Maneuver Design and Motion Planning for Agile Fixed-Wing UAVs

Joshua M. Levin

The Department of Mechanical Engineering
McGill University, Montreal

April 2019

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of Doctor of Philosophy.

Abstract

In recognition of their broad scope of utility, recent years have seen a surge of interest in unmanned aerial vehicles (UAVs). As a result of technological advancements, UAVs have been rapidly expanding into the civilian marketplace. Many of the jobs UAVs have the potential to fill demand high levels of autonomy, and thus the state-of-the-art is constantly being pushed forward. Many such jobs require near-ground autonomous flight through obstacle-dense environments. Achieving proficiency in this regard requires agile motion, precise tracking performance, and efficient real-time planning.

Many non-traditional UAV platforms have been designed to suit various applications. Agile fixed-wing UAVs represent one such class of vehicles. They are characterized by their high thrust-to-weight ratio, large control surfaces, low aspect ratios, and a powerful propeller slipstream (also known as *propwash*). While they were originally marketed towards remote control pilots, their design makes them inherently valuable for autonomous flight. The primary appeal of the design is that it allows for both efficient fixed-wing forward flying *and* agile maneuvering, e.g. stopping mid-flight. In this way, these UAVs begin to bridge the gap in utility between efficient fixed-wing vehicles, and agile rotorcraft.

The broad objective of this thesis is to exploit the full maneuvering capabilities of agile fixed-wing UAVs for autonomous flight. The main topics covered are maneuver design, control, and motion planning. The thesis begins with a discussion of preliminary topics: an aircraft dynamics model, a feedback controller, and an optimization framework, all of which are utilized throughout the following sections of the thesis. Next, an investigation is performed to evaluate the significance of sideslip and propeller slipstream in extreme maneuvering with fixed-wing UAVs. We identify the cost, in terms of performance loss, if either of these two phenomena are not accounted for in maneuver design.

In the following chapter, we propose a strategy for designing and controlling agile maneuvers that takes advantage of the aircraft's full flight envelope. Optimal and dynamically feasible trajectories are generated, along with their associated feedforward control laws. Combining the transient agile maneuvers with steady-state trim conditions, we formulate a maneuver space, i.e. a library of trajectories. The maneuver space acts as a hybrid representation of the vehicle's dynamics, and as such is useful for efficient real-time motion planning. This chapter also includes a description of a heuristic for transitioning between maneuvers, and a methodology for continuously parametrizing agile maneuvers.

As a natural progression towards the ultimate goal of the thesis, the maneuver space is integrated into a real-time motion planner based on the Rapidly-Exploring Random Trees (RRT) algorithm. The planner is used to address the problem of generating a dynamically feasible motion plan to guide the aircraft to a desired goal through a highly-constrained, three-dimensional environment. For the purposes of this thesis, the environment is assumed to be known, and to only contain static obstacles. The planning framework is able to exploit the aircraft's full maneuvering capabilities, and couples well with a control system for effective trajectory tracking.

To conclude the main body of the thesis, simulation and flight test results are presented and discussed. The flight tests are performed in two sets. First, to perform a number of agile maneuvers, in isolation and in series. The experiments validate the feasibility of the maneuvers, and test the efficacy of the proposed control system. The second set of tests validate the real-time motion planner. All experiments, including the real-time motion planning, are implemented using only the sensors and computers mounted on-board the UAV.

Résumé

Les drones existent depuis des décennies. Avec les développements technologiques récent, ils commencent à se développer sur le marché civil. Un grand nombre d'applications des drones demande des niveaux d'automatisations très élevé, donc la recherche technologique avancée est nécessaire. Voler de manière autonome dans des environnements encombrés d'obstacles nécessite, entre autres, de l'agilité, un repérage précis et une planification en temps réel.

