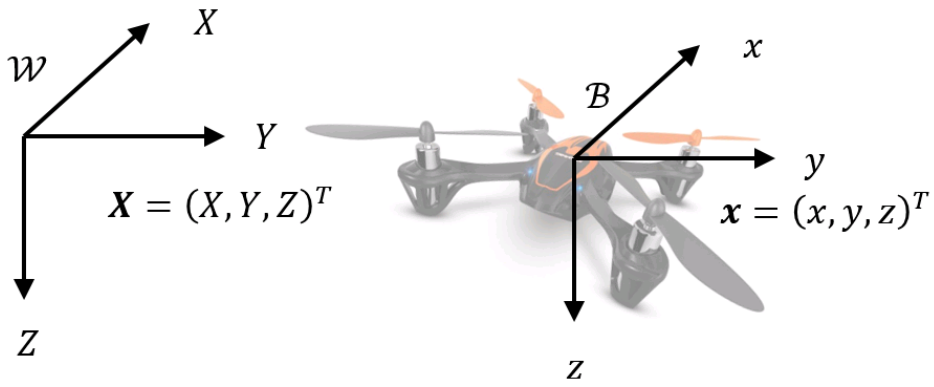


Figure 5.4: The reference frames of a quadrotor.

Assume that the motor forces are proportional to motor speed squared $\omega_{r_i}^2$, and the control inputs $\boldsymbol{U}$ satisfy equation (5.1) with constraints. Physically, the control inputs

U_1, U_2l, U_3l, U_4 represent the total thrust and the total motor torques along the roll, pitch, and yaw axes, respectively. These are the control inputs for altitude and Euler angles. The constraints in (5.1) represent the physical capacity of the motors.

$$\begin{aligned} \boldsymbol{U} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} &= \begin{bmatrix} k_f & k_f & k_f & k_f \\ 0 & k_f & 0 & -k_f \\ k_f & 0 & -k_f & 0 \\ k_m & -k_m & k_m & -k_m \end{bmatrix} \begin{bmatrix} \omega_{r_1}^2 \\ \omega_{r_2}^2 \\ \omega_{r_3}^2 \\ \omega_{r_4}^2 \end{bmatrix} \\ 4k_f\omega_{r,min}^2 < U_1 &\leq 4k_f\omega_{r,max}^2 \\ -k_f\omega_{r,max}^2 \leq U_2 &\leq k_f\omega_{r,max}^2 \\ -k_f\omega_{r,max}^2 \leq U_3 &\leq k_f\omega_{r,max}^2 \\ -2k_m\omega_{r,max}^2 \leq U_4 &\leq 2k_m\omega_{r,max}^2 \end{aligned} \tag{5.1}$$

Other forces include gravity mg, translation drag $c_t\boldsymbol{V}$, rotational drag $c_r\boldsymbol{\omega}^2$, and Coriolis forces from quadrotor body rotation and motor rotations. The state-space model could be written compactly as equation (5.2).

$$\begin{aligned} \dot{\boldsymbol{X}} &= \boldsymbol{V} \\ \dot{\boldsymbol{\Theta}} &= R_v^{-1}\boldsymbol{\omega} \\ \dot{\boldsymbol{V}} &= \boldsymbol{g} + \frac{1}{m}\left(-c_t\boldsymbol{V} - R_{\mathcal{B}\to\mathcal{W}}\begin{bmatrix} 0 \\ 0 \\ U_1 \end{bmatrix}\right) \\ \dot{\boldsymbol{\omega}} &= I^{-1}\left(\boldsymbol{\omega}\times(I\boldsymbol{\omega}) - \boldsymbol{\omega}\times\left(I_r\begin{bmatrix} 0 \\ 0 \\ \Omega \end{bmatrix}\right) - c_r\boldsymbol{\omega}^2 + \begin{bmatrix} U_2l \\ U_3l \\ U_4 \end{bmatrix}\right) \end{aligned} \tag{5.2}$$

In (5.2), $R_{\mathcal{B}\to\mathcal{W}}$ is the rotation matrix from frame $\mathcal{B}$ to $\mathcal{W}$, and R_v is a linear transformation from $\dot{\boldsymbol{\Theta}}$ to $\boldsymbol{\omega}$. Use small case s and c to represent sin and cos functions, respectively, then the matrices are given by

$$R_{\mathcal{B}\to\mathcal{W}} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}$$

$$R_v = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\phi & c\phi c\theta \end{bmatrix}$$

We only gives a brief summary of the model to introduce enough notations for controller derivations. For details, refer to an equivalent dynamic model in [14]. The expanded version of (5.2) is given by equation (5.3) for reference.

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{V}_X \\ \dot{V}_Y \\ \dot{V}_Z \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} V_X \\ V_Y \\ V_Z \\ \omega_x + \omega_y s\phi tan\theta + \omega_z c\phi tan\theta \\ \omega_y c\phi - \omega_z s\phi \\ -\frac{c_t}{m} V_X \\ -\frac{c_t}{m} V_Y \\ -\frac{c_t}{m} V_Z \\ \frac{I_y - I_z}{I_x}\omega_y\omega_z - \frac{c_r}{I_x}\omega_x^2 - \frac{I_r}{I_x}\Omega\omega_y \\ \frac{I_z - I_x}{I_y}\omega_x\omega_z - \frac{c_r}{I_y}\omega_y^2 - \frac{I_r}{I_y}\Omega\omega_x \\ \frac{I_x - I_y}{I_x}\omega_x\omega_y - \frac{c_r}{I_z}\omega_z^2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{c\phi s\theta c\psi + s\phi s\psi}{m} \\ \frac{c\phi s\theta s\psi - s\phi c\psi}{m} \\ \frac{c\phi c\theta}{m} \\ 0 \\ 0 \\ 0 \end{bmatrix} U_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{l}{I_x} \\ 0 \\ 0 \end{bmatrix} U_2 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{l}{I_y} \\ 0 \end{bmatrix} U_3 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{I_z} \end{bmatrix} U_4 \tag{5.3}$$

5.3.3 Controller Design

Given the quadrotor model in Section 5.3.2, we can follow the standard procedure of designing sliding mode controllers, and formulate four independent control laws governing the altitude and attitudes (Euler angles) [14]. However, to achieve horizontal position control, we need to derive the desired roll (ϕ_d) and pitch (θ_d) as intermediate bridges. On one hand, they serve as synthetic control inputs for horizontal motions in X and Y. On the other hand, they are also the reference signals for the actual roll (ϕ) and pitch (θ). As is mentioned before, traditional methods such as backstepping and state augmentation lead to explosion of terms. To avoid that, we apply dynamic surface control (DSC) techniques in the horizontal position controllers.

First, we define the tracking errors. We use the subscript $_d$ to represent the desired value of any variable.

$$\begin{aligned} e_X &\triangleq X - X_d \quad & e_\theta &\triangleq \theta - \theta_d \\ e_Y &\triangleq Y - Y_d \quad & e_\phi &\triangleq \phi - \phi_d \\ e_Z &\triangleq Z - Z_d \quad & e_\psi &\triangleq \psi - \psi_d \end{aligned}$$

Note that to achieve position control, we can specify the desired position (X_d, Y_d, Z_d) and desired yaw (ψ_d). However, we cannot arbitrarily specify the desired pitch (θ_d) and roll (ϕ_d) because they are coupled with the horizontal motions (X_d, Y_d).

Define the sliding surfaces to be some first-order filters of the error terms (5.4) with tuning parameters $\lambda's$ [119]. Note that the first terms in S_ϕ and S_θ are different from the rest.

$$\begin{array}{ll} S_X \triangleq \dot{e}_X + \lambda_X e_X & S_\theta \triangleq \dot{\theta} + \lambda_\theta e_\theta \\ S_Y \triangleq \dot{e}_Y + \lambda_Y e_Y & S_\phi \triangleq \dot{\phi} + \lambda_\phi e_\phi \\ S_Z \triangleq \dot{e}_Z + \lambda_Z e_Z & S_\psi \triangleq \dot{e}_\psi + \lambda_\psi e_\psi \end{array} \tag{5.4}$$

The asymptotic stability proof of a general sliding mode controller is based on Lyapunov's second method. Define a general sliding surface S in a form similar to (5.4). First, we compute $\dot{S}$, or the time derivative of S, and set it equal to $-\eta sgn(S)$. Let the positive definite Lyapunov function be $V = \frac{1}{2}S^2$, then $\dot{V} = S\dot{S} \leq -\eta|S|$ is negative definite, which guarantees $S \to 0$ asymptotically. Once we are on $S = 0$, the tracking error goes to zero exponentially [59]. Here, we simply use the design method without proofs.

5.3.3.1 Vertical and Yaw Controllers

The sliding mode controllers for altitude and yaw are straight-forward. Take the first time derivatives of the corresponding sliding surfaces in (5.4), and set them equal to the robustness terms with parameters η_Z and η_ψ, we have

$$\begin{aligned} \dot{S}_Z &= (\dot{V}_Z - \ddot{X}_d) + \lambda_Z \dot{e}_Z \triangleq -\eta_Z sgn S_Z \\ \dot{S}_\psi &\approx (\dot{\omega}_z - \ddot{\psi}_d) + \lambda_\psi \dot{e}_\psi \triangleq -\eta_\psi sgn S_\psi \end{aligned} \tag{5.5}$$

In (5.5), we approximate $\ddot{\psi}$ to be $\dot{\omega}_z$. Substitute $\dot{V}_Z$ and $\dot{\omega}_z$ from (5.3) into (5.5), we get the vertical and yaw control laws in (5.6).

$$\begin{aligned} U_1 &= \frac{m}{c\phi c\theta}\left[\left(-g + \frac{c_t}{m}V_Z + \ddot{Z}_d\right) - \lambda_Z \dot{e}_Z - \eta_Z sgn S_Z\right] \\ U_4 &= I_z\left[\left(-\frac{I_x - I_y}{I_z}\omega_x\omega_y + \frac{c_r}{I_z}\omega_z^2 + \ddot{\psi}_d\right) - \lambda_\psi \dot{e}_\psi \ - \eta_\psi sgn S_\psi\right] \end{aligned} \tag{5.6}$$

5.3.3.2 Horizontal Controllers

In this section, we derive the horizontal control laws by augmenting our system with DSC filters. The overall horizontal motion control architecture is shown in Figure 5.5. It is a 3-step process. First, we derive the synthetic controls $\bar{\theta}$ and $\bar{\phi}$ required to track X_d and Y_d via sliding mode. At this point, if we use U_3 and U_2 to control $\bar{\theta}$ and $\bar{\phi}$ directly, we will have to take time derivatives of $\bar{\theta}$ and $\bar{\phi}$ and lead to the explosion of terms. Instead, we convert $\bar{\theta}$ and $\bar{\phi}$ into θ_d and ϕ_d, respectively, using DSC filters. The derivatives of θ_d and ϕ_d are implicitly given by the filter updates and thus resolved the term explosion problem. Finally, we can use control inputs U_3 and U_2 to drive θ and ϕ to θ_d and ϕ_d, respectively, via sliding mode. In the following, we show the design process in details.

Derive synthetic inputs $\bar{\theta}$ and $\bar{\phi}$ via sliding mode First, take the time derivative of the sliding surfaces S_X and S_Y in (5.4) and set them equal to the robustness terms with parameters η_X and η_Y, respectively.

$$\begin{bmatrix} X_d \\ Y_d \end{bmatrix} \overset{SM}{\longleftarrow} \begin{bmatrix} \bar{\theta} \\ \bar{\phi} \end{bmatrix} \overset{DSC}{\longleftarrow} \begin{bmatrix} \theta_d \\ \phi_d \end{bmatrix} \overset{SM}{\longleftarrow} \begin{bmatrix} U_3 \\ U_2 \end{bmatrix}$$

Figure 5.5: The horizontal motion control architecture.

$$\begin{aligned} \dot{S}_X &= (\dot{V}_X - \ddot{X}_d) + \lambda_X \dot{e}_X \triangleq -\eta_X sgn S_X \\ \dot{S}_Y &= (\dot{V}_Y - \ddot{Y}_d) + \lambda_Y \dot{e}_Y \triangleq -\eta_Y sgn S_Y \end{aligned} \tag{5.7}$$

Substitute $\dot{V}_X$ and $\dot{V}_Y$ from (5.3) into (5.7) and simplify, we have

$$R(\psi) \begin{bmatrix} c\phi s\theta \\ s\phi \end{bmatrix} = \begin{bmatrix} \frac{m}{U_1}\left(\left(\frac{c_t}{m}V_X + \ddot{X}_d\right) - \lambda_X \dot{e}_X - \eta_X sgn S_X\right) \\ \frac{m}{U_1}\left(\left(\frac{c_t}{m}V_Y + \ddot{Y}_d\right) - \lambda_Y \dot{e}_Y - \eta_Y sgn S_Y\right) \end{bmatrix} \tag{5.8}$$

where the matrix $R(\psi)$ is

$$R(\psi) \triangleq \begin{bmatrix} c\psi & s\psi \\ s\psi & -c\psi \end{bmatrix}$$

Note that the roll ϕ and pitch θ on the left-hand side of equation (5.8) actually represent the synthetic controls for the desired horizontal motions. We denote these two synthetic controls as $\bar{\theta}$ and $\bar{\phi}$ to distinguish them from the actual pitch θ and roll ϕ. We can further simplify (5.8) with small angle approximations: $c\phi s\theta \approx \theta$ and $s\phi \approx \phi$, yielding sliding mode control laws (5.9).

$$\begin{bmatrix} \bar{\theta} \\ \bar{\phi} \end{bmatrix} = R(\psi) \begin{bmatrix} \frac{m}{U_1}\left(\left(\frac{c_t}{m}V_X + \ddot{X}_d\right) - \lambda_X \dot{e}_X - \eta_X sgn S_X\right) \\ \frac{m}{U_1}\left(\left(\frac{c_t}{m}V_Y + \ddot{Y}_d\right) - \lambda_Y \dot{e}_Y - \eta_Y sgn S_Y\right) \end{bmatrix} \tag{5.9}$$

Augment system with DSC filters To proceed, we need to somehow use equation (5.9) in the time derivative of S_θ and S_ϕ, so that the controls U_3 and U_2 can appear. However, as was mentioned in Section 5.3.1, it requires taking time derivatives of (5.9), which leads to explosion of terms. To avoid that, we instead augment the dynamic system (5.2) with two first-order DSC filters shown in (5.10), to indirectly converge the desired pitch θ_d and roll ϕ_d to the synthetic controls $\bar{\theta}$ and $\bar{\phi}$, respectively [123]. The time constants τ_θ and τ_ϕ determine the convergence rates. The initial conditions are set to zeros in both filters.

$$\begin{aligned} \tau_\theta \dot{\theta}_d + \theta_d &= \bar{\theta}, \quad \theta_d(0) = 0 \\ \tau_\phi \dot{\phi}_d + \phi_d &= \bar{\phi}, \quad \phi_d(0) = 0 \end{aligned} \tag{5.10}$$

The DSC filters (5.10) allows us to compute $\dot{\theta}_d$ and $\dot{\phi}_d$ required in $\dot{S}_\theta$ and $\dot{S}_\phi$, without taking time derivatives of $\bar{\theta}$ and roll $\bar{\phi}$. Therefore, no simplifications in the dynamics are

needed to accommodate the explosion of terms. A semi-global stability proof via Lyapunov analysis is lengthy because it involves the error dynamics $(\dot{\theta}_d - \dot{\bar{\theta}})$ and $(\dot{\phi}_d - \dot{\bar{\phi}})$, which again involves explosion of terms. A standard proof is provided in [123]. Here, we simply adopt the results.

Derive control inputs U_3 and U_2 via sliding mode Now we can take the time derivatives of S_θ and S_ϕ, and set them equal to their robustness terms with parameters η_θ and η_ϕ, respectively, with $\ddot{\theta} \approx \dot{\omega}_y$ and $\ddot{\phi} \approx \dot{\omega}_x$.

$$\begin{aligned} \dot{S}_\theta &\approx \dot{\omega}_y + \lambda_\theta(\dot{\theta} - \dot{\theta}_d) \triangleq -\eta_\theta sgn S_\theta \\ \dot{S}_\phi &\approx \dot{\omega}_x + \lambda_\phi(\dot{\phi} - \dot{\phi}_d) \triangleq -\eta_\phi sgn S_\phi \end{aligned} \tag{5.11}$$

In equation (5.11), $\dot{\omega}_y$ and $\dot{\omega}_x$ include the control inputs U_3 and U_2. They ($\dot{\omega}_y$ and $\dot{\omega}_x$), as well as $\dot{\theta}$ and $\dot{\phi}$, are given by (5.3), and $\dot{\theta}_d$ and $\dot{\phi}_d$ are given by re-expressing the DSC filters as (5.12).

$$\begin{aligned} \dot{\theta}_d &= \frac{1}{\tau_\theta}\left(\bar{\theta} - \theta_d\right) \\ \dot{\phi}_d &= \frac{1}{\tau_\phi}\left(\bar{\phi} - \phi_d\right) \end{aligned} \tag{5.12}$$

In equation (5.12), $\bar{\theta}$ and $\bar{\phi}$ are given by (5.9), and θ_d and ϕ_d are given by the DSC filter updates in (5.10). Then, the horizontal control laws are given by

$$\begin{aligned} U_3 &= \frac{I_y}{l}\left[\left(-\frac{I_z - I_x}{I_y}\omega_x\omega_z + \frac{c_r}{I_y}\omega_y^2 + \frac{I_r}{I_y}\Omega\omega_x\right) - \lambda_\theta \dot{e}_\theta - \eta_\theta sgn S_\theta\right] \\ U_2 &= \frac{I_x}{l}\left[\left(-\frac{I_y - I_z}{I_x}\omega_y\omega_z + \frac{c_r}{I_x}\omega_x^2 + \frac{I_r}{I_x}\Omega\omega_y\right) - \lambda_\phi \dot{e}_\phi - \eta_\phi sgn S_\phi\right] \end{aligned} \tag{5.13}$$

Note that the terms in (5.6), (5.9) and (5.13) could be categorized into three parts. Take U_1 as an example. The first part $\left(-g + \frac{c_t}{m}V_Z + \ddot{Z}_d\right)$ cancels the original (possibly nonlinear) dynamics; the second part $(-\lambda_Z \dot{e}_Z)$ is the desired linear dynamics; the third part $(-\eta_Z sgn S_Z)$ is an uncertainty robustness term.

Lastly, the signum function ($sgn(\cdot)$) should be replaced by a smoothed function to avoid chattering. In this paper, we use a function with logistic and linear smoothing (5.14).

$$\eta sgn(S) \leftarrow \left[\eta_1\left(\frac{2}{1 + e^{-cS}} - 1\right) + \eta_2 S\right] \tag{5.14}$$

Other than small angle approximations, we did not make any simplification to the full dynamic model (5.2) throughout the derivations. The innovation lies in the adoption of DSC filters.

5.4 High-Level MPC Controller

This section focuses on the high-level trajectory following controller (Section 5.2.2). A review on the current path following literature is presented in Section 5.4.1. And the controller design is in Section 5.4.4.

5.4.1 Literature Review

Path following is a basic requirement for any type of vehicle. A survey of the UAV path following literature is presented in [125]. It categorizes the algorithms into geometric methods and control techniques. They outputs of the algorithms is a desired acceleration vector, including magnitude and heading.

Pure pursuit [23] and line-of-sight (LOS) [95] guidance laws, is a simple geometric method. The algorithm uses a virtual target point (VTP) on the path. The control law drives the vehicle to chase the VTP, which eventually leads the vehicle onto the path. The stability of LOS guidance laws depends heavily on the selection of the virtual distance parameter, the distance between the VTP and the UAS position projected on the path. However, pure pursuit is not robust and could give large cross-track error under wind disturbance.

Another geometric approach is based on vector fields (VF) [86]. The main idea is that vector field indicate the direction of flow. VFs operate in two modes. When the vehicle is far away from the path, it gives a constant entry heading command. When the vehicle enters a transition zone close to the path, the heading command converges to the path. The stability of the algorithm is shown by Lyapunov stability arguments in [86]. Vector field is known to give low cross-track error, but it has more tuning parameters and could be subjected to chattering.

Geometric methods can track the path well, but they usually has no mechanisms to track speeds. On the other hand, control techniques are another popular type of path following algorithms. They are often capable of tracking speeds at different locations, as well as providing a certain degree of robustness to wind disturbances. A PID controller with feed-forward capability was presented in [103] to give high lateral tracking performance. A well-known linear control technique is linear quadratic regulator (LQR). In [101], the author solves a LQR problem with distance and speed error dynamics to generate an optimal lateral acceleration command. Model predictive control (MPC) [53] uses a similar formulation but with a finite time horizon. Other techniques include slidng mode control [42], feedback linearization [107], backstepping [4], and dynamic programming [118]. In [107], an input-output feedback linearization controller is designed to address the path following problem defined by path, velocity, and yaw specifications. Lastly, a 3D path following algorithm is presented in [22] to manage a fleet of UASs. The error dynamics is defined by position error, heading error, rotational matrix error, and angular velocity error. The time coordination problem is addressed by introducing coordination variables.

The goal of this section is not to invent a new controller, but rather show that the trajectories generated in Chapter 4 can be followed reasonably well by a quadrotor, when

appropriate speed limits are imposed along the sharp turns. Therefore, we would like to not only follow the path, but also track the speeds associated with each waypoint. Geometric approaches cannot achieve the later goal explicitly. Therefore, we choose a simple MPC controller to perform 4D trajectory following.