De nombreux modèles de drone non traditionnels ont été développé pour répondre à plusieurs types d'applications. Les drones agiles à voilure fixe représentent une de ces catégories de véhicules. Ils se caractérisent par leur rapport de poussée-poids élevé, leurs grandes surfaces de contrôle, leurs rapports d'aspect bas et leur soufflé d'hélices puissantes. Originalement conçus pour les pilotes télécommandés, leurs caractéristiques les rends très utiles pour les vols autonomes. L'attrait principal réside dans le fait qu'ils sont capables d'un vol avant efficace et des manœuvres agiles (ex : s'arrêter en plein vol). De cette manière, ils commencent à combler le fossé entre les véhicules à ailes fixes efficaces et les giravions agiles.

L'objectif général de cette thèse est d'exploiter toutes les capacités de manœuvre des drones agiles à voilure fixe, pour les vols autonomes. Les sujets principaux sont la conception des manœuvres, le contrôle et la planification des mouvements. La thèse commence avec une discussion sur les sujets préliminaires: un modèle dynamique d'aéronef, un contrôleur de rétroaction et une structure d'optimisation, qui sont tous utilisés dans les sections suivantes de la thèse. Ensuite, une enquête est menée pour évaluer l'importance du glissement latéral et du courant de l'hélice dans les manœuvres extrêmes avec des drones à voilure fixe. Nous identifions le coût, en termes de perte de performance, si l'un ou l'autre de ces deux phénomènes n'est pas pris en compte dans la conception de la manœuvre.

Au chapitre suivant, nous proposons une stratégie de conception et de contrôle pour les manœuvres agiles qui profite des caractéristiques de vol de l'avion. Des trajectoires optimales et réalisables de manière dynamique sont générées, ainsi que leurs commandes de contrôle prédites. En combinant les manœuvres agiles avec les conditions aéronef, nous formulons un espace de manœuvre - une bibliothèque de trajectoires. L'espace de manœuvre agit comme une représentation de la dynamique du véhicule, qui devient utile pour une planification efficace du mouvement en temps réel. Ce chapitre comprend également des descriptions d'une heuristique de transition entre manœuvres et d'une méthodologie de paramétrage continu des manœuvres agiles.

En tant que progression naturelle vers le but ultime de la thèse, l'espace de manœuvre est intégré dans un planificateur de mouvement en temps réel basé sur RRT. Le planificateur est utilisé pour créer un plan de mouvement réalisable qui guidera l'avion vers l'objectif désiré, dans un environnement tridimensionnel. Dans cette thèse, l'environnement est présumé être connu et contient seulement des obstacles statiques. La structure de planification est capable d'exploiter toutes les capacités de manœuvre de l'aéronef et est couplé à un système de contrôle pour un repérage efficace de trajectoire.

Pour conclure le corps principal de la thèse, des résultats de simulation et de test en vol sont présentés et discutés. Les tests de vol sont effectués en deux séries. Premièrement, en effectuant des manœuvres agiles en isolation et en série. Ces expériences permettent de valider la réalisation des manœuvres et de tester l'efficacité du système de contrôle proposé. La deuxième série de tests valide le planificateur de mouvement en temps réel. Toutes les expériences, y compris la planification de mouvement en temps réel, sont mises en œuvre en utilisant uniquement les capteurs et les ordinateurs montés à bord du drone.

Acknowledgements

First and foremost, I wish to extend my immense gratitude to my supervisors, Professor Meyer Nahon and Professor Aditya Paranjape. I could not have asked for better mentors to guide me through my graduate studies. While directing my work down productive avenues, they also allowed me the freedom to develop and flourish as a researcher. I would additionally like to thank them for their generosity and attentiveness throughout the years; even though the three of us were rarely in the same country at once, it was never a chore to get feedback on my work or to arrange meetings.

Next, I would like to express my gratitude to my many friends and colleagues in the McGill Aerospace Mechatronics Lab. The days flew by working and collaborating with Eitan Bulka on agile fixed-wing UAV projects. Many of our discussions helped refine my work, while others prompted me down new research paths. I was extremely fortunate to be surrounded by many other friends and colleagues who made the lab a truly fruitful and enjoyable place to work. I owe many thanks to my generous labmates who gladly offered their assistance in performing flight tests: Mikkel Jørgensen, Juan Carlos Hernàndez Ramirez, Sahand Rezaei, Fares El Tin, and Romain Chiappinelli. I want to give special thanks to Juan for his persistent generosity in helping me overcome technical issues and clarifying my thoughts about control systems. It was also a pleasure to share an office with Luc Sagnières, Fiona Chui, and Gareth Dicker.