5.4.2 Dynamic Model

The MPC controller uses the following 4-state dynamic model in equation (5.15). The model simply composes of two double integrators reprenting linear motions in X and Y axes. There are two additional viscous damping terms modeling aerodynamic drag in the acceleration.

$$\frac{d}{dt}\begin{bmatrix} X(t) \\ Y(t) \\ V_X(t) \\ V_Y(t) \end{bmatrix} = \begin{bmatrix} V_X(t) \\ V_Y(t) \\ a_X(t) - \frac{c_t}{m}V_X(t) \\ a_Y(t) - \frac{c_t}{m}V_Y(t) \end{bmatrix} \tag{5.15}$$

In state-space form, it can be re-written equivalently as equation (5.16). Obviously, it is a linear model. And we assume to have full access to the states, i.e., the state $\boldsymbol{x}(t)$ is available at the current time t. This assumption is realistic because position and velocity information is generally estimated well in UAS autopilots.

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= A_c\boldsymbol{x}(t) + B_c\boldsymbol{u}(t) \\ \boldsymbol{x}(0) &= \boldsymbol{x}_0 \end{aligned} \tag{5.16}$$

where

$$A_c = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -c_t/m & 0 \\ 0 & 0 & 0 & -c_t/m \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$\boldsymbol{x} = [X \;\; Y \;\; V_X \;\; V_Y]^T, \quad \boldsymbol{u} = [a_X \;\; a_Y]^T$$

To use MPC, we often discretize the continuous model using a first-order Euler expansion with a sampling time Δt. The final form of the state-space model is shown in equation (5.17). The four states are linear positions $X(k)$, $Y(k)$, and linear velocity $V_X(k)$, $V_Y(k)$. The control variables are the horizontal acceleration $a_X(k)$, $a_Y(k)$, which can easily be converted to a magnitude and a heading angle.

$$\boldsymbol{x}(k+1) = A\boldsymbol{x}(k) + B\boldsymbol{u}(k) \tag{5.17}$$

where

$$\begin{aligned} A &= I_4 + A_c\Delta t \\ B &= B_c dt \\ \boldsymbol{x}(k) &= [X(k) \;\; Y(k) \;\; V_X(k) \;\; V_Y(k)]^T \\ \boldsymbol{u}(k) &= [a_X(k) \;\; a_Y(k)]^T \end{aligned} \tag{5.18}$$

5.4.3 Horizontal Control Conversion

To convert the horizontal kinematic control $\boldsymbol{u}$ to horizontal control inputs U_2 and U_3, we assume that the quadrotor is in vertical equilibrium. Then the inputs that could affect the horizontal motions are the desired roll (ϕ_d), pitch (θ_d), and yaw (ψ_d). It turns out that yaw is not effective, so we choose roll and pitch. The conversion is straight forward. First, convert the acceleration command $[a_X \;\; a_Y]^T$ from the inertial frame to the quadrotor body frame $\mathcal{B}$, and then convert it to the desired pitch (θ_d) and roll (ϕ_d). Equation (5.19) summarizes the process.

$$\begin{aligned} a_x &= a_X \cos\psi + a_Y \sin\psi \\ a_y &= -a_X \sin\psi + a_Y \cos\psi \\ \theta_d &= -tan^{-1}(a_x/g) \\ \phi_d &= tan^{-1}(a_y/g) \end{aligned} \tag{5.19}$$

5.4.4 Controller Design

With the model in equation (5.17), we can write down the MPC formulation. Define the tracking error $\tilde{\boldsymbol{x}}(k)$ in equation (5.20).

$$\tilde{\boldsymbol{x}}(k) = \boldsymbol{x}_d(k) - \boldsymbol{x}(k) \tag{5.20}$$

The current state $\boldsymbol{x}(k)$ is clearly defined in equation (5.18). But the desired state $\boldsymbol{x}_d(k)$ is defined differently in different papers. In this dissertation, we adopt the following definition of $\boldsymbol{x}_d(k)$ in Figure 5.6. Each waypoint has a 3D position, a speed, and a direction implicitly defined along the path. First, we fit the set of waypoints within the current time horizon to a cubic spline with continuous slope and curvature. Then, the closest point $[X_d(k) \; Y_d(k)]$, indicated by the red dot, on the path is found by drawing a perpendicular line from the current position $[X(k) \; Y(k)]$. The velocity is $[V_{X,d}(k) \; V_{Y,d}(k)]$, indicated by the red arrow, is found by the speed command and the derivative expression of the spline.

To avoid large tracking error due to excessive speed, the speed command $V_d = ||[V_{X,d};\; V_{Y,d}]||$ is thresholded by a maximum speed as a function of the instantaneous radius $R(\boldsymbol{x}(k))$.

$$V_d(R(\boldsymbol{x}(k))) \leq \sqrt{a_{max} R(\boldsymbol{x}(k))} \tag{5.21}$$

The optimal control problem can then be written as equation (5.22) for a finite time horizon with N steps. The cost function consists of quadratic stage costs for states and control from time steps 0 to $N-1$ and a terminal state cost at time step N. There are several constraints. The dynamic constraint includes both the dynamic equation and initial condition. To limit the jerk, we impose a maximum variation of $j_{max}\Delta t$ between two consecutive controls. And the state $\boldsymbol{x} \in \mathcal{X}$ where $\mathcal{X}$ is a feasible set, and the control $\boldsymbol{u} \in \mathcal{U}$ also has to be feasible.

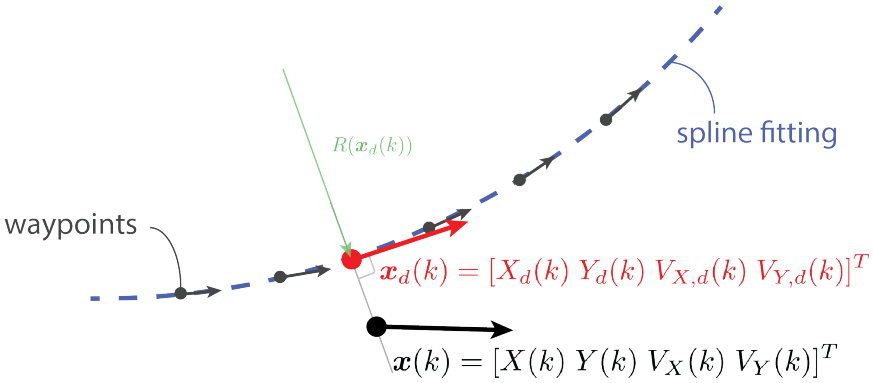


Figure 5.6: Definition of the desired state $\boldsymbol{x}_d(k)$.

$$\begin{aligned} \underset{\boldsymbol{u}(0),\ldots,\boldsymbol{u}(N-1)}{min} \quad & \left\{ \sum_{k=0}^{N-1} \left[\tilde{\boldsymbol{x}}(k)^T Q \tilde{\boldsymbol{x}}(k) + \boldsymbol{u}(k)^T R \boldsymbol{u}(k) \right] + \tilde{\boldsymbol{x}}(N)^T S \tilde{\boldsymbol{x}}(N) \right\} \\ s.t: \quad & \boldsymbol{x}(k+1) = A\boldsymbol{x}(k) + B\boldsymbol{u}(k), \qquad k = 0, \ldots, N-1 \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ & ||\boldsymbol{u}(k+1) - \boldsymbol{u}(k)|| \leq j_{max} \Delta t, \qquad k = 0, \ldots, N-1 \\ & \boldsymbol{x}(k+1) \in \mathcal{X}, \qquad k = 0, \ldots, N-1 \\ & \boldsymbol{u}(k) \in \mathcal{U}, \qquad k = 0, \ldots, N-1 \end{aligned} \tag{5.22}$$

The feasible set of states $\mathcal{X}$ are bounded by a maximum speed V_{max}. The feasible set of controls $\mathcal{U}$ are bounded by a maximum acceleration u_{max}. Note that these constraints form second-order cones, so the optimization problem retains convexity.

$$\begin{aligned} \mathcal{X} &= \{\boldsymbol{x} \in \mathbb{R}^4 \,|\, \boldsymbol{x}(3)^2 + \boldsymbol{x}(4)^2 \leq V_{max}^2\} \\ \mathcal{U} &= \{\boldsymbol{u} \in \mathbb{R}^2 \,|\, ||\boldsymbol{u}|| \leq u_{max}\} \end{aligned} \tag{5.23}$$

The first control action $u^*(0)$ is applied after the optimal control problem is solved. The other controls are discarded. The process is repeated in the next time step. This type of control strategy is called receding horizon control (RHC). To make sure that the RHC is persistently feasible for all choices of cost functions, we also compute the control invariant terminal set $\mathcal{X}_f$ at time horizon N in every time step.

5.5 Simulations

In this section, we first examine the performance of the low-level controllers, then evaluate the combined controller to see how close it can track different flight trajectories.

5.5.1 Low-Level Controllers

In this section, we present the simulation results of the sliding mode controllers. First, we examine the behavior of the sliding mode controllers with DSC filters defined in equation (5.10). Second, the robustness of the sliding mode controllers is evaluated with respect to disturbance, model uncertainty, and measurement noise. All simulations are performed in MATLAB/SIMULINK R2015a [76] with a fixed-time-step solver $dt = 0.02sec$.

The quadrotor parameters are referenced from [73] with a different rotational inertia matrix, so that the model is more realistic.

$$\begin{aligned} m &= 2.0\ kg & \boldsymbol{g} &= [0\ 0\ 9.81]^T\ m/s^2 \\ k_m &= 10^{-6}\ Nm \cdot s^2 & k_f &= 10^{-5}\ N \cdot s^2 \\ c_t &= 10^{-2}\ Ns/m & c_r &= 10^{-2}\ Nm \cdot s^2 \\ l &= 0.2m & I &= 1.2416\ diag([1\ 1\ 2]e\text{-}2)\ kg\ m^2 \end{aligned} \tag{5.24}$$

The reference trajectories in altitude, yaw, and horizontal positions are either sinusoids or zeros. The sinusoidal references are taken from [50] in equation (5.25). The exponential terms are introduced to produce smooth initial references.

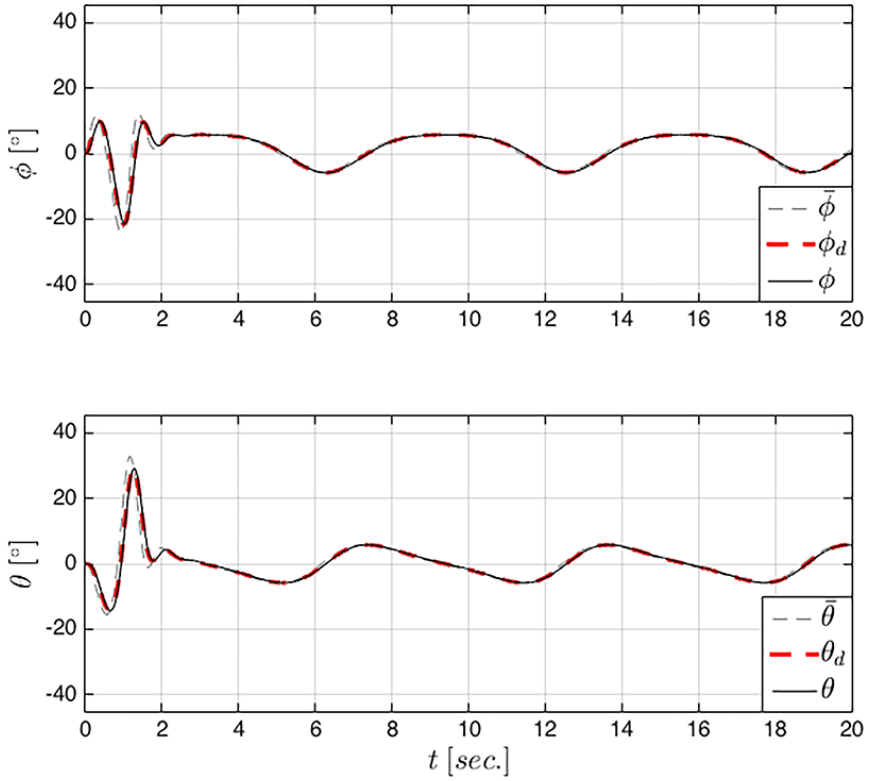
$$\begin{aligned} X_d(t) &= (1 - e^{-t^3})sint\ m \\ Y_d(t) &= (1 - e^{-t^3})cost\ m \\ Z_d(t) &= -(1 - e^{-t^3})\ m \\ \psi_d(t) &= 30(1 - e^{-t^3})sint\ ^\circ \end{aligned} \tag{5.25}$$

The controller gains are listed below in (5.26). There are two sets of $\eta's$ as defined in (5.14) for each of the six control laws (5.6), (5.9), and (5.13). The two time constants τ_ϕ and τ_θ are chosen to minimize filter lags while preserving system stability.

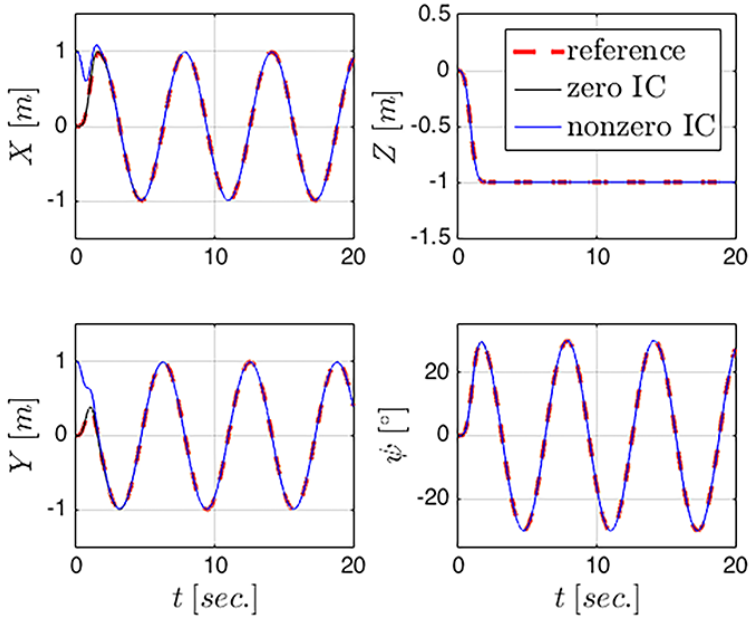
$$\begin{array}{lll} \eta_{1X} = 3 & \eta_{1Y} = 3 & \eta_{1Z} = 4 \\ \eta_{1\phi} = 20 & \eta_{1\theta} = 20 & \eta_{1\psi} = 10 \\ \eta_{2X} = 1 & \eta_{2Y} = 1 & \eta_{2Z} = 5 \\ \eta_{2\phi} = 50 & \eta_{2\theta} = 50 & \eta_{2\psi} = 10 \\ \lambda_X = 2 & \lambda_Y = 2 & \lambda_Z = 3 \\ \lambda_\phi = 50 & \lambda_\theta = 50 & \lambda_\psi = 10 \\ c_X = 5 & c_Y = 5 & c_Z = 50 \\ c_\phi = 80 & c_\theta = 80 & c_\psi = 10 \\ \tau_\phi = 0.1 & \tau_\theta = 0.1 & \end{array} \tag{5.26}$$

5.5.1.1 Ideal Behavior

Figure 5.7 shows the tracking performance of both the DSC filters and sliding mode controllers without any model uncertainties or disturbances. In Figure 5.7a, the gray dashed



(a) DSC filters make ϕ_d and θ_d (dashed red lines) follow $\bar{\phi}$ and $\bar{\theta}$ (dashed gray lines) with small lags, respectively. Then, sliding mode controllers in (5.13) drive θ and ϕ (solid black lines) to θ_d and ϕ_d, respectively.



(b) tracking reference signals with zero (solid black lines) and nonzero (solid blue lines) initial conditions (IC).

Figure 5.7: Tracking performance without disturbance.

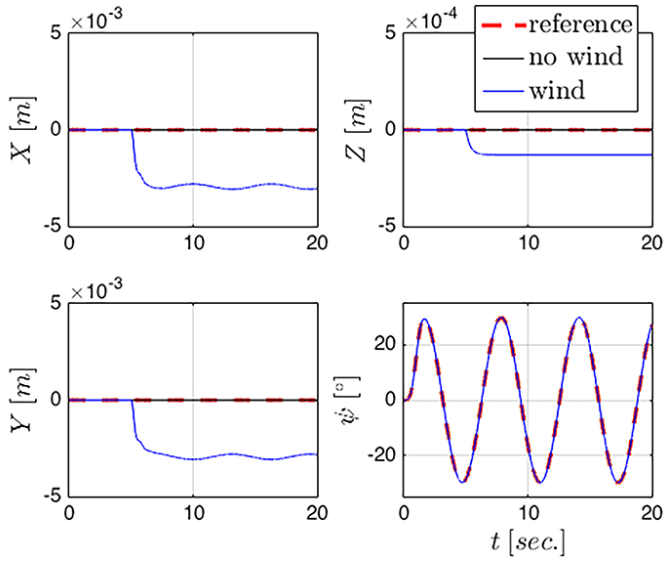


Figure 5.8: Constant wind disturbance

lines indicate the synthetic controls $\bar{\phi}$ and $\bar{\theta}$ required to drive the quadrotor to a desired horizontal position (X_d, Y_d) (equation (5.9)). To use sliding mode control, we need the derivatives of $\bar{\phi}$ and $\bar{\theta}$. However, if their derivatives were computed analytically from equation (5.9), we would have explosion of terms. Instead, by introducing the DSC filters, the reference synthetic controls ϕ_d and θ_d in dashed red lines (equation (5.10)) are able to track $\bar{\phi}$ and $\bar{\theta}$ closely with only small lags. As a result, we can compute their derivatives from equation (5.12) analytically. With ϕ_d, θ_d, and their derivatives, the sliding mode controllers in (5.13) can drive the quadrotor roll (ϕ) and pitch (θ) to θ_d and ϕ_d, respectively. Therefore, horizontal position tracking is achieved.

Figure 5.7b shows the tracking results of a perfectly known quadrotor model, with respect to zero or nonzero initial conditions (IC). In the case with zero IC (black), we have almost perfect tracking for all references. In the case with nonzero horizontal IC, the convergence in X and Y is also fast, both within $3sec$.

5.5.1.2 Robustness

Disturbance: The disturbance model is an external wind field $\boldsymbol{V_{wind}} = [10\ 10\ 10]m/s$ similar to [75], applied right after time $t = 5sec$. To examine the disturbance effect, we used zero references for X, Y, and Z, instead. Figure 5.8 shows that the horizontal and vertical tracking are affected by about $10^{-3}m$ and $10^{-4}m$, respectively. The tracking errors could not be eliminated because we smoothed the signum functions to avoid chattering. The results are acceptable because the errors are so tiny and would not be noticeable.

Model uncertainty: We examined the robustness of model uncertainty by varying mass m and rotational inertia RI [14]. We first considered an uncertainty of 20% on mass and 10%

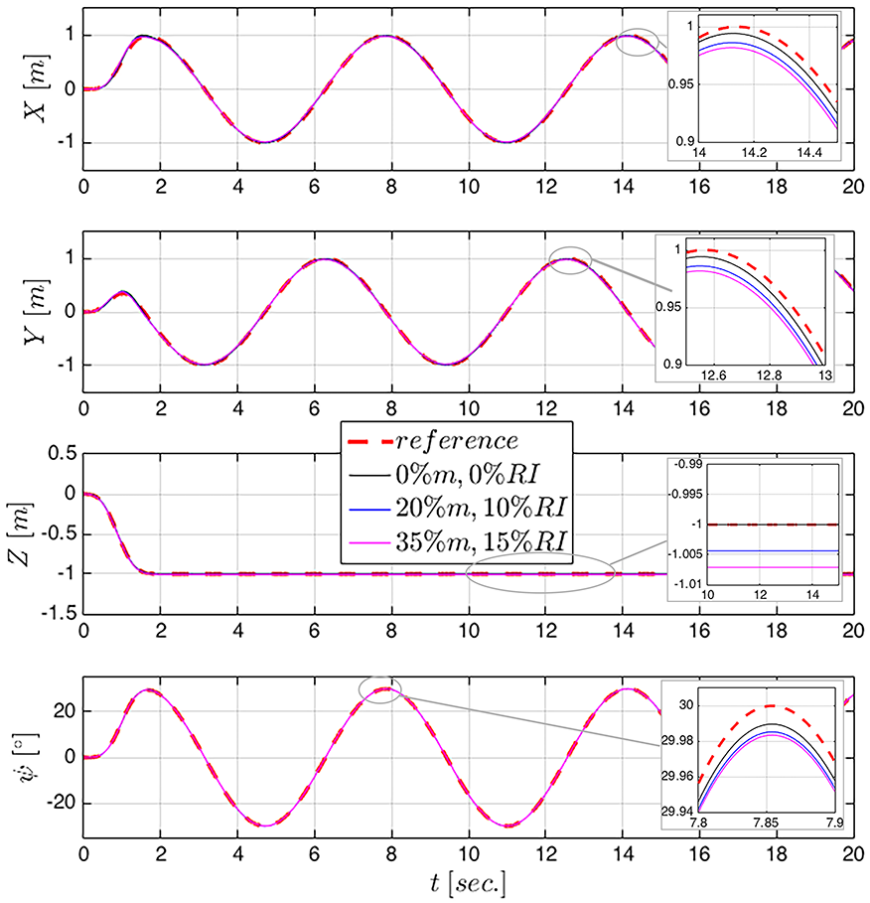


Figure 5.9: Mass and rotational inertia (RI) uncertainty

on RI, and second an uncertainty of 35% on mass and 15% on RI. The tracking references are given by (5.25). Figure 5.9 shows the simulation results. The difference in performance could not be observed in the overall tracking plots. When zoomed in (gray boxes), we can see that the altitude tracking is affected by $0.005m$ to $0.010m$ due to mass uncertainty, and the horizontal tracking is affected by about $0.01m$ due to both mass and RI uncertainty. Again, the additional tracking errors due to mass and RI are quite small. The controllers are robust to mass and RI uncertainty.

Measurement noise: Lastly, we examined to robustness to measurement noise by inserting white Gaussian noise (5.27) (in SI units) [15] into the 12 states given by (5.2). The MATLAB command $randn(12, 1)$ generates a noise vector of 12 elements with mean 0 and standard deviation (SD) 1. Then, the noise vector is vector multiplied by the desired SD's. The references are again given by (5.25).

$$\begin{aligned} Noise = randn(12,1)([10;\ 10;\ 10;\ 1;\ 1;\ 1; \\ 10;\ 10;\ 10;\ 1;\ 1;\ 1]10^{-2}) \end{aligned} \tag{5.27}$$

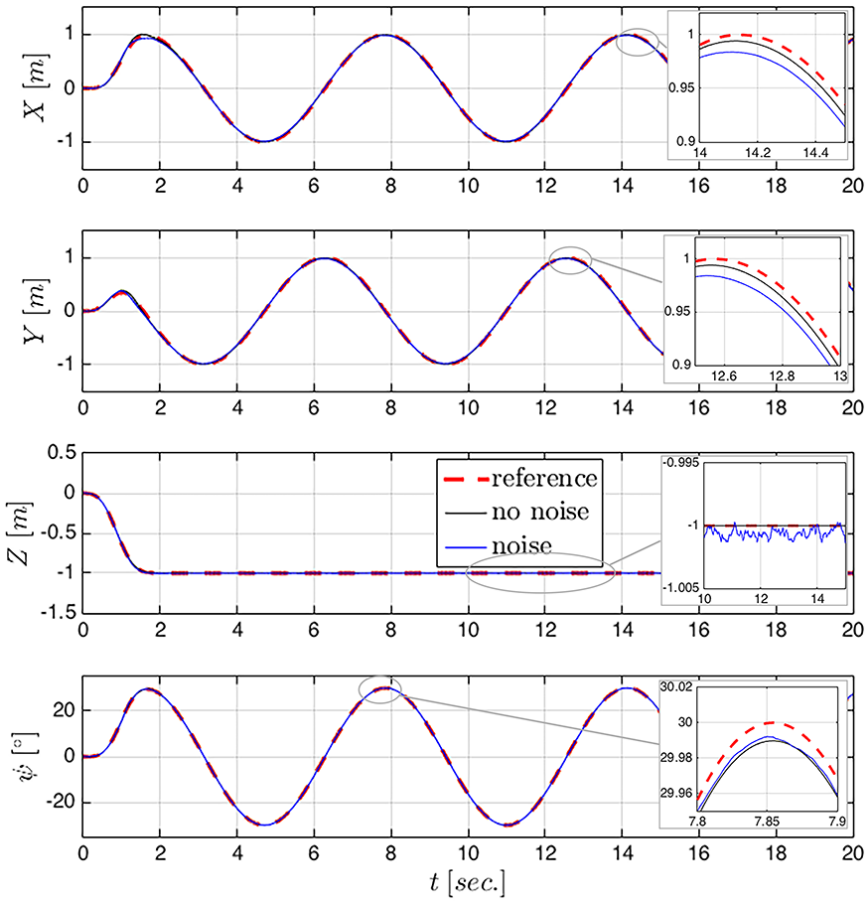


Figure 5.10: Measurement noise

The simulation results are shown in Figure 5.10. The tracking of the first 2 to 3sec are affected slightly by noise. The zoomed in view in Z shows the noisy trajectory with an average error of about $0.001m$ (blue line). Yaw tracking becomes a little noisy but is essentially unchanged. Interestingly, observe that the X and Y tracking trajectories are still smooth under noisy measurements. It is due to the smoothing effect of the DSC filters, which could be observed in Figure 5.11. The controllers are robust to measurement noise.

In summary, by introducing DSC filters, we overcome the term explosion problem in sliding mode control of an underactuated quadrotor. This technique is also applicable to other nonlinear control methods, i.e. feedback linearization. The filters introduce small lags but facilitate us to obtain time derivatives of complex intermediate signals. The controllers were first simulated under unmatched initial conditions in a perfect model. Simulation indicated fast convergence to all reference signals. The robustness of the controllers were tested under wind disturbance, mass uncertainty, and measurement noise. The controllers performs very well in all tests. In the future, we would like to make the controllers adaptive to model uncertainty and varying wind disturbance.

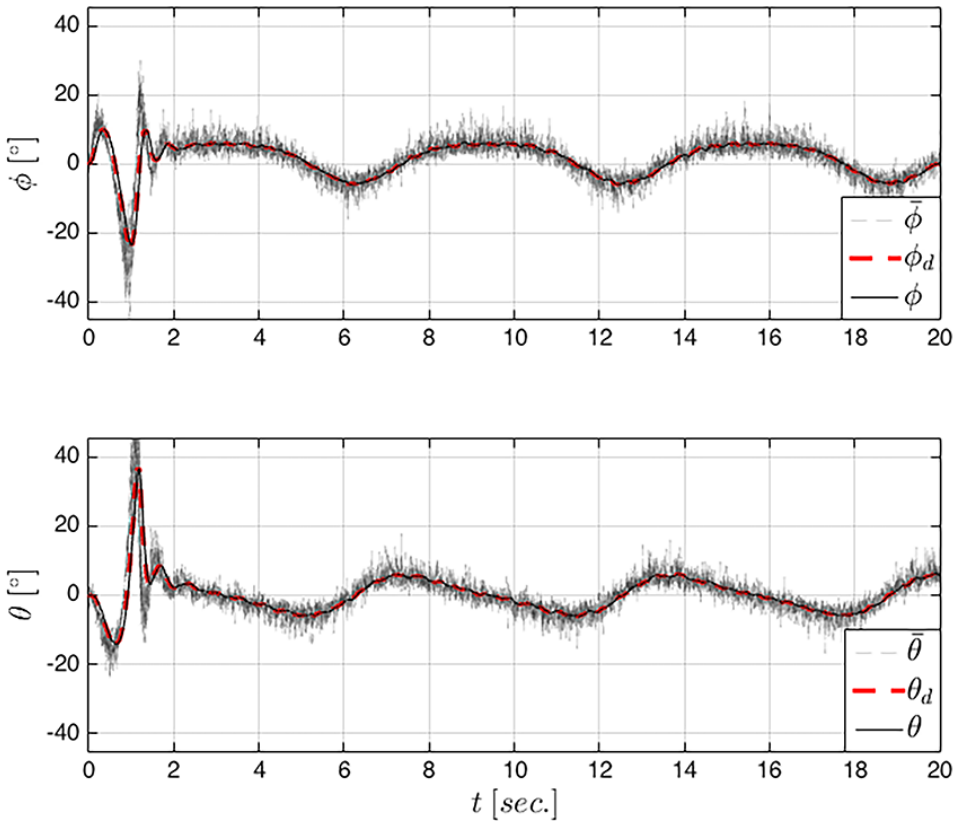


Figure 5.11: Synthetic controls with DSC filters under noisy measurements.

5.5.2 Overall Controller

A trajectory following example is simulated using the controller defined in Figure 5.2. The low-level sliding controller parameters are in equation (5.26), and the high-level MPC controller parameters are in equation (5.28). The reference trajectory is taken from Section 4.6.1 with a constant speed command $v_{des} = 20m/s$ and no wind. The maximum available horizontal acceleration is a_{max}, and the maximum centripetal acceleration is $a_{max,turn}$.

$$\begin{array}{ll} a_{max} = 10m/s^2 & N = 20 \\ a_{max,turn} = 3m/s^2 & \Delta t = 0.05s \\ j_{max} = 10m/s^3 & \end{array} \tag{5.28}$$

The tracking result is shown in Figure 5.13. We compared the tracking errors at certain locations along the path in Table 5.1 with another geometric path following approach called vector field (VF). The VF approach is reviewed in Section 5.4.1. Note that the VF approach is not capable of tracking speed commands. However, among the other geometric methods, it is known to give lower cross-track errors but tend to give oscillatory paths. Figure 5.13a shows a top view and elevation view to show the horizontal cross track errors and vertical tracking errors. Figure 5.13b is a perspective view with all paths inside the elevated terrain surface

(Section 4.6.1). Together they provide a qualitative comparison of the two approaches. In the following, we only focus on the horizontal cross-track errors.

Table 5.1: Cross-track error comparison between MPC and VF.

Point Coord. [m]	MPC [m]	VF [m]
$(-69.0, -61.2)$	1.4	2.0
$(194.0, 238.4)$	2.4	11.8
$(341.0, 279.4)$	0.6	9.3
$(382.0, 318.7)$	1.4	9.9

The MPC tracking errors is quite consistent, ranging from $0.58m$ to $2.4m$, regardless of the path curvature. In contrast, the errors in VF is much worse at high curvature locations, ranging from $2.0m$ to $11.8m$ in this example. The tracking errors in MPC is significant lower than VF at locations with large curvature. There are two reasons for the better tracking result. First, MPC can track the smaller speed command at large curvature locations, and thus requires less centripetal acceleration to make turns. The MPC speed tracking time history is shown in Figure 5.12. In general, the speed command is a constant at $20m/s$, but it is reduced at large curvature locations highlighted in green. Second, MPC naturally provides a feed-forward control whereas VF does not. Consequently, the feedback control element of MPC requires less robustness to uncertainty and therefore tracks the desired trajectory with less error. In conclusion, under perfect modeling without wind disturbance, MPC is capable of tracking most of the 4D trajectories generated from Chapter 4.

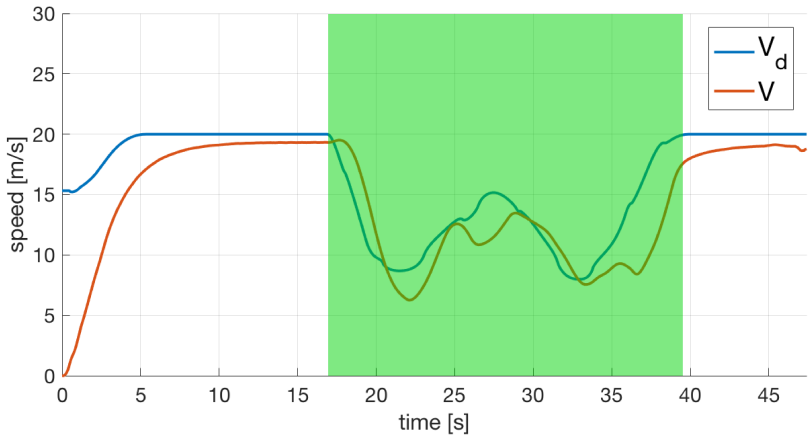
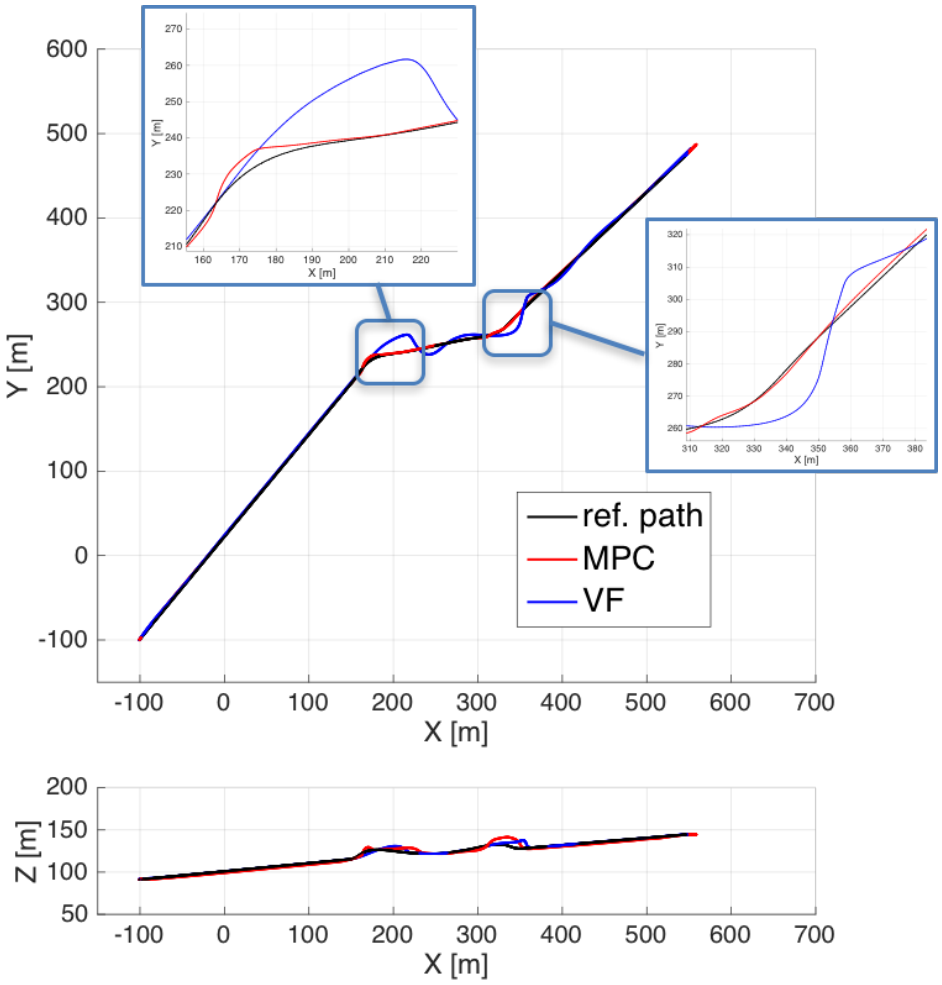
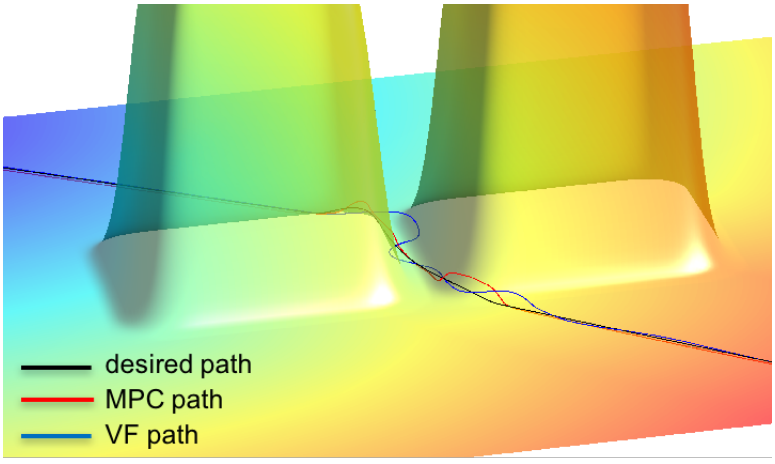


Figure 5.12: MPC speed tracking time history. V_d is desired speed, and V is the actual speed achieved by the UAS. The green region indicate locations with high curvatures.

We also tried the simulation with wind disturbance. The wind is modeled as a constant force blowing in a direction perpendicular to the path. Case *Wind 1* and *Wind 2* represent a



(a) 2D view.



(b) 3D view overlaid with elevated DEM surface.

Figure 5.13: Tracking performance without wind disturbance.

mild wind and a strong wind condition, respectively. The lateral forces exerted in case *Wind 1* and *Wind 2* are equivalent to $0.3mg$ and $0.7mg$, respectively. The upward vertical forces reduces the available lateral acceleration due to the vertical force equilibrium constraint.

Figure 5.14 shows the qualitative tracking comparison for the same scenario. The black solid line indicate the reference path. The other solid lines with different colors are MPC trajectories, while the dashed lines are VF paths. At location (1), the errors in MPC is again significantly lower that VF's. Table 5.2 shows the largest cross-track errors at the two high curvature locations. Both methods give significant errors in Location (1) compared to Location (2). In this case, MPC gives significantly lower errors under mild wind condition, $6.3m$ compared to $22.5m$ in the VF path. Such errors are comparable to GPS errors ($\sim 5m$).

However, in the extreme case *Wind 2*, both methods give errors larger than $10m$. It makes sense because the UAS has to spend a lot of energy to even maintain hovering. Therefore, large errors are expected regardless of the control methods. In reality, such windy conditions should be avoided if possible. Indeed, the trajectories generated from Chapter 4 tend to avoid such high wind areas to minimize the energy consumption. In this perspective, the flight planning system also help stabilize the UAS by avoiding high wind.

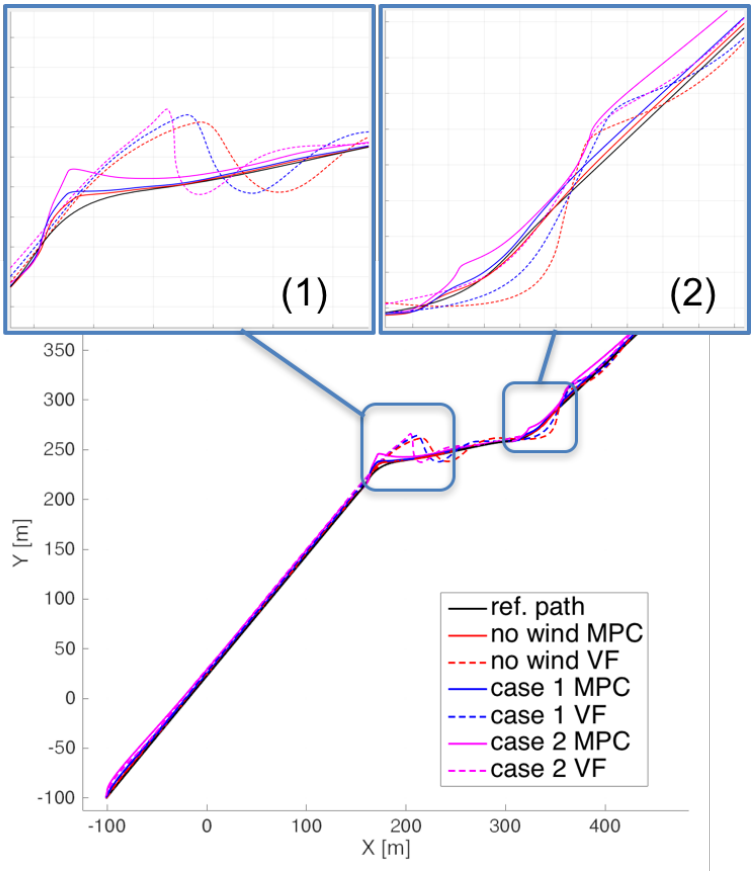


Figure 5.14: Tracking performance with wind disturbance.

Table 5.2: Cross-track error comparison between MPC and VF.

Location Cases	Wind Force $[N]$	(1)		(2)	
		MPC $[m]$	VF $[m]$	MPC $[m]$	VF $[m]$
No wind	$(0, 0, 0)$	2.4	11.8	1.4	9.9
Wind 1	$(-2, 2, 0, 2)$	6.3	22.5	1.6	6.0
Wind 2	$(-5, 5, 0, 5)$	13.5	25.6	6.2	11.1

Chapter 6

Collision Avoidance with Manned Aircraft

6.1 Introduction

This chapter is an exploration of the drone collision avoidance problem in urban areas. The main contribution is a safety control framework that enables a UAS to perform collision avoidance with manned aircraft during autonomous navigation. To formulate a meaningful collision avoidance problem, we first identify the manned and unmanned aircraft of interest. Airspace management proposals from the FAA [92] and corporations such as Google [39] and Amazon [28, 102] envisage drones flying below $500ft$, or $150m$ in class G airspace, i.e., below all the aircraft carrying people. At this altitude, emergency flights involving news, police, and EMS helicopters occupy a majority of the possible flights [39]. On the other hand, UAS flying in urban areas need vertical take-off, landing, hovering ability, and omni-directional maneuverability. This makes most urban UAS multirotors, and quadrotors are the simplest type of multirotors. Hence we focus on collision avoidance between quadrotors and helicopters.

For safety critical type of application, it is important to capture the worst-case scenario. Although airspace design can efficiently separate helicopters from drones, the separation is not 100% guaranteed in urgent scenarios such as helicopter emergency flights, which accounts for a majority of low-altitude helicopter missions. In addition, it is not safe to assume exclusive avoidance responsibility to human pilots because pilot errors account for 85% of crashes in general aviation [69]. To ensure safety, the quadrotor must take avoidance actions in the last minute, before a collision becomes unavoidable.

In general, sense-and-avoid (SAA) technology could be divided into two categories, namely collaborative and non-collaborative [28]. Non-collaborative SAA relies on remote sensing technologies to detect obstacles. Aircraft with these sensors could detect a wide range of obstacles. However, for small UAS (sUAS) less than $50lb$ [92], we have limited payload capacity and detection range. Some state-of-the-art sensors with direct distance measurements

are small radars [7], lidars [46], and 3D cameras [98], with ranges of $400m$, $60m$, and $30m$, respectively. Significant UAS SAA research has been conducted on them [65, 108, 48, 128, 18]. Monocular cameras may or may not give distance information, depending on the detection algorithms. For static obstacles, distance information is possible via monocular SLAM [35, 62]. However, for moving objects, only optical-flow algorithm is available [126, 106, 87], with no distance information. Other research efforts focus on fusion of both types of sensors [49, 24, 112].