Special thanks go to my parents, Ellen and Lewis Levin, who have always enabled, supported, and encouraged the furthering of my education. Their unconditional love and support has enabled me to pursue my dreams. Despite having absolutely no expertise in the field, my dad has always been the first and most scrupulous reviewer of almost everything I have ever written, including my publications.

Finally, to Katherine McDonald, thank you for sticking with me through this all. Your support never wavered over the four years of long-distance brought on by my decision to pursue a Ph.D. degree. Words cannot express how thankful I am for the ways in which you have prioritized my ambitions in the toughest decisions we have had to make throughout our lives. As wonderful as it was to study at McGill and live in Montreal, I am utterly grateful to now be living in the same city, under the same roof. I would be remiss not to mention here my studious study buddy, Yoda, who was wonderfully entertaining company while writing this thesis.

The work in this thesis was made possible with the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC) Postgraduate Scholarship,

and the Les Vadasz Fellowship received through the McGill Engineering Doctoral Award (MEDA).

Claims of Originality

The main contributions of this thesis are listed below, and will be revisited in more detail in the conclusion:

- An investigation into the role of sideslip and propeller slipstream in the extreme maneuver capabilities of agile fixed-wing UAVs is performed. The cost, in terms of performance loss, is identified if either of the two phenomena is not accounted for in maneuver design.
- A maneuver space for agile fixed-wing UAVs is developed, integrating steady-state trim conditions and transient agile maneuvers, which are dynamically feasible and exploit the full extent of the aircraft's flight envelope.
- A novel method for trajectory parametrization is proposed, making use of dynamic time warping. The parametrization makes the agile maneuver space more robust while incurring a negligible cost to the computational load needed in flight.
- The maneuver space is integrated as a library into a real-time, RRT-based motion planning algorithm. The maneuver space enables the planner to exploit the aircraft's full flight envelope, and ensures the trajectories it generates are dynamically feasible. The trajectory solutions are composed of the aircraft's full 12-state vector, as well as all of its control inputs. As a feature of the way in which the maneuver space is integrated into the planning algorithm, the size of the library has no effect on planning time; typically, the computational cost of RRT increases with the number of primitives. Furthermore, the implementation places no constraints on the sequencing of primitives.
- Flight tests demonstrations are performed for executing agile maneuvers, and for real-time motion planning. The tests make use of only on-board sensing and computing equipment; no external sensors or ground computers are used for autonomous flight.

Large parts of this thesis have appeared in the following publications:

Joshua M Levin, Meyer Nahon, and Aditya A Paranjape. Aggressive turn-around manoeuvres with an agile fixed-wing UAV. In *20th IFAC Symposium on Automatic Control in Aerospace*, pages 242–247. Elsevier, 2016. doi: 10.1016/j.ifacol.2016.09.042

Joshua M Levin, Aditya Paranjape, and Meyer Nahon. Agile fixed-wing UAV motion planning with knife-edge maneuvers. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 114–123. IEEE, 2017. doi: 10.1109/

icuas.2017.7991475

Joshua M Levin, Aditya A Paranjape, and Meyer Nahon. Sideslip and slipstream in extreme maneuvering with fixed-wing unmanned aerial vehicles. *Journal of Guidance, Control, and Dynamics*, 41(7):1610–1616, May 2018. doi: 10.2514/1.G003086

Joshua M Levin, Aditya A Paranjape, and Meyer Nahon. Agile maneuvering with a small fixed-wing unmanned aerial vehicle. *Robotics and Autonomous Systems*, 116: 148–161, March 2019. doi: 10.1016/j.robot.2019.03.004

Joshua M Levin, Aditya A Paranjape, and Meyer Nahon. Motion planning for a small aerobatic fixed-wing unmanned aerial vehicle. In *The International Conference on Intelligent Robots and Systems (IROS)*, pages 8464–8470. IEEE, 2018. doi: 10.1109/IROS.2018.8593670