On the other hand, collaborative SAA relies on vehicle-to-vehicle (V2V) long-range communication, broadcasting and subscribing GPS-based traffic data of aircraft nearby. Google provides a discussion on collaborative SAA systems in [39]. The transceiver candidates are Dedicated Short-Range Communications (DSRC) and Automatic Dependent Surveillance-Broadcast (ADS-B). The communication range goes from $900m$ (DSRC) [82] to $24km$ (ADS-B) [111]. However, collaborative SAA also has several short-comings. The first one is communication delay. ADS-B Out has a bounded transmission latency of up to $2s$ [5]. Other WiFi-based technologies such as WiMax and 4G LTE suffer from insufficient geo-spatial coverage and indefinite/unpredictable communication delays and are not suitable for safety critical communications [27]. Second, the position information is obtained from GPS, typically accurate to $5m$ near ground [136]. In addition, obstacles outside the network are not detected.

The collision avoidance problem of interest involves high-speed vehicles. Typical quadrotors and helicopters can travel at $30m/s$ and $70m/s$, respectively. The $400m$ detection range from non-collaborative SAA implies a $4s$ reaction time under perfect weather condition, which may not be realistic in worst-case collision with uncertainties. The large range requirement also indicates its insensitivity to GPS position errors ($5m$, or 1.3%). Therefore, collaborative SAA is a more feasible choice. Indeed, one can observe that the emergence of the small UAS industry today stimulates the rapid growth of small-size ADS-B technology. Examples are XPS-TR from Sagetech [111] ($100g$ and $12W$ max power) and PING2020 from uAvioni [97]. In this chapter, we conservatively assume that the vehicles of interest are equipped with collaborative SAA. Therefore, the collision avoidance algorithm must be capable of handling communication delays.

Given the collision detection mechanism, what remains is the collision avoidance algorithm, which constitutes the contribution of this chapter. The safety controller generates optimal control actions when collisions are possible. The algorithm should be able to incorporate bounded uncertainties such as communication delay, wind disturbance, sensor uncertainty, and vehicle dynamics. However, the uncertainties are vehicle and scenario specific, and their numerical values are not the focus of this chapter. Our simulation only show a typical scenario, indicating our algorithm can handle these types of uncertainties. We want to address the following scenario: A manned helicopter ambulance equipped with (ADS-B)-Out is flying at a low altitude to pick up a patient, when a quadrotor UAS with (ADS-B)-In is delivering packages autonomously. The quadrotor can receive real-time traffic information of the helicopter. If collision is possible, the quadrotor would avoid the manned helicopter. The avoidance maneuver is performed exclusively by the quadrotor UAS.

In the given drone-helicopter scenario, the current right of way procedure would require to stop all drone activities until the helicopter completes its mission. However, this procedure is highly inefficient and costly because quadrotors consume a lot of energy when hovering. In addition, the pilot error issue is not addressed. In a safety critical system, the proposed safety controller provide the last safety guarantee when everything in the current procedure failed.

6.2 Literature Review

Aircraft collision avoidance algorithms has a rich history in the literature, and is generally referred to as the free-flight problem. There are geometry and probability based approaches [37, 94, 66, 60], as well as optimal control based methods [79, 44, 114, 78]. Our goal is to reduce the side effect of avoidance as much as possible, which implies optimality.

The first attempt on addressing the collision avoidance problem in an optimal manner is presented in [114, 78]. The avoidance problem is formulated as a path planning problem, in which the optimal path could be found by solving a mixed integer programming problem. The forbidden zone is set by rectangular constraints for simplicity. However, this is a centralized approach, which requires avoidance actions from both vehicles. In addition, solving MIQP becomes computationally expensive when the number of binary variables increases.

In [44], the collision avoidance problem between multiple quadrotors is solved by a decentralized optimal control approach. The optimal control is defined as the maximum acceleration with a constant optimal heading. Consequently, a real-time avoid set can be computed. The avoidance is activated only when collision becomes inevitable. However, the safety controller is derived from a simplified two-dimensional model. Once the controller is applied to a full 3D dynamic model, the optimality is not longer guaranteed due to delays from rotational inertia and aerodynamic drag. Although the model difference is almost negligible in the low-acceleration ($\sim 1.5m/s^2$) low-speed ($\sim 2m/s$) drone-drone avoidance scenario [44], it is no longer valid in the high-acceleration ($\sim 10m/s^2$) high-speed ($\sim 100m/s$) drone-helicopter scenario.

For UASs, the introduction of ADS-B facilitates the detection and avoidance of nearby manned and unmanned aircraft. ACAS X applies equally well to collision avoidance between UASs. The autonomous nature of UASs with ADS-B make collision avoidance between them much easier because it reduces uncertainty due to pilot errors. With predictable trajectories, collisions can be resolved much earlier in time before potential collisions can occur. But avoiding manned aircraft could be a little tricker when we assume human pilots has top priority in the airspace. In this chapter, we would like to investigate an alternative approach to address the collision avoidance problem between a small UAS with ADS-B In and a manned helicopter with ADS-B Out.

6.3 Safety Controller

6.3.1 Horizontal Safety Controller from Kinematic Model

To keep the system implementable in real time, the safety controller is derived from a simplified horizontal kinematic model [44]. The safety controller includes two components. The first component is a horizontal avoidance maneuver, $\boldsymbol{u_h} = [a_h \ \theta_h]^T$, which represents the quadrotor's acceleration $\boldsymbol{u_h}$ with magnitude $a_h \in [0, a_{max}]$ and direction $\theta_h \in [0, 2\pi]$, where θ_h is measured from the positive x-direction in the relative frame centered at the helicopter (Figure 6.1). The second component is a switching surface, or the avoid set boundary ∂K_h, in which the optimal avoidance maneuver $\boldsymbol{u_h^*}$ must be initiated to avoid intrusion into a safety set S_h. We will elaborate the two components below in details.

Figure 6.1 highlights some position trajectories in solid red lines under optimal control $\boldsymbol{u_h^*}$. The helicopter is at the origin, and the quadrotor has a relative position $\boldsymbol{x}(t) = (x(t), y(t))$ and is heading toward the helicopter with relative velocity $\dot{\boldsymbol{x}}(t)$. The quadrotor should not enter the safety set S_h in the present time $t = 0$, defined by the solid gray circle with radius r_{min} around the helicopter. The set of all $\boldsymbol{x}(0)$ lies on the boundary of ∂S_h.

To avoid entering S_h, the quadrotor starts the avoidance maneuver at some initial time $t = t_0$ in the past when touching the avoid set K_h. The set of all possible $\boldsymbol{x}(t_0)$ defines the boundary of the avoid set, ∂K_h, indicated by the dashed red curve in Figure 6.1. The complement of K_h is the maximal controlled invariant set [72], in which there is no restrictions on control. If optimal control $\boldsymbol{u_h^*}$ is applied during $[t_0, 0]$, then the quadrotor would fly tangent to the boundary of the safety set ∂S_h at the present time $t = 0$, and does not enter S_h.

For convenience, the relative frame is rotated such that $\dot{x}(t_0) = 0$ and $\dot{y}(t_0) < 0$ on ∂K_h. In other words, points on ∂K_h only have velocity components in the -y axis. For legibility, we abuse the notation here by hiding this rotation. A backward rotation should be made after the optimal control angle θ_h^* is computed [44].

Define state variables $\boldsymbol{x_h}(t) = [\boldsymbol{x}(t); \dot{\boldsymbol{x}}(t)]$, the relative kinematics is:

$$\begin{aligned} \dot{\boldsymbol{x}}_{\boldsymbol{h}}(t) &= f(\boldsymbol{x_h}(t), \boldsymbol{u_h}(t)) \\ \frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \\ \dot{x}(t) \\ \dot{y}(t) \end{bmatrix} &= \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ a_h(t)cos\theta_h(t) \\ a_h(t)sin\theta_h(t) \end{bmatrix} \end{aligned} \tag{6.1}$$

Consider the dynamical system (6.1) over the time interval $[t_0, 0]$ and define a continuously differentiable function $l : \mathbb{R}^4 \to \mathbb{R}$ in (6.2). If $l(\boldsymbol{x_h}(0)) \geq 0$, then we are outside of S_h [72].

$$l(\boldsymbol{x_h}(0)) = x^2(0) + y^2(0) - r_{min}^2 \tag{6.2}$$

The objective is to maximize the value function $J(\boldsymbol{x_h}(t), \boldsymbol{u_h}(t), t) = l(\boldsymbol{x_h}(0))$ subjected to dynamical and input constraints, defined in (6.3). Note that the value function only has a terminal value at time $t = 0$.

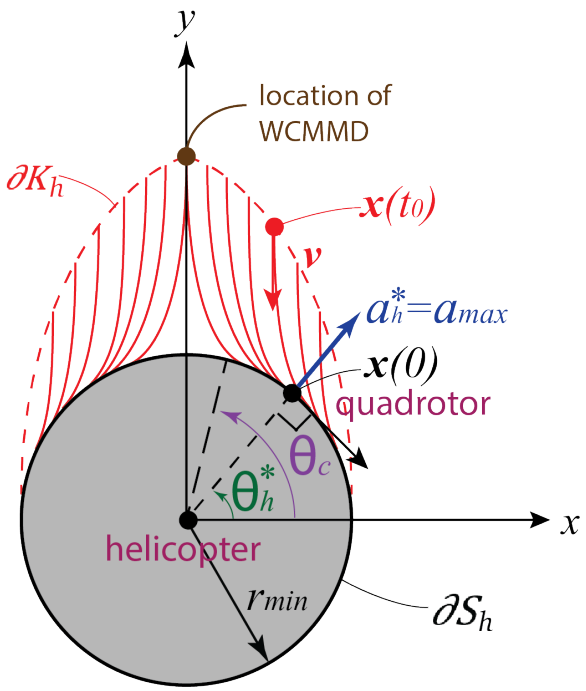


Figure 6.1: The safety set S_h is in gray and the avoid set K_h is in red.

$$\begin{aligned} \max_{\boldsymbol{u_h}} \quad & J(\boldsymbol{x_h}(t), \boldsymbol{u_h}(t), t) = l(\boldsymbol{x_h}(0)) \\ \text{subject to} \quad & f(\boldsymbol{x_h}, \boldsymbol{u_h}) - \dot{\boldsymbol{x}}_{\boldsymbol{h}} = 0 \\ & 0 \leq a_h \leq a_{max} \\ & 0 \leq \theta_h \leq 2\pi \end{aligned} \tag{6.3}$$

Use Pontryagin's Maximum Principle [44], we obtain the optimal acceleration for $t \in [t_0, 0]$.

$$a_h^*(t) = a_{max}; \quad \theta_h^*(t) = atan2\left(\frac{y(0)}{x(0)}\right) \tag{6.4}$$

Secondly, to answer the question on when to apply the optimal control, we need to compute the avoid set $\partial K_h = \{(x(t_0), y(t_0))\}$ by kinematics [8]. Define the relative velocity $v = |\dot{y}(t_0)|$. Given a particular θ_h^* and r_{min} on ∂S_h, we have

$$\begin{aligned} x(t_0) &= \left(r_{min} - \frac{v sin^2\theta_h^*}{2a_{max}}\right) cos\theta_h^* \\ y(t_0) &= \left(r_{min} + \frac{v}{2a_{max}}\left(1 + cos^2\theta_h^*\right)\right) sin\theta_h^* \end{aligned} \tag{6.5}$$

The derivation of (6.5) is provided in Appendix .

From Figure 6.1, we can see that θ_h^* is only defined on certain parts of ∂S_h: $\theta_h^* \in [0, \theta_c] \cup [\pi - \theta_c, \pi]$ for some $\theta_c \triangleq \theta_h^*\Big|_{x(t_0)=0}$ shown in (6.6). At $\theta_h^* = \theta_c$ (or $x(t_0) = 0$) and speed v, we have the worst-case scenario, giving us the worst-case minimum maneuver distance (WCMMD) to be $y(t_0)\Big|_{\theta_h^*=\theta_c}$ in (6.5).

$$\theta_c = sin^{-1}\frac{\sqrt{2a_{max}r_{min}}}{v}, \quad 2a_{max}r_{min} < v^2 \tag{6.6}$$

Each point on ∂K_h maps uniquely to a θ_h^* on ∂S_h. Therefore, we first generate ∂K_h using (6.5) for a dense set of $\theta_h^* \in [0, \theta_c] \cup [\pi - \theta_c, \pi]$, then compare the quadrotor's relative position $\boldsymbol{x}(t)$ to points on ∂K_h. If $\boldsymbol{x}(t)$ is very close to a particular point on ∂K_h, then we can find the corresponding θ_h^* and apply the optimal acceleration $\boldsymbol{u_h^*} = [a_h^*\ \theta_h^*]^T$.

6.3.2 Vertical Safety Controller from Kinematic Model

In this section, we design the vertical safety controller. Define $\boldsymbol{x_v} = [z(t); \dot{z}(t)]$, where z represents the relative height referenced from the helicopter. The new kinematics system is a combination of (6.1) and (6.7), with $a_h = 0$ in (6.1) and (6.7) being the vertical kinematics. The constraint on $a_v(t)$ is from [47].

$$\begin{aligned} \dot{\boldsymbol{x}}_{\boldsymbol{v}}(t) &= f(\boldsymbol{x_v}(t), a_v(t)) \\ \frac{d}{dt}\begin{bmatrix} z(t) \\ \dot{z}(t) \end{bmatrix} &= \begin{bmatrix} \dot{z}(t) \\ a_v(t) \end{bmatrix}, \quad -a_{v,max} \le a_v(t) \le a_{v,max} \end{aligned} \tag{6.7}$$

We define a new cylindrical safety set ∂S_v by extruding the original safety set ∂S_h in Figure 6.1 upward and downward by a minimum height h_{min} (Figure 6.2). The goal is to ensure the quadrotor is outside of the cylinder.

To make a fair worst-case comparison with the horizontal safety controller, we assume that $\boldsymbol{x_v}(t_0) = 0$, or the two vehicles are at the same altitude with zero vertical speed before avoidance. Formulate the optimal control problem according to [44], the optimal control is given by (6.8).

$$a_v^* = a_{v,max}\ sgn(z(t_0)), \quad a_{v,max} \in (0, g) \tag{6.8}$$

Ideally, (6.8) says that when the quadrotor is below the helicopter at t_0, apply minimum thrust to obtain a large downward acceleration. Otherwise, when the quadrotor is above the helicopter, apply the maximum thrust. In reality, the available downward acceleration is less than g due to drag and the minimum thrust constraint in (5.1) to maintain the desired quadrotor pose.

The vertical avoid set is derived again from backward dynamics. For a dense set of $\theta_h \in [0, \pi]$ and $|z(t_0)| \in [-h_{min}, h_{min}]$, we calculate the new avoid set $\partial K_v = \{(x(t_0), y(t_0))\}$ satisfying (6.9).

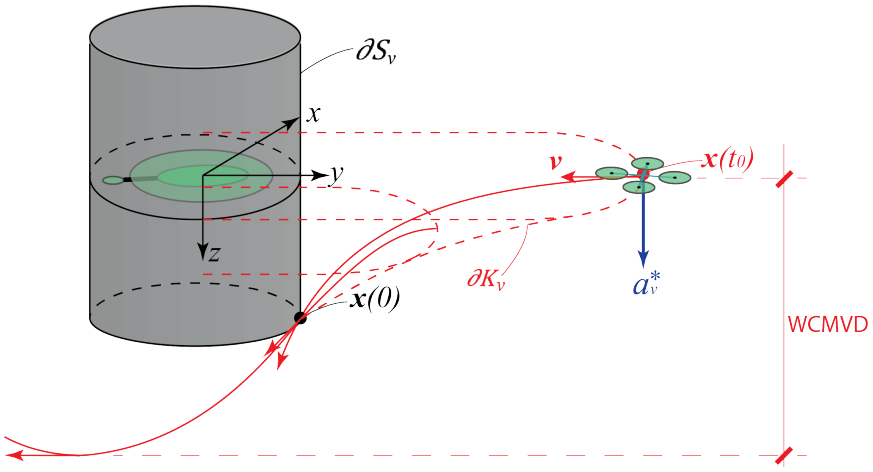


Figure 6.2: Visualization of downward vertical avoidance. The safety set S_v is in gray, and the avoid set K_v is in red.

$$\begin{aligned} x(t_0) &= r_{min} cos\theta_h \\ y(t_0) &= r_{min} sin\theta_h + v\sqrt{\frac{2\left(h_{min} - |z(t_0)|\right)}{|a_v^*|}} \end{aligned} \tag{6.9}$$

The shape of the avoid set ∂K_v is a half circle offset by a certain distance away from the helicopter indicated by the dashed red lines in Figure 6.2. Once the quadrotor hits ∂K_v, the safety controller is activated. The WCMMD is obtained by substituting the relative altitude $z(t_0) = 0$, relative heading $\theta_h(t_0) = \pi/2$, and maximum relative speed v into equation (6.9). or the quadrotor is at the same altitude as the helicopter and heading toward it right before the avoidance. The MMD drops to r_{min} as $|z(t_0)|$ approaches h_{min} either upward or downward.

In the vertical avoidance, we are concerned with not only MMD, but also the maximum vertical deviation (MVD), or the difference in altitude before and after the vertical avoidance. The worst-case MVD (WCMCD) in Figure 6.2 occurs together with the WCMMD. Right after a downward avoidance, the vertical speed of the quadrotor reaches a local maximum, and it will continue to go down for a while until it starts rising back to the original altitude. The downward MVD values are determined by h_{min} when accelerating and the maximum thrust when decelerating.

6.3.3 Refine the Avoid Sets

The kinematic safety controllers is a good starting point. However, maximum accelerations a_h^* and a_v^* could not be achieved instantaneously and constantly in reality. For a quadrotor, rotational inertia and aerodynamic drag delays reduce the maximum acceleration. Therefore, the safety controller have to be activated a little earlier before touching the avoid set ∂K_h, we define this time to be the earlier reaction time (ERT). In the worst-case, when the quadrotor

and the helicopter are flying directly towards each other at maximum speed, we have the worst-case earlier reaction time (WCERT). In this section, we show two implementation refinements. First, we need to convert the acceleration command from the safety controller to Euler angle commands. This process is described in Section 6.3.3.1 for the horizontal safety controller and Section 6.3.3.2 for the vertical safety controller. Then, we present an iterative algorithm to obtain the WCERT with a quadrotor dynamic model in Section 6.3.3.3. The WCERT values are then used to refine the avoid set to ensure safety.

6.3.3.1 Horizontal Control Conversion

To convert the horizontal kinematic control $\boldsymbol{u_h^*}$ to horizontal control inputs U_2 and U_3 in Section 5.3.2, we assume that the quadrotor is in vertical equilibrium when performing the avoidance. Then the inputs that could affect the horizontal motions are the desired roll (ϕ_d), pitch (θ_d), and yaw (ψ_d). It turns out that yaw is not effective, so we choose roll and pitch. The conversion is straight forward. First, convert the optimal acceleration $[a^*, \theta^*]$ from the relative frame to the quadrotor body frame $\mathcal{B}$, and then convert it to the desired pitch (θ_d) and roll (ϕ_d). Equation (6.10) summarizes the process.

$$\begin{aligned} a_x^* &= a_h^* \cos(\theta^* - \psi) \quad a_y^* = a_h^* \sin(\theta^* - \psi) \\ \theta_d &= -atan2(a_x^*, g) \quad \phi_d = atan2(a_y^*, g) \end{aligned} \tag{6.10}$$

Lastly, the low level controllers in Chapter 5 is simulated to track θ_d and ϕ_d defined in (5.19) using the control inputs U_2 and U_3 defined in (5.1).

6.3.3.2 Vertical Control Conversion

To apply the vertical kinematic control to the 3D model, we use the following conversion by ignoring drag. In this case, we can control U_1 directly with constraints in (5.1), and no low-level controllers are needed.

$$\theta_d = \phi_d = 0; \quad U_1^* \approx g - a_v^*$$

6.3.3.3 Iterative Algorithm

With control conversion (5.19), we can apply kinematic control law (6.4) at ∂K_h, defined by (6.5) and (6.6), to the dynamical system (5.2) with control inputs (5.1). However, the quadrotor will intrude the safety set S_h due to control delay introduced from rotational inertia and aerodynamic drag in the full dynamic model. To resolve the intrusion, we need to compute the worst-case earlier reaction time (WCERT), or the time the quadrotor has to react earlier before touching the safety set S_h, defined in Section 6.3.1, to prevent collision in the worst-case scenario.

Algorithm 1 computes WCERT for the horizontal safety controller. The function $\boldsymbol{x}(t) = worst\_case(wcert)$ simulates the worst-case avoidance scenario when the kinematic controller

Data: dt, $wcert$
Result: $wcert$

dt; % time increment
$wcert = 0$; % current WCERT
$\boldsymbol{x}(t) = worst\_case(wcert)$;
while $min_t|\boldsymbol{x}(\boldsymbol{t})| < r_{min}$ **do**
  $wcert = wcert + dt$;
  $\boldsymbol{x}(t) = worst\_case(wcert)$;
end

Algorithm 1: Iterative algorithm to compute WCERT.

reacts $wcert$ earlier, and return the relative horizontal position $\boldsymbol{x}(t)$ for the whole simulation period.

We start the simulation from the safety sets (6.5). If the minimum relative distance $min_t|\boldsymbol{x}(\boldsymbol{t})|$ is less than r_{min}, then we increment $wcert$ and simulate again. In this case, we set dt to be $0.02s$ in the case without drag and $0.05s$ with drag, which gives a spacial resolution of $2m$ and $5m$, respectively, when maximum speed is $v = 100m/s$.

Algorithm 1 is applied to a range of a_{max} and v defined in Section 6.3.1. The result is a look-up table WCERT(a_{max},v). With this look-up table, WCMMD can be re-computed. The generation of WCERT values is time-consuming but performed off line. In real flights, we can just run the light-weight kinematic safety controller in real-time but react WCERT(a_{max},v) earlier. If the dynamic model in the simulation is close to the real dynamics, then we are safe. To obtain usable results, system identification should be performed to establish conservative quadrotor parameters, which is out of the scope of this chapter and not discussed. Interested readers can refer to [14].

6.3.4 The Hybrid Controller

The position controller with DSC filters mentioned in Section 6.3.3.1 and the safety controller together constitute a hybrid controller. The safety controller decides when the safety maneuver starts and terminates. Figure 6.3 shows the hybrid automaton of the controller switching mechanism. At the beginning, the quadrotor receives a waypoint command, and the position controller starts to drive the quadrotor to the destination. When a helicopter is detected, an avoidance set calculation is performed using equation (6.5). The switching occurs when the quadrotor intrudes ∂K_h, or$||\boldsymbol{x}(t)||_2 \leq ||\boldsymbol{x}(t_0)||_2$. Once the safety controller is activated, the quadrotor avoids the helicopter by applying a_{max} in the optimal direction θ_h^*. After $||\boldsymbol{x}(t)||_2 > ||\boldsymbol{x}(t_0)||_2$, the optimal controller is deactivated, and the quadrotor switches back to position control. If another helicopter is passing by, the optimal control is enabled again.

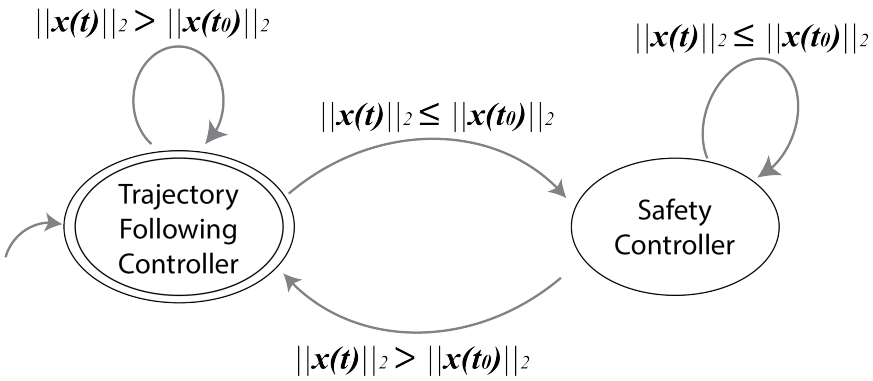


Figure 6.3: The hybrid automaton indicating the controller switching mechanism.

6.4 Simulation

The simulations are divided into two parts. The first part examines the behavior of the two safety controllers, and the second part presents how the hybrid automaton behaves in the scenario defined at the end of Section 6.1.

Before presenting the results, we should first justify the parameters chosen for the simulations. The parameter values, listed in (6.11), are chosen carefully and conservatively to model a realistic quadrotor model (3DR Solo [2]). However, these parameters vary a lot from different quadrotors, communication systems, and wind conditions. Therefore, the results are not intended to cover all possible cases. What we can guarantee is that our safety control algorithm could give conservative results given conservative parameters for a given quadrotor with a known communication delay and bounded wind disturbance.

$$\begin{array}{lll} m = 1.5\ kg & r_{min} = 20m & I = diag([5\ 5\ 8]e\text{-}3)\ kg \cdot m^2 \\ k_m = 10^{-6}\ Nm \cdot s^2 & l = 0.2m & k_f = 5.0 \times 10^{-6}\ N \cdot s^2 \\ c_t = 0.50\ Ns/m & c_r = 0.10\ Nm \cdot s^2 & \boldsymbol{g} = [0\ 0\ 9.81]^T\ m/s^2 \end{array} \tag{6.11}$$

There are four key parameters which dominates the simulation results, namely the rotational inertia (RI) I, the translation drag coefficient c_t, the communication delay Δt_{delay}, and the minimum separation distance r_{min}.

- $\boldsymbol{I} = diag(I_x, I_y, I_z)$: We approximate the rotational inertia (RI) by approximating the 3DR Solo as a solid cylinder of radius $0.1m$ and height $0.1m$. The calculation follows from [8], and the system identification results in [14] validates our calculation.

- c_t: Instead of adopting results from [14], which yields an unrealistically large terminal speed, we picked a more conservative value of $0.5Ns/m$, which yields a terminal horizontal speed of $v_{max} = 50m/s$, by assuming a maximum horizontal thrust of $\sqrt{3}mg$ and translation drag of $c_t v_{max}$ [84].

- Δt_{delay}: In all the feasible collaborative SAA technologies discussed in Section 6.1, ADS-B gives the largest bounded communication delay, $2.0s$. We use this conservative value throughout the simulations, to approximate the worst-case behavior [5, 43].

- r_{min}: The minimum separation distance r_{min} is tunable to satisfy future minimum separation standards. It is set $20m$, or twice the size of a typical helicopter, throughout the simulations.

The simulation is expected to be realistic, although a hardware implementation is necessary to confirm the effectiveness of the hybrid controller. It will be left as future work.

6.4.1 Safety Controllers

6.4.1.1 Horizontal Safety Controller

In this section, two set of simulation results are presented. First, we examine how the 2D safety controller behaves when applied to a full 3D quadrotor model. Then, we present the WCMMD and WCERT results after running Algorithm 1 defined in Section 6.3.3.3, covering all practical accelerations and relative velocities for our specific setup. All safety sets are produced by griding 120 points evenly for $\theta_h^* \in [0, \theta_c] \cup [\pi - \theta_c, \pi]$.

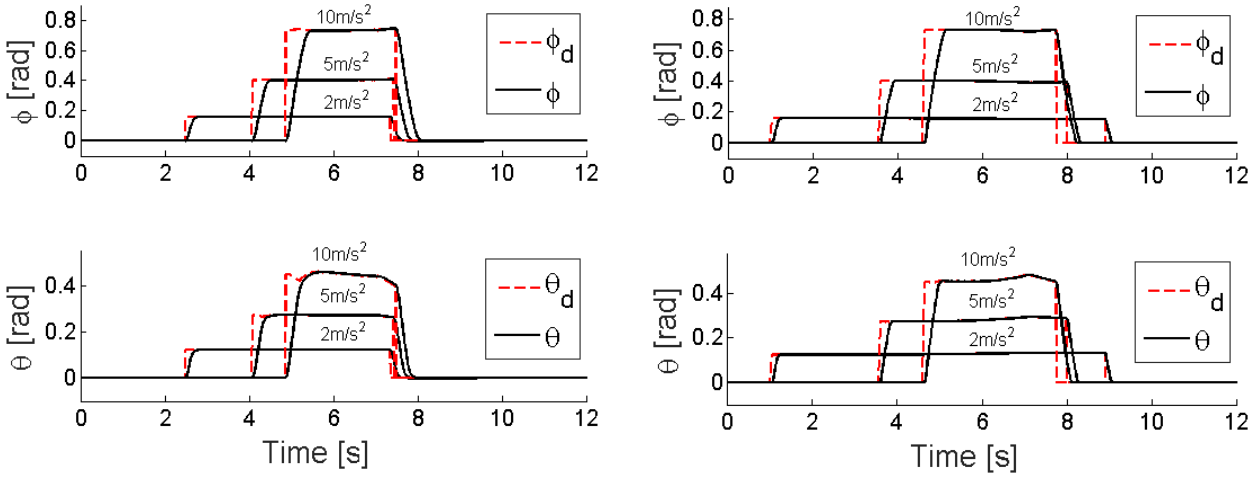
Behavior: Figure 6.4 shows the behavior of the safety controller. Two factors, namely the magnitude of the maximum acceleration and drag force, are investigated. The dashed red lines are the reference signals generated from the high-level safety controller, while the black solid lines are the tracking signals from the low-level sliding mode angle controllers [14]. The quadrotor is originally at rest. The parameters are

$$a_{max} = [2\ 5\ 10]\, m/s^2 \qquad v = 70m/s$$

In Figure 6.4a, when a_{max} is low, say $2m/s^2$, the constant acceleration assumption holds very well, but in sacrifice it takes a long time ($4.9s$) to finish the avoidance maneuver. At the beginning of the avoidance maneuver, the tracking delay between the dashed red line and the solid black line is $0.28s$, which results in a ERT of $0.1s$.

The avoidance duration is shortened to half as much ($2.6s$) when $a_{max} = 10m/s^2$, but the desired pitch θ_d has some variation at the beginning and the end of the avoidance. The variation exists because the quadrotor could not achieve the desired acceleration instantaneously due to rotational inertia (RI). Nonetheless, the assumption of applying constant maximum acceleration remains valid. The tracking delay is $0.56s$, which gives an ERT of $0.2s$. When $a_{max} = 5m/s^2$, the result is somewhere in between. As a_{max} increases, the tracking delay increases, and ERT increases almost propotional to the tracking delay. Therefore, without drag, ERT is solely determined by the tracking delay at the beginning of the avoidance maneuver.

In the normal case with drag force (Figure 6.4b), the variation in desired pitch (θ_d) is worse, although the desired roll (ϕ_d) stays almost constant. θ_d is generally increasing during



(a) without drag force

(b) with drag force

Figure 6.4: The 2D optimal safety control is applied to a 3D quadrotor model.

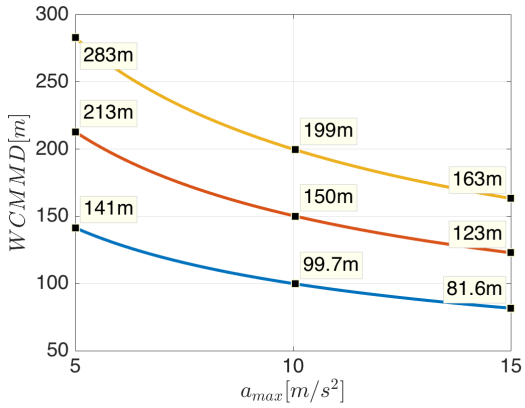
the avoidance maneuver, especially at $a_{max} = 10m/s^2$. It makes sense because as the velocity goes up, the drag force goes up, and to generate an equivalent a_{max}, a larger pitch angle is needed. The avoidance duration is longer, range from $7.9s$ (when $a_{max} = 2m/s^2$) to $3.1s$ (when $a_{max} = 10m/s^2$).

However, with drag, ERT is not a strong function of tracking delay anymore. With similar tracking delays, the ERT is $1.54s$ when $a_{max} = 2m/s^2$ and $0.42s$ when $a_{max} = 10m/s^2$. This result is the complete opposite to the no-drag case. As a_{max} increases, the drag force increases faster, and to counteract the drag, a larger ERT is required.

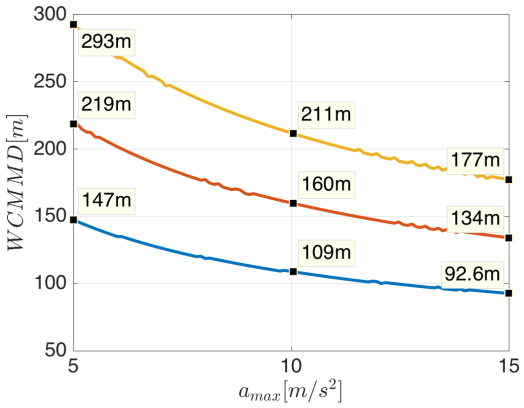
Worst-case scenarios: In this section, we discuss the simulation results of the safety controller considered in Section 6.3. The WCMMD and WCERT are observed for a range of maximum horizontal relative velocity v and maximum horizontal accelerations a_{max}. In these worst-case scenarios, the quadrotor and helicopter are heading directly to each other. The effect of rotational inertia (RI), drag forces, and communication delay described in Section 6.4 are evaluated.

The ranges for a_{max} and v are chosen for reasons. A quadrotor can typically produce a maximum thrust of twice as its weight, or $2mg$, which can generate a maximum horizontal acceleration of $17m/s^2$ when tilting at 60° at a constant altitude. Therefore, we picked a_{max} to be between 5 and $15m/s^2$. A commercial helicopter typically has a maximum horizontal speed of $70m/s$, while a quadrotor has a maximum cruise speed of $30m/s$. Therefore, we set the maximum horizontal relative speed v to be between 50 and $100m/s$.

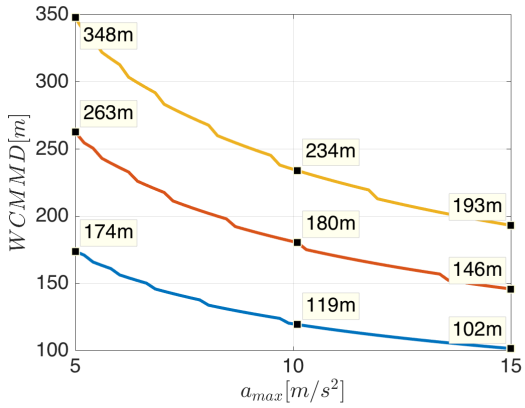
For clarity, we only present WCMMD results for three relative speeds (Figure 6.5). Figure 6.5a shows the WCMMD result by using the horizontal safety controller on a kinematic quadrotor. It serves as the base of comparison. Figure 6.5b, 6.5c, and 6.5d give the WCMMD results by using the horizontal safety controller on a quadrotor with RI, drag, and communication delay, respectively. The axis ranges are the same as Figure 6.5a for comparison



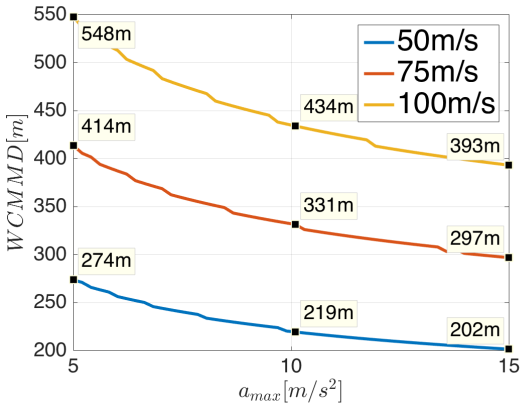
(a) kinematics only



(b) with RI

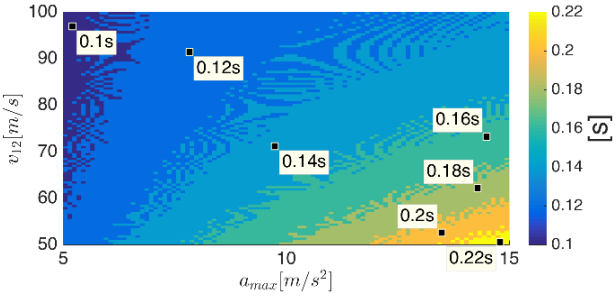


(c) with RI and drag

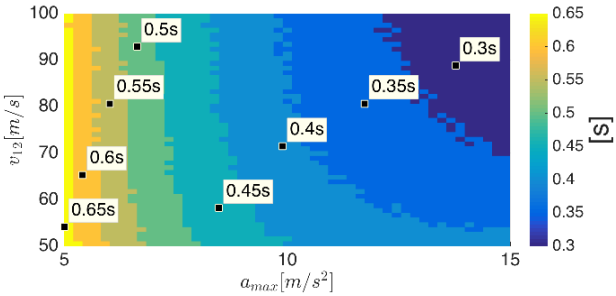


(d) with RI, drag, and comm. delay

Figure 6.5: Summary of WCMMD for the horizontal safety controller, as defined in Section 6.3.1 ($r_{min} = 20m$).



(a) with RI



(b) with RI and drag

Figure 6.6: Summary of WCERT for the horizontal safety controller, as defined in Section 6.3.3.3 ($r_{min} = 20m$).

purpose.

The WCMMD increases faster and faster from the lower-right corner to the upper-left corner. In the lower-right corner, when a_{max} is high and v is low, we get small WCMMD ranging from $93m$ (no drag) to $202m$ (with drag and communication delay). In the upper-left corner, with small a_{max} and large v, we get large WCMMD ranging from $293m$ to $548m$. Compared with Figure 6.5a, RI gives a maximum of $10m$, or 3%, increase in the WCMMD. The WCMMD increments due to drag are $55m$, or 18%, in the upper left corner and $10m$, or 10%, in the lower right corner. Lastly, depending on the relative velocity v, the WCMMD increment of communication delay range from $100m$ (30%) to $200m$ (60%).

The more interesting result comes in the WCERT maps. First, the WCERT values without drag is always smaller than the ones with drag. The WCERT ranges from $0.10s$ to $0.22s$ without drag and from $0.30s$ to $0.65s$ with drag. Second, the two WCERT maps in Figure 6.6 show opposite patterns with respect to a_{max}. With RI only (Figure 6.6a), the WCERT increase as a_{max} increases. It is because a larger a_{max} implies a larger tilting angle (ϕ_d and θ_d), which takes more time for the sliding mode controllers to rotate the quadrotor. With both RI and drag (Figure 6.6b), the WCERT decreases as a_{max} increases. It says that as a_{max} gets larger, it becomes easier to overcome the drag, and the WCERT becomes smaller. To take into account the communication delay, one can simply add the delay ($2.0s$ for ADS-B) on top of Figure 6.6.

6.4.1.2 Vertical Safety Controller

In this section, we present results for the vertical safety controller. For a cylindrical safety set with radius r_{min} and height h_{min} equal to $20m$, and a maximum vertical acceleration of $1g$, the WCMMD result is shown in Figure 6.7. Unlike the horizontal controller, the result is purely a function of the relative horizontal speed v. When $v = 100m/s$, we need a WCMMD of $220m$ with drag and $365m$ with drag. Drag gives WCMMD increments from $20m$ (17%) to $40m$ (18%). The $1sec$ communication delay adds the largest increments in WCMMD, and the drag adds about $40m$. Rotational inertia has no effect because the quadrotor does not rotate in this scenario. Nonetheless, the WCMMD due to drag and delay are consistent with the results in Figure 6.5b, 6.5c, and 6.5d, respectively, at a constant acceleration of $10m/s^2$.

Table 6.1 shows the results for WCERT and WCMVD as defined in Section 6.3.2. The WCERT is zero without drag and $0.42sec$ with drag. In this case, since the quadrotor does not rotate when performing vertical avoidance, there is no tracking delay, and thus no WCERT without drag. With drag, WCERT only depends on maximum acceleration a_{max}, which is a constant $a_{v,max}$, thus we have a constant WCERT.

Notice that the WCMVD with drag ($26.6m$) is smaller than without drag ($39.3m$). The difference is produced by drag. Without drag, the quadrotor in minimum thrust is almost experiencing free fall, with only minimum thrust $4k_f\omega_{r,min}^2$ to maintain the quadrotor pose. Constraint in (6.7) implies that the quadrotor stabilizes itself at an altitude deviation of about $2h_{min}$ by symmetry. With drag, we can actually stop the quadrotor faster because

Table 6.1: The WCERT and WCMVD results for the vertical safety controller.

	WCERT $[s]$	WCMVD $[m]$
w/o drag	0.00	39.3
w/ drag	0.42	26.6

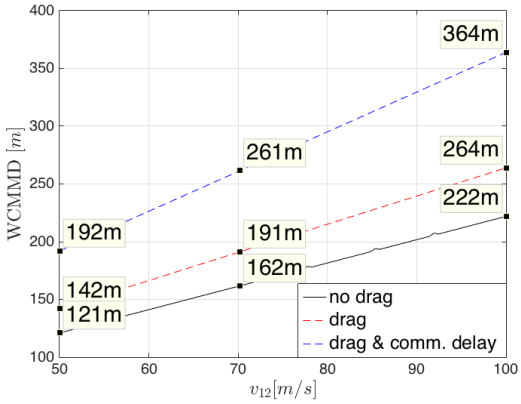


Figure 6.7: The WCMMD result for the vertical safety controller ($r_{min} = h_{min} = 20m$).

the vertical drag also contributes to the deceleration. Therefore, the conservative WCMVD is from the no-drag case.

6.4.1.3 Comparison

We see a trade off when the practical constraints of the horizontal and vertical controllers are considered. Compared with the vertical controller, the horizontal controller has demanding requirements on the tilting angle, but it does not lose altitude given enough thrust. Therefore, both controllers have advantages in different applications. When the quadrotor is flying over a wide-open area or is heavily loaded, the vertical controller is preferred since losing $50m$ of altitude would not affect the navigation too much, but tilting the quadrotor by 45° is difficult. On the other hand, if the quadrotor is flying in areas with high obstacle density without much payload, the horizontal controller becomes preferable. In this case, losing $10m$ of altitude may force the quadrotor to hit a tall building or tower.