Joshua M Levin, Meyer Nahon, and Aditya A Paranjape. Real-time motion planning with a fixed-wing UAV using an agile maneuver space. In review with Autonomous Robots

Notation

Abbreviations

AML	The McGill Aerospace Mechatronics Lab
ATA	Aggressive Turn-Around
CTH	Cruise-to-Hover Transition
DTW	Dynamic Time Warping
EKF	Extended Kalman Filter
ESC	Electronic Speed Controller
HIL	Hardware-in-the-Loop
HTC	Hover-to-Cruise Transition
IMU	Inertial Measurement Unit
MEX	C MATLAB Executable File
NLP	Nonlinear Programming Problem
PWM	Pulse Width Modulation
QGC	QGroundControl
RC	Remote Control
RMSE	Root-Mean-Square Error
RRT	Rapidly-Exploring Random Trees
UAV	Unmanned Aerial Vehicle

Symbols

A_{prop}	Propeller disk area
b	Wing span
$\bar{c}$	Mean aerodynamic chord
$\mathbf{C}$	Rotation matrix

$C_{l_{\delta_a}}, C_{l_{\delta_r}}, C_{m_{\delta_e}}, C_{n_{\delta_r}}$	Control derivative coefficients
$\mathbf{E}$	Vector of angular errors about the body frame
$\mathbf{F}$	Force vector
$\mathbf{g}$	Gravitational acceleration vector
$\mathbf{I}$	Inertia matrix about the centre of gravity and resolved in the body frame
J	Optimal control cost function
K	Feedback gain
L, M, N	Moments about the body frame
m	Mass
$\mathbf{M}$	Moment vector
p, q, r	Angular velocity components in the body frame
$\mathbf{q}$	Quaternion vector
$\mathbf{r}$	Position vector
S	Wing area
t	Time
T	Thrust
u, v, w	Velocity components in the body frame
$\mathbf{u}$	Control input vector
$\mathbf{V}$	Velocity vector
$V_{i0,avg}$	Momentum-averaged induced velocity
$\mathbf{V}_s(x, r)$	Slipstream airflow velocity field
x, y, z	Position components in the inertial frame
$\mathbf{x}$	State vector of aircraft
α, β, γ	Attitude components in the wind frame: angle-of-attack, sideslip, and flight path angle
$\delta_a, \delta_e, \delta_r$	Control surface deflections: ailerons, elevator, and rudder
η	Node of RRT motion planning tree
ρ	Density of air
ϕ, θ, ψ	Attitude components in the body frame: roll, pitch, and yaw
$\boldsymbol{\omega}$	Angular velocity
ω_T	Rotational speed of motor and propeller
$(\cdot)_{\text{aero}}$	Aerodynamic
$(\cdot)_{B,I,R}$	Resolved in the body frame, inertial frame, desired frame
$(\cdot)^{\text{cg}}$	Centre of gravity
$(\cdot)_{\text{fb,ff}}$	Feedback, feedforward
$(\cdot)_{h,v}$	Horizontal, vertical surface

$(\cdot)_{\text{ref}}$	Reference
$(\cdot)_{\text{seg}}$	Aircraft segment
$(\cdot)_T$	Thrust
$(\cdot)^\times$	Cross product matrix
$(\cdot)^\intercal$	Transposed
$\odot$	Quaternion product

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Historically, unmanned aerial vehicles (UAVs) have most commonly been associated with military applications. In recent years, however, there has been a shift in interest towards civilian applications and a corresponding increase in research and development in this area. It is expected, for instance, that the integration of civilian UAVs in the US National Air Space will permit job creation, estimated at 103,776 by 2025 [9]. These vehicles are now being proposed and utilized for uses as diverse as detection and mapping of forest fires, monitoring of long distance power lines and pipelines, aerial search and rescue, wildlife monitoring, road traffic monitoring, and police surveillance.