6.4.2 Hybrid Controller

Here we present a typical package-delivery scenario with the horizontal safety controller. A quadrotor is typically designed to produce a maximum thrust of $2mg$, which theoretically can give a maximum horizontal acceleration of $14m/s^2$ when hovering. In this scenario, we use a more conservative value of $10m/s^2$. Figure 6.8 shows a high-speed high-acceleration scenario. In Figure 6.8a, the quadrotor takes off from the origin and is trying to get to the

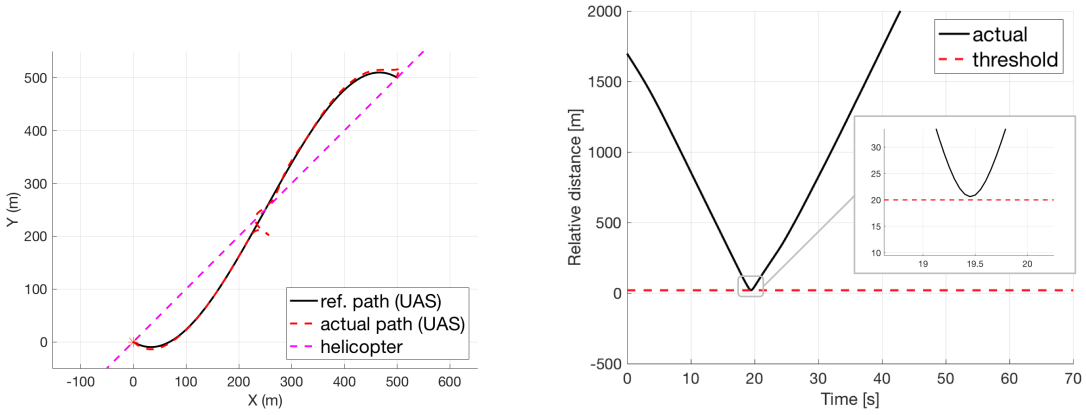
point (X, Y, Z) = (500m, 500m, -100m) when a helicopter is flying toward it in a smooth spline. In this scenario, the parameters are

$$v_{heli} = 70m/s \quad v_{quad} = 20m/s \quad v = 90m/s$$
$$r_{min} = 20m \quad a_{max} = 10m/s^2$$

By employing the safety controller, the quadrotor successfully avoids the helicopter and gets to the final destination (Figure 6.8a) with relative distances to the helicopter always less than the threshold . Assume that there is no communication delay, when the WCERT is set to 0.1s, the minimum relative distance between the two vehicles is 20.6m, which is barely larger than the nominal threshold of 20m, indicating that the safety controller performs optimally. Note that if the vehicles are equipped with ADS-B, the WCERT should take into account the 2.0s communication delay, giving a WCERT value of 2.1s.

Figure 6.9 shows the time history of the horizontal command and control signals. In the X, Y, and Z time history plots, the red dashed lines indicate desired destination, and the black solid lines give the actual flight trajectory. In the θ, ϕ, and ψ plots, the blue dashed lines show the angle command generated from either the MPC or safety controller, and the red dashed lines give command from the sliding position controller (see Section 5.3.3). At the beginning, trajectory following via MPC is enabled, and the actual trajectory (solid black) follow the MPC acceleration commands. At time $t = 17.4s$, the quadrotor switches to the safety (angle) control until $t = 19.9s$, which gives an avoidance duration of 2.5s. The tracking delay in pitch (θ) is long ($\sim 0.5s$) since the error between the desired pitch and the actual pitch is large at the beginning of the avoidance. After arriving at the destination ($t \geq 43.8s$), the quadrotor swiches to position control. Overall, the quadrotor follows the trajectory closely in MPC control mode.

The efficiency of the safety controller could also be evaluated by the total flight time. In this scenario, the total flight time without avoidance is 58s, and that with avoidance is 65s, which gives an overall navigation delay of 7s. After deducting the 2.5s avoidance maneuver, the navigation time is increased by 4.5s. If the destination were 10 miles away, the total flight time would be around 13 minutes. The optimality ensures that the extra time spent on avoidance is not significant compared to the whole trip.



(a) Horizontal paths.

(b) Relative distance time history.

Figure 6.8: The quadrotor avoids a helicopter on the way to a destination.

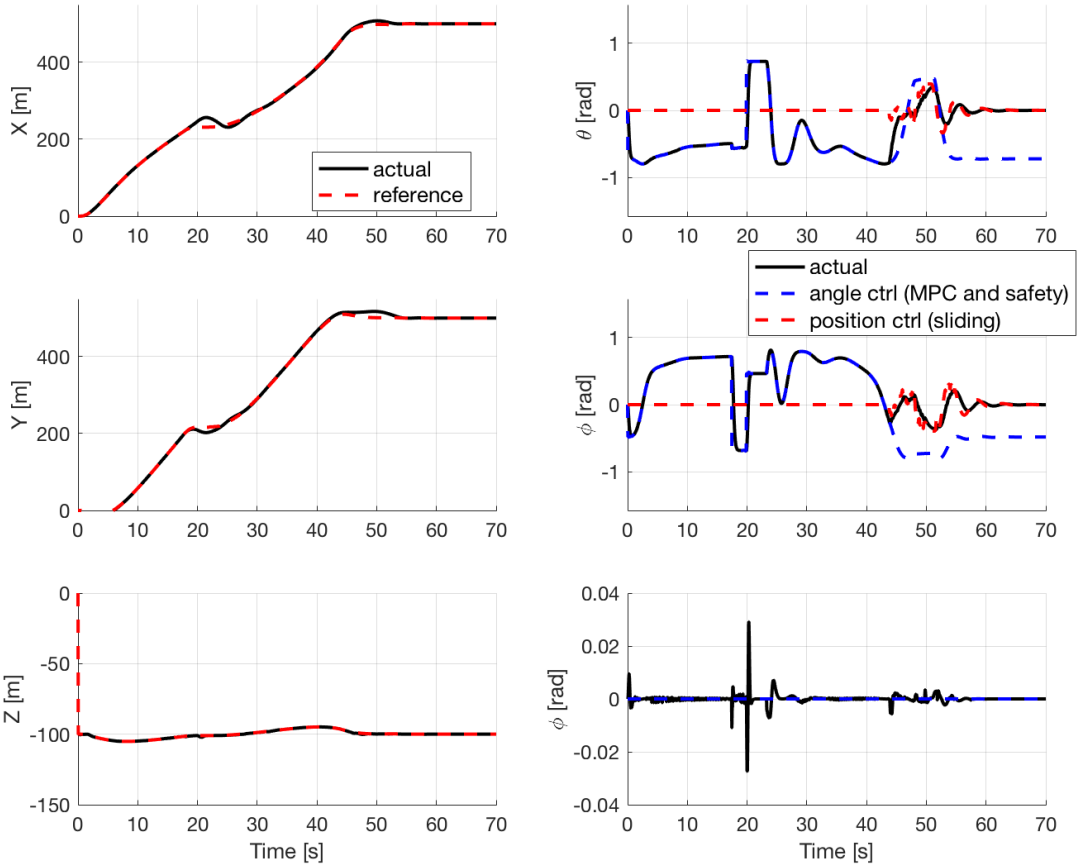


Figure 6.9: Time history of reference and control signals.

Chapter 7

Summary and Recommendations

7.1 Summary

NASA Unmanned Traffic Management (UTM) is a new research field building the fundamental framework for future unmanned aircraft system (UAS) flight operations. In this dissertation, we make an attempt to address the flight planning problem for a single multi-rotor UAS. We have demonstrated that the flight planning system is able to generate energy-optimal 4D trajectories for a UAS, given deterministic environmental information. The system mainly composes of a power consumption model at steady state and an optimal routing engine. This is the first time, to our knowledge, that the effect of wind on energy is optimized for a multi-rotor UAS. The flight planning trajectory is verified in the in-flight stage. We build a feedback controller capable of performing trajectory following as well as collision avoidance with a single manned aircraft. Together, they provide a comprehensive framework to perform a single UAS operation.

The main goal of the flight planning system is to assess the energy requirements, specifically the energy demand, to complete a mission safely (Chapter 2). In Chapter 3, we develop a multirotor power consumption model to perform the energy assessment. We focus on the steady state behavior of the UAS. The model consists of three components, namely induced power, profile power, and parasite power. We designed three experiments to identify the model parameters and validated the model for different airspeeds and payloads. Although we modeled the UAS as a single-propeller vehicle, the identification and validation results give good fitting results with small percent errors. Lastly, we use the model to simulate and illustrate how speed, altitude, and regional variation affect the energy consumption of a small UAS.

To minimize the energy consumption or trip time in a flight, we formulated an optimal control problem for UAS flight planning (Chapter 4). We start by formulating a general optimal control problem with cost and constraints. This problem is in general hard to solve due to the anisotropic nature of wind, meaning that power cost can vary depending on the navigation direction. But we show that by fixing either the ground speed or airspeed,

we can simplify the general formulation and solely optimize the navigation direction. The optimal path is solved numerically by the Ordered Upwind Method (OUM). By running the algorithm in elevated and smoothed terrain surfaces, or digital elevation models (DEM), and 3D wind fields, we can produce 4D trajectories capable of avoiding pre-specified static obstacles as well as achieving significant energy savings in a realistic scenario.

We then demonstrate the feasibility of the trajectories by designing controllers to follow them during flights. In general, it is difficult to follow a time-stamped trajectory due to re-planning triggered by external disturbances and uncertainties. Instead, we focus on speed-stamped trajectories. A two-level control architecture is proposed in Chapter 5. The high-level MPC controller takes cross-track and speed errors as inputs and issues acceleration commands as outputs. The acceleration commands are then converted to Euler angle commands. The low-level sliding controllers take the angle commands as reference inputs and generate the necessary forces and moments, or motor RPM's.

Lastly, we present a safety controller in the context of high-speed manned-unmanned aircraft collision avoidance. A computationally feasible 2D optimal safety controller minimizing the avoidance duration [44] is modified to be suitable for a high-speed quadrotor with rotational inertia and aerodynamic drag. The combined hybrid controller is capable of performing optimal avoidance when flying to a destination autonomously.

7.2 Recommendations

This dissertation is an attempt to address the problem of fully autonomous UAS flight management. We sketched down a set of pre-flight and in-flight tasks and proposed an initial set of solutions to each task. There are still many open problems to be addressed.

First, the flight planning procedure can be further optimized. The current flight planning system uses a brute force approach to optimize altitude and vehicle speed. We believe that by incorporating the vertical structure of wind, it is possible to simplify the altitude search process. Lower altitude flights are in general energy-saving but more risky due to obstacles. This trade-off could be studied further. For speed optimization, we only look for constant ground speed or constant airspeed trajectories. In reality, there could be better solutions with varying speeds along the trajectories. This part could be investigated in the future. Another big missing piece in this dissertation is battery capacity estimation. In Chapter 2, we assume that the battery capacity is perfectly known. But in reality, it is a complicated topic affected by many factors. If battery is the main energy source for UAS operations in the future, we need further research to address this issue. The flight duration of a multirotor is rather short. To overcome this bottleneck, exploring energy sources with higher energy density may also be a good research direction.

The most urgent task is to improve the quality of environmental data. Currently, the wind data resolution is not satisfactory for UAS operations. The best wind data we could find is simulated by the EFMH research lab [32]. Second, the fidelity of terrain and wind data publicly available is largely unknown [135]. Validation methods or assessments are

needed to quantify the statistics of such data. If statistical information such as variance or probabilistic distributions were available, we could try another promising routing solution based on risk-averse control, to further quantify the required energy contingency.

To ensure safety, energy estimation is critical. In this dissertation, we focus exclusively on the power consumption modeling of multirotor UAS. In reality, there are many other types of UASs. To further apply the optimal flight planning framework to UTM, we shall include other existing models for fixed wings and other aircraft accordingly. Another approach is to require UAS manufacturers to develop aircraft performance curves for their vehicles, which is the current practice in manned aviation. Second, instead of attaching a pitot tube, we could use a model to estimate wind. But this approach requires an accurate dynamic model of the aircraft, especially lift and drag parameters. Online parameter adaptation algorithms with some linear models can be applied to estimate such parameters. Recent development in neural network may be another feasible approach.

The optimal control problem for trajectory generation is rather preliminary. The biggest issue is that the current algorithm does not include vehicle dynamical constraints. Although the trajectories is inside a smoothed terrain surface, there is no guarantee that the UAS can follow the trajectory closely even without uncertainty. On approach to address this issue is to project the 3D OUM paths and obstacle information in the 2D horizontal plane, and formulate another optimization problem offline to generate cubic splines with minimal cross-track errors. The current MPC trajectory horizon is implemented in a similar manner but performed online. During the process, the speed commands can be modified accordingly. Second, the computation speed of OUM is also not satisfactory. The optimal cost map in Figure 4.12 takes minutes to hours to compute in Julia. If the running time could be reduced to a few seconds, we may possibly perform real-time re-routing and consider time-varying wind fields. Third, instead of restricting the paths in an uneven 2D surface, we may compute the paths in 3D by allowing a range of altitudes. But again, it requires a significant boost of computation speed. Lastly, to use OUM, the current work assumes known ground speed as a function of vehicle position. We can also explore the possibility to relax this assumption.

For the trajectory following controllers (Chapter 5), the main issue is to increase the robustness in the MPC controller. A simple extension is to use Stochastic MPC (SMPC) [90]. In SMPC, we could use long-run expected average cost, and the constraints can be interpreted probabilistically, allowing for a small violation probability. Second, we need to further quantify the patterns and bounds of the wind uncertainty, and to define the largest wind disturbance a UAS can possibly reject. If wind is too large in a certain region, it should be handled in the planning stage.

Lastly, the safety controller presented in Chapter 6 shall be considered as the last effort to resolve collision after everything else has failed. In reality, collisions can be avoided well before they become safety threatening. This is especially true with the introduction of ADS-B. Another good approach is to provide structures, such as platooning on air highways [20], to the reduce the complexity of collision avoidance computations. In intersections, assigning priority could be another promising solution [19, 139]. In future UTM, a variety of safety controllers with different capabilities shall be integrated in a state machine, to ensure the

redundancy of the safety system.

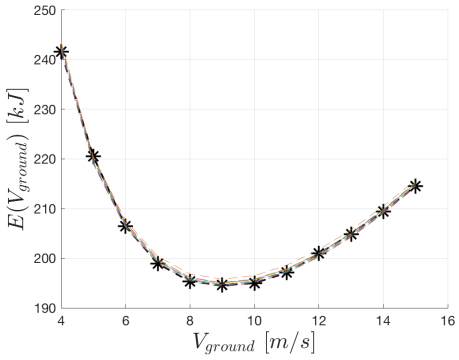
Part III

Appendix

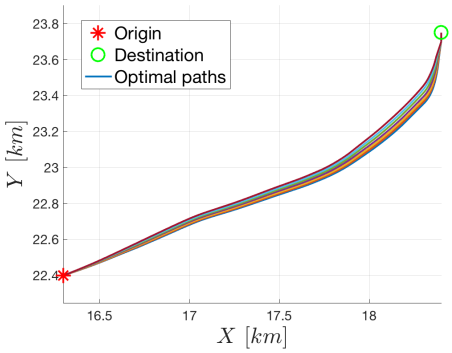
Appendix A

Energy Consumption vs. Trip Duration

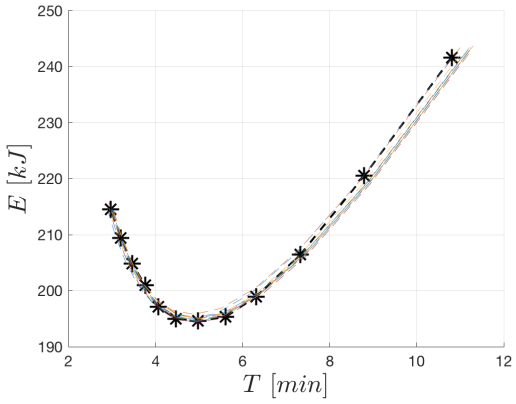
This appendix shows the additional simulation results in Section **??**.



(a) Optimal energy consumptions.

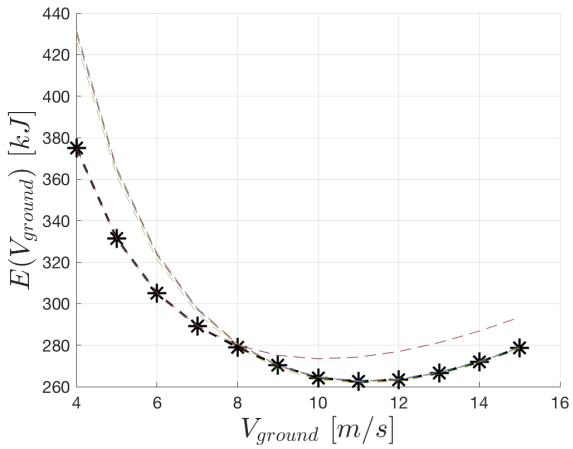


(b) The corresponding optimal paths.

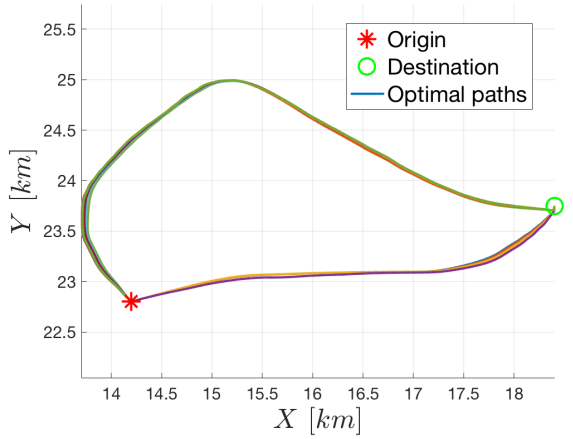


(c) The corresponding flight durations.

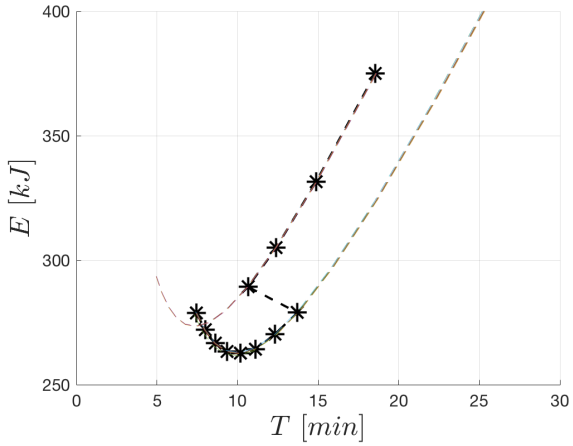
Figure A.1: Repeat the iteration process for a set of ground speeds for another origin.



(a) Optimal energy consumptions.

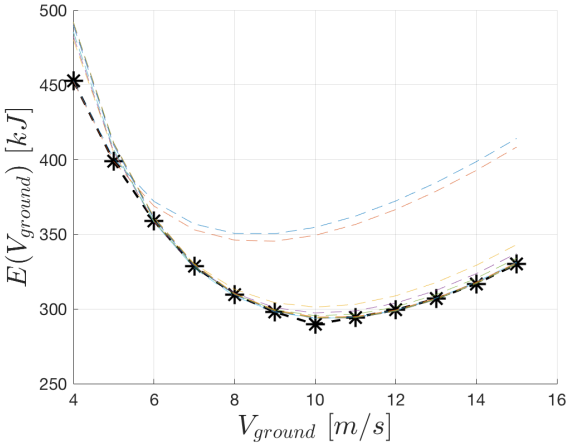


(b) The corresponding optimal paths.

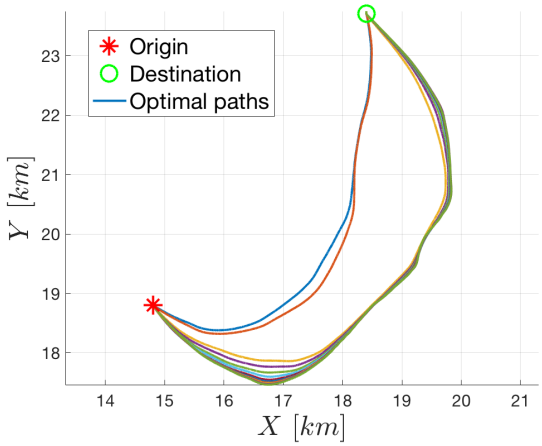


(c) The corresponding flight durations.

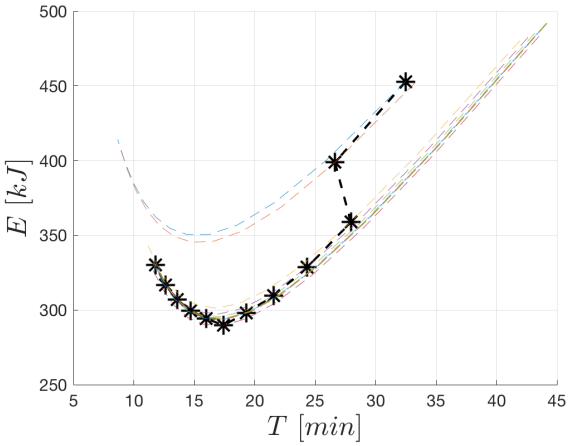
Figure A.2: Repeat the iteration process for a set of ground speeds for another origin.



(a) Optimal energy consumptions.



(b) The corresponding optimal paths.



(c) The corresponding flight durations.

Figure A.3: Repeat the iteration process for a set of ground speeds for another origin.

Appendix B

Exhaustive Search Results

This appendix shows the exhaustive search results in Section 2.6.7.

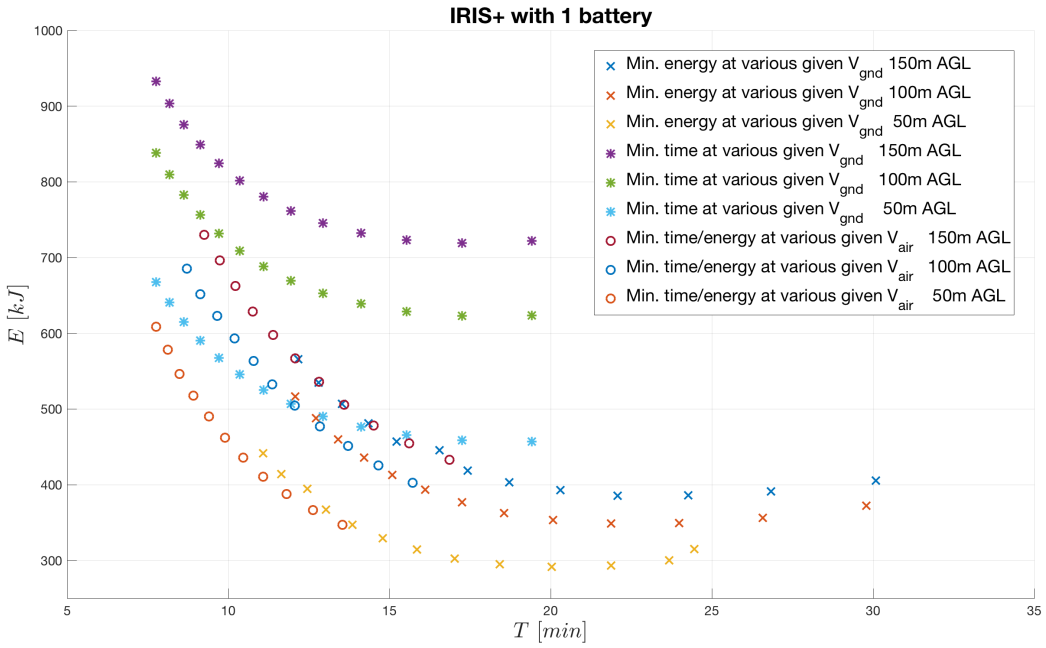


Figure B.1: IRIS+ with one battery: $14N$ self weight.

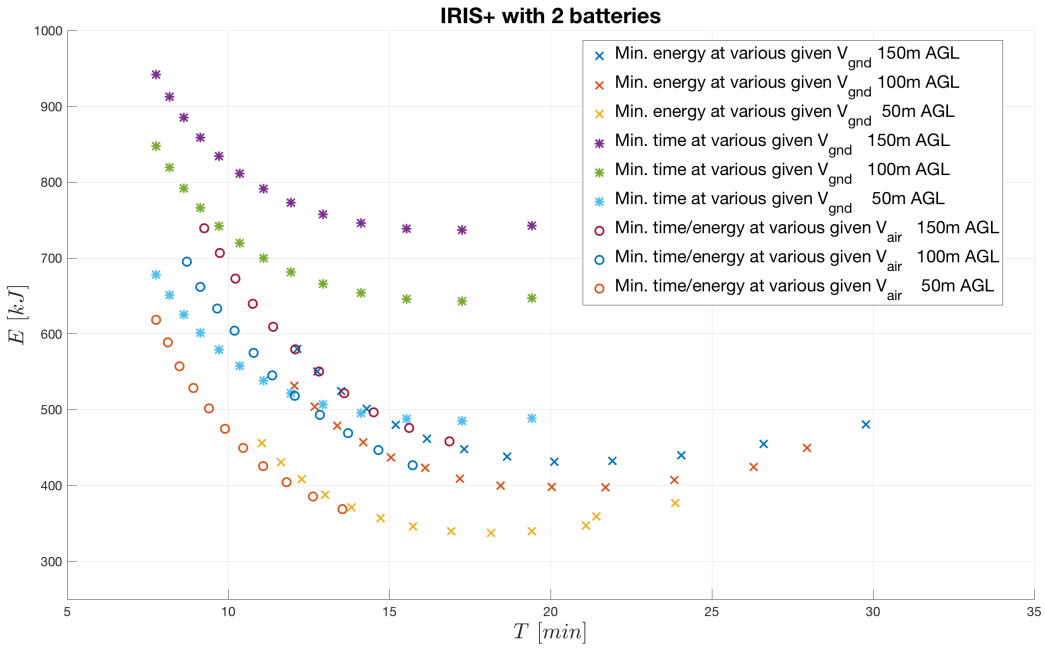


Figure B.2: IRIS+ with two battery: $17N$ self weight.

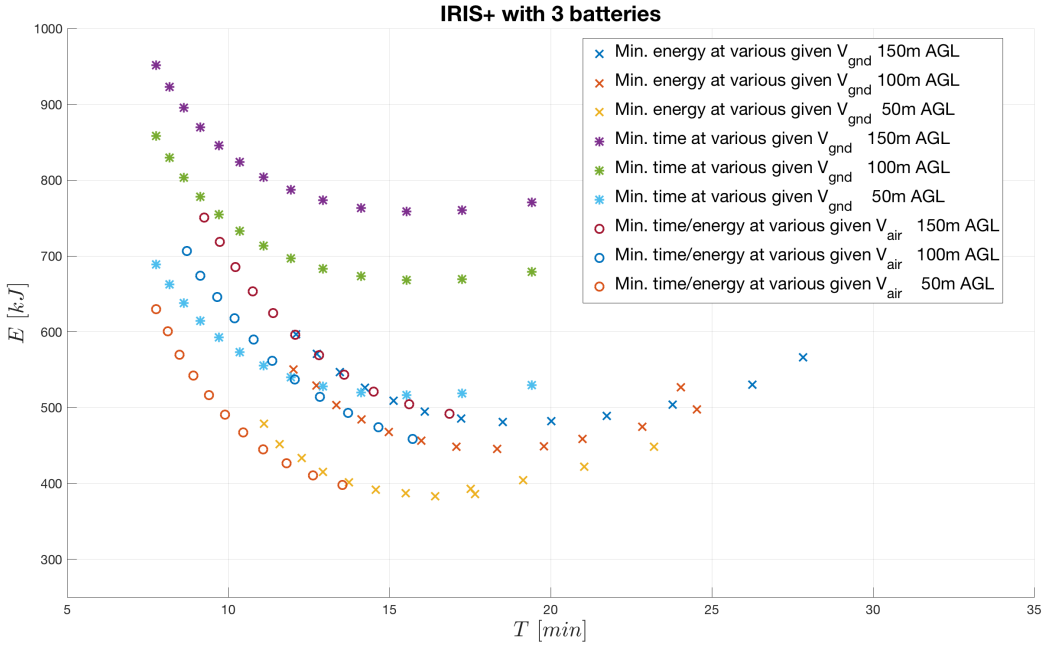


Figure B.3: IRIS+ with three battery: $20N$ self weight.

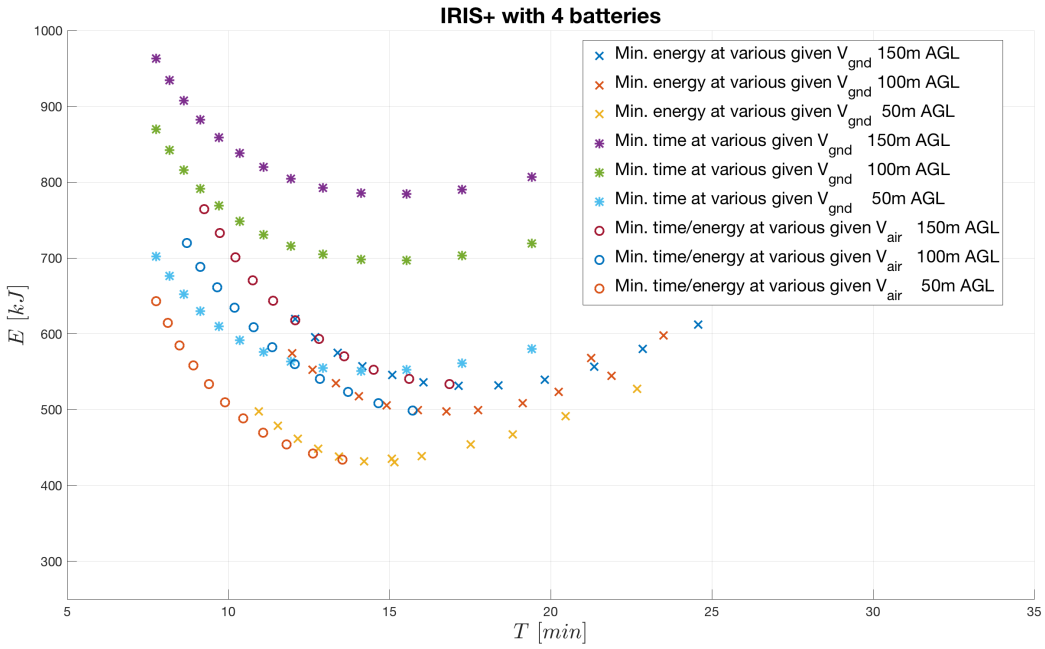


Figure B.4: IRIS+ with four battery: $23N$ self weight.

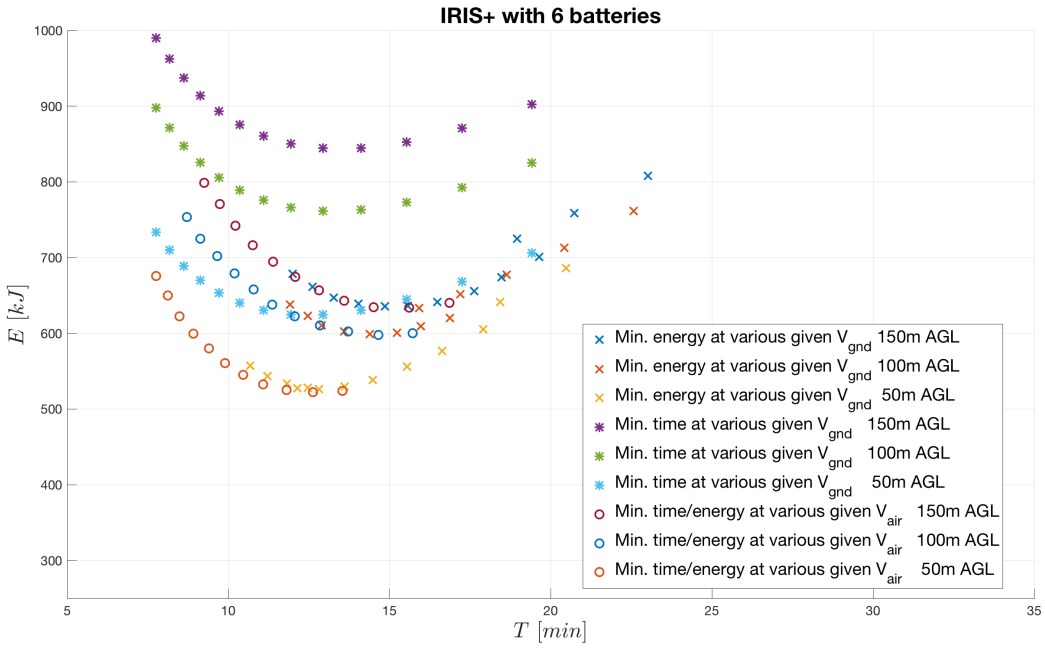


Figure B.5: IRIS+ with six battery: $29N$ self weight.

Appendix C

Derivation of Power Components

C.1 Induced Power

From energy conservation,

$$\begin{aligned} P_i = TV &= \frac{1}{2}\dot{m}V_\infty^2 - \frac{1}{2}\dot{m}V_0^2 \\ &= \frac{1}{2}\dot{m}(V_\infty - V_0)(V_\infty + V_0) \end{aligned} \tag{C.1}$$

From momentum conservation,

$$T = \dot{m}\,(V_\infty - V_0) \tag{C.2}$$

Substitute equation (C.2) into (C.1) and simplify, we have

$$V = \frac{1}{2}\,(V_\infty + V_0) \tag{C.3}$$

The mass flow rate is,

$$\dot{m} = \rho A V \tag{C.4}$$

Substitute (C.4) into (C.2), and then substitute (C.3), we get

$$T = \frac{1}{2}\rho A(V_\infty^2 - V_0^2) \tag{C.5}$$

Solve for V_∞, we have

$$V_\infty = \sqrt{\frac{2T}{\rho A} + V_0^2} \tag{C.6}$$

Therefore, by substituting (C.3) to (C.1) and then substituting (C.6), we obtain the analytical expression for the idealized induced power in equation (3.1).

Appendix D

Derivation of the HJB Equation

In this section, we present a derivation of (4.5). The process is similar to the one presented in [9].

We start from the cost function in equation (D.1). Note that because our cost is energy, there is no terminal cost, and only stage costs are considered.

$$J(\boldsymbol{x}, \boldsymbol{u}) = \int_{s_0}^{s_f} \underset{C}{g(\boldsymbol{x}(s), \boldsymbol{u}(s))ds} \tag{D.1}$$

Define the value function at $\boldsymbol{x}(s)$ to be

$$V(\boldsymbol{x}) = \min_{\boldsymbol{u} \in \mathcal{U}} \; J(\boldsymbol{x}, \boldsymbol{u}) \tag{D.2}$$

Then Bellman's optimality principle can be written as

$$\begin{aligned} V(\boldsymbol{x}(s)) &= \min_{\boldsymbol{u} \in \mathcal{U}} \; \{V(\boldsymbol{x}(s + \Delta s)) + \int_{s}^{s+\Delta s} g(\boldsymbol{x}(\bar{s}), \boldsymbol{u}(\bar{s}))d\bar{s})\} \\ V(\boldsymbol{x}(s_f)) &= 0 \end{aligned} \tag{D.3}$$

We introduce the following notation to discretize the state and control:

$$\begin{aligned} \boldsymbol{x}_k &:= \boldsymbol{x}(s_0 + k\Delta s) = \boldsymbol{x}(s) \\ \boldsymbol{u}_k &:= \boldsymbol{u}(s_0 + k\Delta s) = \boldsymbol{u}(s) \end{aligned} \tag{D.4}$$

Define the discrete approximation of the value function $V(\boldsymbol{x}(s))$ as $\tilde{V}(\boldsymbol{x}_k)$, we can re-write equation (D.3) as

$$0 = \min_{\boldsymbol{u}_k \in \mathcal{U}} \; \left\{ \frac{\tilde{V}(\boldsymbol{x}_{k+1}) - \tilde{V}(\boldsymbol{x}_k)}{\Delta s} + g(\boldsymbol{x}_k, \boldsymbol{u}_k) \right\} \tag{D.5}$$

Let $\Delta s \to 0$, then we have

$$0 = \min_{\boldsymbol{u}\in\mathcal{U}} \left\{ \frac{dV(\boldsymbol{x})}{ds} + g(\boldsymbol{x}, \boldsymbol{u}) \right\} \tag{D.6}$$

If we expand the derivative term $\frac{dV}{ds}$, and substitute the dynamic equation $\frac{d\boldsymbol{x}}{ds} = \boldsymbol{u}$ (equation (4.21b)), then equation (D.6) becomes

$$\begin{aligned} 0 &= \min_{\boldsymbol{u}\in\mathcal{U}} \left\{ \frac{\partial V}{\partial s} + \nabla_{\boldsymbol{x}} V(\boldsymbol{x})^T \frac{d\boldsymbol{x}}{ds} + g(\boldsymbol{x}, \boldsymbol{u}) \right\} \\ &= \min_{\boldsymbol{u}\in\mathcal{U}} \left\{ \frac{\partial V}{\partial s} + \nabla_{\boldsymbol{x}} V(\boldsymbol{x})^T \boldsymbol{u} + g(\boldsymbol{x}, \boldsymbol{u}) \right\} \\ &= \min_{\boldsymbol{u}\in\mathcal{U}} \left\{ \nabla_{\boldsymbol{x}} V(\boldsymbol{x})^T \boldsymbol{u} + g(\boldsymbol{x}, \boldsymbol{u}) \right\} \text{ (time invariant case)} \end{aligned} \tag{D.7}$$

The term $\frac{\partial V}{\partial s} = 0$ because the wind field is assumed time invariant and the parameter s is a parametrization of time, which results in equation (4.5).

Appendix E

Derivation of Avoid Set Boundary

Figure E.1 shows the trajectory from $(x_{12}(t_0), y_{12}(t_0))$ to $(x_{12}(0), y_{12}(0))$. Decompose the trajectory along and perpendicular to the direction of θ_1^* into d_1 and d_2.

Along the direction in d_1, the initial speed is $v_i = v_{12}(t_0)sin\theta_1^*$, the final speed is $v_f = 0$, and it is under constant acceleration $-a_{max}$. Along the direction in d_2, the vehicle is under uniform linear motion at constant speed $(v_{12}(t_0)cos\theta_1^*)$, and the time $t_0 = \left(\frac{v_{12}(t_0)sin\theta_1^*}{a_{max}}\right)$. Therefore, we have

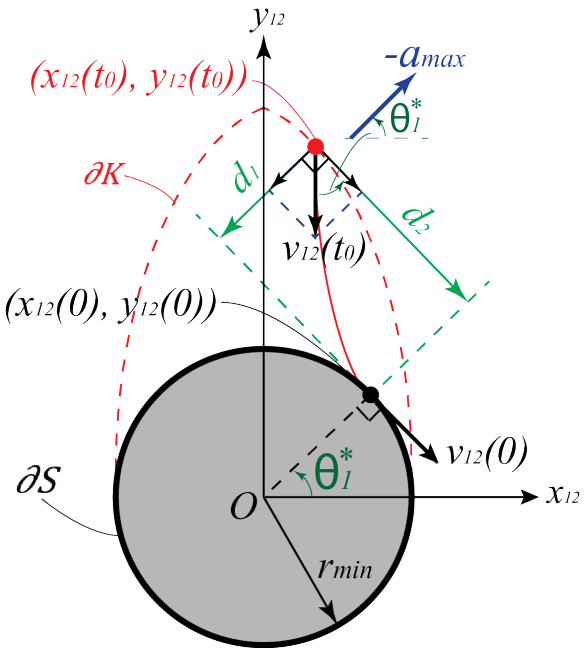


Figure E.1: Backward dynamics diagram.

$$d_1 = \frac{(v_{12}(t_0)sin\theta_1^*)^2}{a_{max}}$$
$$d_2 = (v_{12}(t_0)cos\theta_1^*)\left(\frac{v_{12}(t_0)sin\theta_1^*}{a_{max}}\right)$$

Lastly, by geometry,

$$x_{12}(t_0) - d_1 cos\theta_1^* + d_2 sin\theta_1^* = r_{min} cos\theta_1^*$$
$$y_{12}(t_0) - d_1 sin\theta_1^* - d_2 cos\theta_1^* = r_{min} sin\theta_1^*$$

which results in equation (6.5).

Bibliography

[1] *3D Robotics IRIS+*. (Date last accessed 14-October-2016). URL: http://3dr.com/support/articles/207358106/iris/.

[2] *3DR Solo Smart Drone*. (Date last accessed 01-April-2016). URL: http://3drobotics.com/solo-drone/.

[3] Victor Agubra and Jeffrey Fergus. "Lithium ion battery anode aging mechanisms". In: *Materials* 6.4 (2013), pp. 1310–1325.

[4] Mousumi Ahmed and Kamesh Subbarao. "Nonlinear 3-d trajectory guidance for unmanned aerial vehicles". In: *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*. IEEE. 2010, pp. 1923–1927.

[5] "Airworthiness Approval of Automatic Dependent Surveillance - Broadcst(ADS-B) Out Systems". In: AC 20-165A. Federal Aviation Administration, 2012.

[6] Dmitri Aleksandrov and Igor Penkov. "Energy consumption of mini UAV helicopters with different number of rotors". In: *11th International Symposium" Topical Problems in the Field of Electrical and Power Engineering*. 2012, pp. 259–262.

[7] *ART SYS 360*. (Date last accessed 01-April-2016). URL: http://www.artsys360.com/.

[8] Ferdinand P Beer and ER Johnston. *Statics and Dynamics*. 1972.

[9] Dimitri P Bertsekas et al. *Dynamic programming and optimal control*. Vol. 1. 2. Athena scientific Belmont, MA, 2017.

[10] John T Betts. "Survey of numerical methods for trajectory optimization". In: *Journal of Guidance control and dynamics* 21.2 (1998), pp. 193–207.

[11] John T Betts and Evin J Cramer. "Application of direct transcription to commercial aircraft trajectory optimization". In: *Journal of Guidance Control and Dynamics* 18 (1995), pp. 151–151.

[12] Samir Bouabdallah, Andre Noth, and Roland Siegwart. "PID vs LQ control techniques applied to an indoor micro quadrotor". In: *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. Vol. 3. IEEE. 2004, pp. 2451–2456.

[13] Samir Bouabdallah and Roland Siegwart. "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor". In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE. 2005, pp. 2247–2252.

[14] Hakim Bouadi et al. "Adaptive sliding mode control for quadrotor attitude stabilization and altitude tracking". In: *Computational Intelligence and Informatics (CINTI), 2011 IEEE 12th International Symposium on*. IEEE. 2011, pp. 449–455.

[15] M Bouchoucha et al. "Step by step robust nonlinear PI for attitude stabilisation of a four-rotor mini-aircraft". In: *Control and Automation, 2008 16th Mediterranean Conference on*. IEEE. 2008, pp. 1276–1283.

[16] Michel Broussely et al. "Main aging mechanisms in Li ion batteries". In: *Journal of power sources* 146.1-2 (2005), pp. 90–96.

[17] Arthur Earl Bryson. *Applied optimal control: optimization, estimation and control*. CRC Press, 1975.

[18] Jeffrey Byrne, Martin Cosgrove, and Raman Mehra. "Stereo based obstacle detection for an unmanned air vehicle". In: *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE. 2006, pp. 2830–2835.

[19] Mo Chen, Jennifer C Shih, and Claire J Tomlin. "Multi-vehicle collision avoidance via hamilton-jacobi reachability and mixed integer programming". In: *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE. 2016, pp. 1695–1700.