Unmanned aerial vehicles typically fall into two categories: fixed-wing and rotorcraft. Fixed-wing aircraft generate lift by moving forward and creating airflow over their wings. Rotorcraft achieve their lift from rotating blades. Rotorcraft are usually chosen for tasks that make use of their high maneuverability, such as their ability to handle precisely at low speeds and stop mid-flight, however, they lack the efficiency and endurance of fixed-wing aircraft. Advancements in research have begun to bridge the gap between these two categories of UAVs by increasing the agility of fixed-wing aircraft, and in so doing, broadening their suitability for missions requiring endurance *and* maneuverability. In this regard, it is worth noting that some other strategies have been pursued to reach the same end, including the design of unconventional aircraft configurations. For example, UAVs have been re-engineered with articulated wings [10] and flexible wings [11] to increase their agility. There have also been efforts to increase the endurance of rotorcraft [12, 13].

FIGURE 1.1: An agile fixed-wing UAV.

A modern class of fixed-wing UAVs, henceforth referred to as *agile fixed-wing UAVs*, are designed to be highly maneuverable. They are physically able to perform maneuvers such as: aggressive turns, flips, and transitions into nose-up hovering [14–16]. One such aircraft, used here for flight tests, is seen in Fig. 1.1. Their maneuverability is in large part due to their high thrust-to-weight ratio, which allows the aircraft's thruster to dominate its motion. They are also characterized by low aspect ratios, large control surfaces that can deflect to large angles, and a powerful propeller slipstream (also known as propwash).

The most impressive demonstrations of their maneuvering capabilities are exhibited by expert RC pilots during aerobatic competitions. Videos of these competitions expose the great potential that could be harnessed for autonomous flight. Although previous literature has explored agile UAV flight, piloted demonstrations prove that there is further room for the agility of these aircraft to be exploited. Our research was also motivated by the opportunity to capitalize on a state-of-the-art aircraft dynamics model that was recently developed in the McGill Aerospace Mechatronics Lab (AML). In comparison to the existing literature, this model is highly accurate and captures the full flight envelope of an agile fixed-wing UAV, from -180 to 180 degrees in angle-of-attack and sideslip. We believe that this uniquely comprehensive and accurate model can be used to make advancements in maneuver design, control, and motion planning.

1.2 Objectives of Dissertation

In broad terms, the topic of this thesis is autonomous flight with agile fixed-wing UAVs. Achieving full autonomy requires the coordination of many moving parts; to keep the scope of the research realistic, this thesis focuses on maneuver design, control, and planning for agile fixed-wing flight.

The first major goal of the thesis is to design agile maneuvers that take advantage of the aircraft's full flight envelope. The aim is to develop a general and systematic approach to designing and controlling such maneuvers, and to be able to demonstrate them in hardware experiments.

Once the maneuvers are designed and tested, we move on towards the goal of integrating the maneuvers into a motion planning framework. There are many different scenarios for which a motion planner is applicable. Here, we consider the specific problem of generating a feasible motion plan to guide the aircraft to a desired goal region through a highly-constrained, three-dimensional, known environment with static obstacles. We recognize that the last two descriptors of the motion planning problem are contentious in terms of their real-world practicality, however, a planner that satisfies this scenario could presumably be augmented in the future to handle more sophisticated conditions. By way of example, this planner could be used to generate a 'global' plan based on the known environment, which could then be enhanced by a local obstacle detection and avoidance system that diverts and returns the aircraft to the global plan as needed. The particular objectives of the planning framework are that it be able to exploit the aircraft's maneuvering capabilities, and that it couple with a control system such that the plan can effectively be tracked. We also intend for the planner to run in real-time, and do so using the limited computational resources available for use on-board a lightweight UAV. All flight tests rely solely on on-board sensing and computing equipment.

1.3 Literature Review

The literature review covers the following topics: aircraft dynamics modeling, trajectory generation, flight control systems, motion planning, and experimental demonstrations. While the dynamics model used here is not the original work of this thesis, it is integral to the contributions made, and thus should be understood in the broader context of the literature. The term *trajectory generation* is used here to essentially be interchangeable with *maneuver design*, i.e. generating trim conditions or finite-time transitions between

two states. This ought not be confused with motion planning, which is the algorithmic piecing together of trajectories to connect starting and goal states in an environment with obstacles. There can be significant overlap between control system design and trajectory generation (or even motion planning), but we attempt to separate these topics in the literature review for the sake of clarity and organization. The review mainly focuses on fixed-wing UAV research, however, there are some relevant works to be considered that stray outside of this fairly narrow field.

1.3.1 Aircraft Dynamics Modeling

While there has been extensive research conducted on the modeling of conventional aircraft and combat aircraft [17], these techniques do not necessarily extend well to agile fixed-wing UAVs. These aircraft are characterized by their high thrust-to-weight ratios, low aspect ratios, and large control surfaces; and for these reasons are able to maneuver effectively, even at very low speeds. Their highly nonlinear dynamic behavior would be cumbersome to capture accurately using the traditional linear stability derivatives approach. Their nonlinear dynamics, coupled with the substantial forces and slipstream generated by the thruster, call for new modeling techniques that span the full flight envelope. Only recently has research focused on developing new approaches to the modeling of agile fixed-wing UAVs and similar small air vehicles [18, 19].

Acquiring flight data is relatively easy and inexpensive, and thus system identification techniques have become prevalent for these aircraft [20–22]. However, capturing the full flight envelope of a highly maneuverable vehicle this way is impractical. Physics-based models are also prevalent, but are often simplified and limited to a small portion of interest of the aircraft's flight envelope [23–25]. Recently, a small body of research has been dedicated to fully and realistically modeling the dynamics of this class of aircraft through first principles [26], with contributions focusing on aerodynamics [27, 28], thruster dynamics [29, 30], and propeller slipstream models [7, 31]. In this thesis, we make use of the model described in [7, 27, 29].

1.3.2 Trajectory Generation

Trajectory generation is the process of designing feasible trajectories, and in some cases, determining their corresponding control inputs. While some simple trajectory generation problems can be solved analytically, most require the use of numerical methods, which

can be computationally intensive. This study has largely been shaped by advancements in the robotics and control community [32–35].

There are a number of approaches to generating trajectories for UAVs. Schollig et al. derived basic motion primitives for quadcopter flight by studying the elements of dance, namely, the usage of: time, space, energy, and structure [36]. Dynamically feasible trajectories can be revealed by flying or simulating flight under the control of feedback controllers [37, 38]. Another prevalent strategy is to learn trajectories from the flight of expert human pilots [14, 39–41]. A disadvantage of this approach is the need to rely on the competence of the pilot. It is difficult for a pilot to consistently perform the same maneuver manually, and the performance will inevitably be sub-optimal. Anecdotally, Mellinger et al. sought to define dynamically feasible trajectories for a quadrotor and found that human pilots were unable to fly the vehicle in their desired manner [38].

Alternatively, trajectories can be generated via optimization, typically by solving optimal control problems [22, 42, 43]. An optimal control problem is defined by the dynamics of the aircraft - which may be expressed as a system of differential equations - and an additional set of constraints. There are many methods for solving optimal control problems which all loosely fall into two categories: direct and indirect methods. Indirect methods use analytical or numerical procedures to solve an infinite-dimensional problem by finding a solution where the total differential of the performance measure is zero. In direct methods, nonlinear programming is used to solve the problem based on the calculus of variations or the maximum principle [44]. The optimal control approach avoids the issues associated with learning from piloted flight, and can extract the optimal trajectory and control policy in the absence of external disturbances (e.g. from wind) and state estimation errors. An additional benefit of this method is that specific functional objectives can be incorporated into the maneuver design by imposing boundary conditions and path constraints. Well-defined, functional maneuvers are suitable for primitive-based real-time motion planning, such as in [45], where the motion planning algorithm is applied to vehicles described by hybrid representations: trim conditions and finite-time transitions between them. Generating trajectories by solving optimal control problems is the approach taken in this thesis. We find this approach particularly advantageous here because it allows us to exploit the previously mentioned high-fidelity dynamics model, to generate agile motions that are not bound to specific regions of the aircraft's flight envelope.