[20] Mo Chen et al. "Safe Platooning of Unmanned Aerial Vehicles via Reachability". In: *arXiv preprint arXiv:1503.07253* (2015).

[21] Fotini Katopodes Chow. "Subfilter-scale turbulence modeling for large-eddy simulation of the atmospheric boundary layer over complex terrain." In: (2004).

[22] Venanzio Cichella et al. "A Lyapunov-based approach for time-coordinated 3D path-following of multiple quadrotors". In: *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE. 2012, pp. 1776–1781.

[23] Gianpaolo Conte, Simone Duranti, and Torsten Merz. "Dynamic 3D path following for an autonomous helicopter". In: *Proc. of the IFAC Symp. on Intelligent Autonomous Vehicles*. 2004, pp. 5–7.

[24] Pascal Cornic et al. "Sense and avoid radar using Data Fusion with other sensors". In: *Aerospace Conference, 2011 IEEE*. IEEE. 2011, pp. 1–14.

[25] Navid Dadkhah and Bérénice Mettler. "Survey of motion planning literature in the presence of uncertainty: considerations for UAV guidance". In: *Journal of Intelligent & Robotic Systems* 65.1-4 (2012), pp. 233–246.

[26] Bogdan Dancila, Ruxandra Mihaela Botez, and Dominique Labour. "Altitude optimization algorithm for cruise, constant speed and level flight segments". In: *AIAA guidance, navigation, and control conference*. 2012, p. 4772.

[27] Kashif Dar et al. "Wireless communication technologies for ITS applications [topics in automotive networking]". In: *Communications Magazine, IEEE* 48.5 (2010), pp. 156–162.

[28] *Determining Safe Access with a Best Equipped, Best-Served Model for Small Unmanned Aircraft Systems*. (Date last accessed 01-April-2016). URL: `http://utm.arc.nasa.gov/docs/Amazon_Determining%20Safe%20Access%20with%20a%20Best-Equipped,%20Best-Served%20Model%20for%20sUAS[2].pdf`.

[29] Francis T Durso and Carol A Manning. "Air traffic control". In: *Reviews of human factors and ergonomics* 4.1 (2008), pp. 195–244.

[30] Warner L Ecklund, David A Carter, and Ben B Balsley. "A UHF wind profiler for the boundary layer: Brief description and initial results". In: *Journal of Atmospheric and Oceanic Technology* 5.3 (1988), pp. 432–441.

[31] Jack Elston and Eric W Frew. "Unmanned aircraft guidance for penetration of pretornadic storms". In: *Journal of guidance, control, and dynamics* 33.1 (2010), pp. 99–107.

[32] *Environmental Fluid Mechanics and Hydrology Group*. (Date last accessed 01-September-2017). URL: `http://efmh.berkeley.edu/efmhgroup/about.html`.

[33] M Exum et al. "Functional Requirements for NextGen TBO Separation Management Capabilities for En Route Operations". In: *MP100136R1, MITRE Center for Advanced Aviation System Development, McLean, VA* (2011).

[34] Dave Ferguson, Maxim Likhachev, and Anthony Stentz. "A guide to heuristic-based path planning". In: *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)*. 2005, pp. 9–18.

[35] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. "SVO: Fast semi-direct monocular visual odometry". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 15–22.

[36] Ralph Howard Fowler. *The elementary differential geometry of plane curves*. 20. University Press, 1920.

[37] Ran Y Gazit and J David Powell. "Aircraft collision avoidance based on GPS position broadcasts". In: *Digital Avionics Systems Conference, 1996., 15th AIAA/IEEE*. IEEE. 1996, pp. 393–399.

[38] Chad Goerzen, Zhaodan Kong, and Bernard Mettler. "A survey of motion planning algorithms from the perspective of autonomous UAV guidance". In: *Journal of Intelligent and Robotic Systems* 57.1-4 (2010), pp. 65–100.

[39] *Google UAS Airspace System Overview*. (Date last accessed 01-April-2016). URL: `http://utm.arc.nasa.gov/docs/GoogleUASAirspaceSystemOverview5pager[1].pdf`.

[40] FAA Handbook. "8083-21". In: *Rotorcraft Flying Handbook* (2000).

[41] Hongwen He, Rui Xiong, and Jinxin Fan. "Evaluation of lithium-ion battery equivalent circuit models for state of charge estimation by an experimental approach". In: *Energies* 4.4 (2011), pp. 582–598.

[42] Anthony J Healey and David Lienard. "Multivariable sliding mode control for autonomous diving and steering of unmanned underwater vehicles". In: *IEEE journal of Oceanic Engineering* 18.3 (1993), pp. 327–339.

[43] Albert Helfrick. *Principles of avionics*. Avionics Communications, 2010.

[44] Gabriel M Hoffmann and Claire J Tomlin. "Decentralized cooperative collision avoidance for acceleration constrained vehicles". In: *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE. 2008, pp. 4357–4363.

[45] Gabriel M Hoffmann et al. "Quadrotor helicopter flight dynamics and control: Theory and experiment". In: *Proc. of the AIAA Guidance, Navigation, and Control Conference*. Vol. 2. 2007, p. 4.

[46] *Hokuyo UTM-30LX*. (Date last accessed 01-April-2016). URL: `https://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html`.

[47] *How to Choose Motor and Propeller for Quadcopter and Multicopter*. URL: `http://blog.oscarliang.net/how-to-choose-motor-and-propeller-for-quadcopter/`.

[48] Stefan Hrabar. "3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs". In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE. 2008, pp. 807–814.

[49] Stefan Hrabar et al. "Combined optic-flow and stereo-based navigation of urban canyons for a UAV". In: *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE. 2005, pp. 3309–3316.

[50] Mu Huang et al. "Adaptive tracking control of underactuated quadrotor unmanned aerial vehicles via backstepping". In: *American Control Conference (ACC), 2010*. IEEE. 2010, pp. 2076–2081.

[51] Yong K Hwang and Narendra Ahuja. "A potential field approach to path planning". In: *IEEE Transactions on Robotics and Automation* 8.1 (1992), pp. 23–32.

[52] International Civil Aviation Organization, ICAO Engine Exhaust Emissions Data. Tech. rep. Doc 9646-AN/943, 2005.

[53] Stephen Jackson et al. "Tracking controllers for small UAVs with wind disturbances: Theory and flight results". In: *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE. 2008, pp. 564–569.

[54] Matthew R Jardin and Arthur E Bryson. "Neighboring optimal aircraft guidance in winds". In: *Journal of Guidance, Control, and Dynamics* 24.4 (2001), pp. 710–715.

[55] *Jeppesen Commercial Flight Planning and Dispatch*. (Date last accessed 26-Feburary-2017). URL: `http://ww1.jeppesen.com/personal-solutions/aviation/general-aviation-flight-planning.jsp`.

[56] Tony Jimenez. "The Wind Powering America Anemometer Loan Program: A Retrospective". In: *Contract* 303 (2013), pp. 275–3000.

[57] Wayne Johnson. *Helicopter theory*. Courier Corporation, 2012.

[58] *Julia Language*. (Date last accessed 14-October-2016). URL: `http://julialang.org/`.

[59] Hassan K Khalil. *Nonlinear control*. Prentice Hall, 2014.

[60] Kwang-Yeon Kim, Jung-Woo Park, and Min-Jea Tahk. "UAV collision avoidance using probabilistic method in 3-D". In: *Control, Automation and Systems, 2007. ICCAS'07. International Conference on*. IEEE. 2007, pp. 826–829.

[61] Ralph D Kimberlin. *Flight testing of fixed wing aircraft*. Aiaa, 2003.

[62] Georg Klein and David Murray. "Parallel tracking and mapping for small AR workspaces". In: *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE. 2007, pp. 225–234.

[63] Parimal Kopardekar. "Safely Enabling Low-Altitude Airspace Operations: Unmanned Aerial System Traffic Management (UTM)". In: (2015).

[64] Parimal H Kopardekar. "Unmanned Aerial System (UAS) Traffic Management (UTM): Enabling Low-Altitude Airspace and UAS Operations". In: (2014).

[65] Bernd Korn and Christiane Edinger. "UAS in civil airspace: Demonstrating sense and avoid capabilities in flight trials". In: *Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th*. IEEE. 2008, pp. 4–D.

[66] Jimmy Krozel, Tysen Mueller, and George Hunter. "Free flight conflict detection and resolution analysis". In: *AIAA Guidance, Navigation, and Control Conf., Paper*. 1996, pp. 96–3763.

[67] Daewon Lee, H Jin Kim, and Shankar Sastry. "Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter". In: *International Journal of control, Automation and systems* 7.3 (2009), pp. 419–428.

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

[69] Guohua Li et al. "Factors associated with pilot error in aviation crashes". In: *Aviation, space, and environmental medicine* 72.1 (2001), pp. 52–58.

[70] Jun Li and Yuntang Li. "Dynamic analysis and PID control for a quadrotor". In: *Mechatronics and Automation (ICMA), 2011 International Conference on*. IEEE. 2011, pp. 573–578.

[71] Bor Yann Liaw et al. "Modeling capacity fade in lithium-ion cells". In: *Journal of power sources* 140.1 (2005), pp. 157–161.

[72] John Lygeros, Claire Tomlin, and Shankar Sastry. "Controllers for reachability specifications for hybrid systems". In: *Automatica* 35.3 (1999), pp. 349–370.

[73] Tarek Madani and Abdelaziz Benallegue. "Backstepping control for a quadrotor helicopter". In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE. 2006, pp. 3255–3260.

[74] Tarek Madani and Abdelaziz Benallegue. "Backstepping sliding mode control applied to a miniature quadrotor flying robot". In: *IEEE Industrial Electronics, IECON 2006-32nd Annual Conference on*. IEEE. 2006, pp. 700–705.

[75] Tarek Madani and Abdelaziz Benallegue. "Backstepping sliding mode control applied to a miniature quadrotor flying robot". In: *IEEE Industrial Electronics, IECON 2006-32nd Annual Conference on*. IEEE. 2006, pp. 700–705.

[76] *MATLAB*. URL: `http://www.mathworks.com/products/matlab/`.

[77] David Francis Maune. *Digital elevation model technologies and applications: the DEM users manual*. Asprs Publications, 2007.

[78] Daniel Mellinger, Alex Kushleyev, and Vijay Kumar. "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 477–483.

[79] PK Menon, GD Sweriduk, and B Sridhar. "Optimal strategies for free-flight air traffic conflict resolution". In: *Journal of Guidance, Control, and Dynamics* 22.2 (1999), pp. 202–211.

[80] *Mission Planner from Ardupilot (Open Source)*. (Date last accessed 26-Feburary-2017). URL: `http://ardupilot.org/planner/docs/mission-planner-overview.html`.

[81] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games". In: *Automatic Control, IEEE Transactions on* 50.7 (2005), pp. 947–957.

[82] Ikbal Chammakhi Msadaa, Pasquale Cataldi, and Fethi Filali. "A comparative study between 802.11 p and mobile WiMAX-based V2I communication networks". In: *Next Generation Mobile Applications, Services and Technologies (NGMAST), 2010 Fourth International Conference on*. IEEE. 2010, pp. 186–191.

[83] Yash Mulgaonkar et al. "Power and weight considerations in small, agile quadrotors". In: *SPIE Defense+ Security*. International Society for Optics and Photonics. 2014, 90831Q–90831Q.

[84] Bruce Roy Munson, Donald F Young, and Theodore Hisao Okiishi. *Fundamentals of fluid mechanics*. New York, 1990.

[85] *NASA DEM Models*. (Date last accessed 14-October-2016). URL: https://lpdaac.usgs.gov/dataset_discovery/aster/aster_products_table/ast14dem.

[86] Derek R Nelson et al. "Vector field path following for miniature air vehicles". In: *IEEE Transactions on Robotics* 23.3 (2007), pp. 519–529.

[87] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. "DTAM: Dense tracking and mapping in real-time". In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2320–2327.

[88] Michael Nolan. *Fundamentals of air traffic control*. Cengage learning, 2010.

[89] Timothy R Oke. *Boundary layer climates*. Routledge, 2002.

[90] Frauke Oldewurtel. "Stochastic model predictive control for energy efficient building climate control". PhD thesis. 2011.

[91] Alberto Olivares, Manuel Soler, and Ernesto Staffetti. "Multiphase mixed-integer optimal control applied to 4d trajectory planning in air traffic management". In: *Proceedings of the 3rd International Conference on Application and Theory of Automation in Command and Control Systems*. ACM. 2013, pp. 85–94.

[92] "Operation and Certification of Small Unmanned Aircraft Systems". In: 14 CFR Parts 21, 43, 45, 47, 61, 91, 101, 107, and 183. Federal Aviation Administration, 2015.

[93] Douglas M Pargett and Mark D Ardema. "Flight path optimization at constant altitude". In: *Journal of guidance, control, and dynamics* 30.4 (2007), pp. 1197–1201.

[94] Jung-Woo Park, Hyon-Dong Oh, and Min-Jea Tahk. "Uav collision avoidance based on geometric approach". In: *SICE Annual Conference, 2008*. IEEE. 2008, pp. 2122–2126.

[95] Sanghyuk Park, John Deyst, and Jonathan P How. "Performance and Lyapunov stability of a nonlinear path-following guidance method". In: *Journal of Guidance Control and Dynamics* 30.6 (2007), pp. 1718–1728.

[96] Duc-Kien Phung and Pascal Morin. "Modeling and energy evaluation of small convertible UAVs". In: *IFAC Proceedings Volumes* 46.30 (2013), pp. 212–219.

[97] *PING2020 uAvioni*. URL: http://www.uavionix.com/products/ping2020/.

[98] *PointGrey Bumblebee XB3*. (Date last accessed 01-April-2016). URL: https://www.ptgrey.com/support/downloads/10132.

[99] William H Press et al. *Numerical recipes in C*. Vol. 2. Cambridge university press Cambridge, 1996.

[100] FM Ralph et al. "Doppler sodar and radar wind-profiler observations of gravity-wave activity associated with a gravity current". In: *Monthly weather review* 121.2 (1993), pp. 444–463.

[101] Ashwini Ratnoo, PB Sujit, and Mangal Kothari. "Adaptive optimal path following for high wind flights". In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 12985–12990.

[102] *Revising the Airspace Model for the Safe Integration of Small Unmanned Aircraft Systems*. (Date last accessed 01-April-2016). URL: `http://utm.arc.nasa.gov/docs/Amazon_Revising%20the%20Airspace%20Model%20for%20the%20Safe%20Integration%20of%20sUAS[6].pdf`.

[103] Ihnseok Rhee, Sanghyuk Park, and Chang-Kyung Ryoo. "A tight path following algorithm of an UAS based on PID control". In: *SICE Annual Conference 2010, Proceedings of*. IEEE. 2010, pp. 1270–1273.

[104] *Richmond Field Station, UC Berkeley*. (Date last accessed 26-Feburary-2017). URL: `http://rfs-env.berkeley.edu/index.html`.

[105] James F Roberts, Jean-Christophe Zufferey, and Dario Floreano. "Energy management for indoor hovering robots". In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE. 2008, pp. 1242–1247.

[106] Gonzalo R Rodríguez-Canosa et al. "A real-time method to detect and track moving objects (DATMO) from unmanned aerial vehicles (UAVs) using a single camera". In: *Remote Sensing* 4.4 (2012), pp. 1090–1111.

[107] Ashton Roza and Manfredi Maggiore. "Path following controller for a quadrotor helicopter". In: *American Control Conference (ACC), 2012*. IEEE. 2012, pp. 4655–4660.

[108] Martin Russ et al. "LIDAR-based object detection on small UAV: Integration, Experimentation and Results". In: *AIAA Infotech Conference*. 2012.

[109] Megan S Ryerson, Mark Hansen, and James Bonn. "Fuel consumption and operational performance". In: *Transportation Research* Part D (2011), pp. 305–314.

[110] Megan S Ryerson et al. "Landing on empty: estimating the benefits from reducing fuel uplift in US Civil Aviation". In: *Environmental Research Letters* 10.9 (2015), p. 094002.

[111] *Sagetech*. URL: `http://www.sagetechcorp.com/`.

[112] L Rodriguez Salazar et al. "A Novel System for Non-Cooperative UAV Sense-And-Avoid". In: *proceedings of European Navigation Conference*. 2013.

[113] Hermann Schlichting. "Boundary-layer theory". In: (1968).

[114] Tom Schouwenaars et al. "Mixed integer programming for multi-vehicle path planning". In: *European control conference*. Vol. 1. 2001, pp. 2603–2608.

[115] James A Sethian. "Fast marching methods". In: *SIAM review* 41.2 (1999), pp. 199–235.

[116] James A Sethian and Alexander Vladimirsky. "Ordered upwind methods for static Hamilton–Jacobi equations: Theory and algorithms". In: *SIAM Journal on Numerical Analysis* 41.1 (2003), pp. 325–363.

[117] Alex Shum, Kirsten Morris, and Amir Khajepour. "Convergence rate for the ordered upwind method". In: *Journal of Scientific Computing* 68.3 (2016), pp. 889–913.

[118] Jorge Estrela da Silva and Joao Borges de Sousa. "A dynamic programming approach for the motion control of autonomous vehicles". In: *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE. 2010, pp. 6660–6665.

[119] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*. Vol. 60. Prentice-Hall Englewood Cliffs, NJ, 1991.

[120] Tracy Lorraine Smith et al. "11.1 CONVECTION FORECASTS FROM THE HOURLY UPDATED, 3-KM HIGH RESOLUTION RAPID REFRESH (HRRR) MODEL". In: (2008).

[121] Tracy Lorraine Smith et al. "Convection forecasts from the hourly updated, 3-km High Resolution Rapid Refresh Model". In: *Preprints, 24th Conf. on Severe Local Storms, Savannah, GA, Amer. Meteor. Soc*. Vol. 11. 2008.

[122] Manuel Soler. *Commercial aircraft trajectory planning based on multiphase mixed-integer optimal control*. Omm Editorial, 2013.

[123] Bongsob Song and J Karl Hedrick. *Dynamic surface control of uncertain nonlinear systems: an LMI approach*. Springer Science & Business Media, 2011.

[124] Banavar Sridhar, Hok K Ng, and Neil Y Chen. "Aircraft trajectory optimization and contrails avoidance in the presence of winds". In: *Journal of Guidance, Control and Dynamics* 34.5 (2011), pp. 1577–1583.

[125] PB Sujit, Srikanth Saripalli, and Joao Borges Sousa. "Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless". In: *Control Systems, IEEE* 34.1 (2014), pp. 42–59.

[126] Ashit Talukder and Larry Matthies. "Real-time detection of moving objects from moving vehicles using dense stereo and optical flow". In: *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. Vol. 4. IEEE. 2004, pp. 3718–3725.

[127] Takafumi Taniguchi, Luka Eciolaza, and Michio Sugeno. "Quadrotor control using dynamic feedback linearization based on piecewise bilinear models". In: *Computational Intelligence in Control and Automation (CICA), 2014 IEEE Symposium on*. IEEE. 2014, pp. 1–7.

[128] Teodor Tomić et al. "Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue". In: *Robotics & Automation Magazine, IEEE* 19.3 (2012), pp. 46–56.

[129] Yen-Hsi Richard Tsai et al. "Fast sweeping algorithms for a class of Hamilton–Jacobi equations". In: *SIAM journal on numerical analysis* 41.2 (2003), pp. 673–694.

[130] *Tutorial Lithium Ion Battery Model*. (Date last accessed 15-November-2017). URL: `https://powersimtech.com/drive/uploads/2016/12/Tutorial-Lithium-Ion-Battery-Model.pdf`.

[131] *UAS Flight Planning System*. (Date last accessed 28-October-2017). URL: `https://lzl200102109@bitbucket.org/lzl200102109/flightplanning.git`.

[132] *UAS Flight Planning System*. (Date last accessed 28-October-2017). URL: `https://lzl200102109@bitbucket.org/lzl200102109/powerconsumptionmodel.git`.

[133] *Using an Airspeed Sensor*. (Date last accessed 24-July-2017). URL: `http://ardupilot.org/plane/docs/airspeed.html`.

[134] Michael P Vitus, Steven L Waslander, and Claire J Tomlin. "Locally optimal decomposition for autonomous obstacle avoidance with the tunnel-MILP algorithm". In: *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE. 2008, pp. 540–545.

[135] *Wind Decision Support for UAS Operations, NASA Weather Workshop*. (Date last accessed 14-October-2016). URL: `http://aviationsystems.arc.nasa.gov/utm-weather/presentations/9_2016_Evans_UTM%20Weather_Workshop_V3.1.pdf`.

[136] Michael G Wing, Aaron Eklund, and Loren D Kellogg. "Consumer-grade global positioning system (GPS) accuracy and reliability". In: *Journal of forestry* 103.4 (2005), pp. 169–173.

[137] Rong Xu and Ümit Özgüner. "Sliding mode control of a quadrotor helicopter". In: *Decision and Control, 2006 45th IEEE Conference on*. IEEE. 2006, pp. 4957–4962.

[138] Ernst Zermelo. "Über die Navigation in der Luft als Problem der Variationsrechnung". In: *Jahresbericht der deutschen Mathematiker–Vereinigung, Angelegenheiten* 39 (1930), pp. 44–48.

[139] Yue J Zhang, Andreas A Malikopoulos, and Christos G Cassandras. "Optimal control and coordination of connected and automated vehicles at urban traffic intersections". In: *American Control Conference (ACC), 2016*. IEEE. 2016, pp. 6227–6232.