Trajectories are often defined by and generated using specific conditions (e.g. boundary conditions). As an alternative to tediously generating and storing each particular trajectory that may be desired, some methods of parametrization have been developed. Using

a small set of trajectories, Dever et al. developed an algorithm to create continuously parametrized classes of dynamically feasible trajectories for autonomous vehicles [46]. The algorithm was experimentally applied to a three-degree-of-freedom helicopter. Similarly, Paranjape et al. were interested in the parametrization of extreme maneuvers. In their work, two families of motion primitives were designed to address the problem of fast flight through a forest: 3D circular paths between two points and aggressive turn-around maneuvers [47]. Both classes of primitives were continuously parametrized based on their corresponding control input sequences. In this thesis we present a novel method of continuous trajectory parametrization that uses Dynamic Time Warping [48]. The methodology strategically splits off-line and on-line computations such that the parametrization process is computationally light during flight.

1.3.2.1 Agile Maneuvers for Fixed-Wing UAVs

While some of the literature considers the agility of trajectories in a general sense, e.g. the aggressiveness of turns, it can be useful to target specific *agile maneuvers* that achieve some desirable purpose. Examples of such maneuvers include hovering [14, 24, 25], where the aircraft is suspended nose-up in mid-air by its propeller, or perching [15, 49, 50], where the aircraft transitions into a hover as it lands on a vertical surface. Another useful maneuver that has garnered attention is the knife-edge [51, 52]. This maneuver consists of rolling the aircraft by 90 degrees and, optionally, maintaining this roll angle before recovering to wings-level flight. In addition to being a fundamental aerobatic maneuver, the knife-edge presents a method of passing through passages more narrow than the aircraft's wingspan.

Many other agile maneuvers, such as tailslides and inverted spins, are simulated via piloted flight in [26]. In [53], the authors demonstrated simulations of a turn-around maneuver in which the aircraft first pulls its flight path angle up to nearly 90 degrees, then performs a 180 degree roll, drops its nose, and recovers to level flight. The maneuver was restricted to the vertical plane, similar to an Immelmann turn. Paranjape et al. investigated a turn-around maneuver [47], that was demonstrated in indoor flight tests.

This thesis provides a general and systematic approach to generating agile maneuvers that are not restricted to specific regions of the aircraft's flight envelope. We design aggressive turn-arounds, transitions into and out of hover, and a knife-edge maneuver.

1.3.3 Flight Control Systems

Many approaches to controller development exist for conventional fixed-wing flight. The control systems of most relevance here are those that are able to track aggressive or agile maneuvers. Flight control systems developed to execute these maneuvers have included closed-loop and open-loop control strategies. Frank et al. developed controllers individually using linear quadratic techniques for each of the following maneuvers: perched landing, hovering, transitioning from hovering to forward flight, and transitioning from forward flight to hover [24]. Cory and Tedrake performed an in-depth study of the hovering of fixed-wing UAVs, and subsequently developed an optimal linear controller [54]. In a later work, they developed a controller for perched landing [15]. The vehicle under consideration here was a glider with only one actuator – its elevator. Instead of relying on high thrust-to-weight ratios that are typical of agile fixed-wing UAVs, they developed a nonlinear controller that exploits the pressure drag on the aircraft created at high angles-of-attack. Also working with a glider, Moore et al. applied an approach termed *LQR-Trees* to perching and were able to define a range of initial conditions for which the maneuver could be performed [49] consistently. LQR-Trees combines locally valid linear quadratic regulator (LQR) controllers with a nonlinear feedback policy. In [52], a control system capable of performing knife-edge maneuvers is developed and demonstrated experimentally by navigating an aircraft through obstacles. In this work, a direct collocation method was used for trajectory planning, and time-varying linear quadratic regulators (TVLQRs) were used to stabilize the open-loop trajectories.

Desbiens and Cutkosky also considered perched landing, with the objective of minimizing the requirements of the fixed-wing UAV's sensing and control technology [55]. They were successful in this endeavor by using open-loop control to send the aircraft into a feasible flight envelope defined by pitch and velocity such that it could cling to a wall using microspines [56]. In a later work, Glassman et al. fortified this concept by exploiting barrier certificates to estimate a region of attraction for which the UAV could perch on a wall [57]. Piedmonte and Feron also found open-loop control to be practical for extreme maneuvering. In their work on aggressive maneuvering of autonomous aerial vehicles, they switch between a feedback control system for conventional flight modes and an open-loop 'maneuver logic' for aggressive maneuvering [41]. This control logic strategy follows from breaking down control input curves into segments that together generate a maneuver. Just as it is for learning trajectories, human-piloting is a prevalent strategy for learning control input sequences for classes of extreme maneuvers [14, 40, 41].

The control system employed here combines optimal open-loop control policies with a physics-based feedback controller. While staying near the nominal trajectory, the open-loop policies give the benefit of optimality. The feedback controller has a nested PD/PI structure that achieves tracking of time-dependent trajectories, including ones that are transient and aggressive. The scope of the thesis did not extend to addressing the problem of external disturbances or system uncertainty, and thus the robustness of the control system is neither imposed by design, nor verified systematically.

1.3.4 Motion Planning

Motion planning is a central component of autonomous flight; defined here as the automatic guidance of a robot through an environment with obstacles, from a specified initial state to a goal region. A sufficiently sophisticated motion planning algorithm will consider not only obstacles, but the kinematic and dynamic constraints of the robot. The algorithm will search through the robot's configuration space to find a 'path' to the goal. In addition to the geometric path the algorithm finds, it may also return the robot's full-state time history and/or the input commands needed to follow the path. Advancements in motion planning have largely been shaped by two fields: robotics and artificial intelligence (AI); and dynamical systems and controls [58]. The unique challenges of motion planning for UAVs, especially fixed-wings, are that of accounting for their dynamic constraints and exploiting their flight envelope.

Unless a ground station is going to be used to send commands to the aircraft, small UAVs are generally limited to the processing capabilities of small, lightweight computers. A class of algorithms that are well-suited for efficient real-time planning with limited resources are sampling-based algorithms. The most prevalent sampling-based methods are the Probabilistic Roadmap (PRM) [59], Rapidly-Exploring Random Trees (RRT) [60], and their variants [61–63]. The PRM algorithm is a multi-query algorithm that is probabilistically complete, but requires solving two-point boundary value problems to steer the system between two states. Solving a boundary value problem is a costly operation that can be impractical to do in real-time with limited resources. The RRT algorithm, on the other hand, is a single-query planner that is highly effective at generating dynamically feasible trajectories rapidly. The RRT algorithm handles complex constraints easily, finding a path to the goal region with minimal map exploration. An asymptotically optimal version of RRT exists, termed RRT* [62]. This algorithm succeeds by finding a feasible solution quickly and then improving upon it in the remaining time to converge to the global optimal motion plan. It has been demonstrated on a robotic arm [64], and extended to

suit non-holonomic dynamic constraints [62]. Recently, there have been implementations of RRT-based algorithms for fixed-wing UAVs [65, 66]. Koyuncu et al., in their work on motion planning for agile unmanned air vehicles, used the RRT algorithm as a top-level planner to identify a path from initial to goal positions [67]. From here, they get rid of all unnecessary waypoints through a line-of-sight argument, and then a low-level planner selects dynamically feasible paths to connect the waypoints using a PRM algorithm. In a later work, the same authors proposed another multi-layer planner that incorporates an RRT-based algorithm and the line-of-sight argument, this time employing a B-Spline method to generate dynamically feasible trajectories to join together waypoints [68].

A useful concept termed the 'maneuver automaton' is formalized in Frazzoli et al. [69], to address the problem of planning in the high-dimensional state space of robots like fixed-wing UAVs. The maneuver automaton, which captures formal properties of a trajectory library, is used as a hybrid representation of a vehicle model, wherein motion primitives are used to pre-compute a cost-to-go map. In a real-time implementation of RRT, the maneuver automaton replaces the vehicle's first-order dynamics. In Frazzoli et al. [45], the states of the automaton are trim states of the vehicle, and maneuvers are used to transition between the states. In Gavrilets et al. [41], the concept is explored for learning motion primitives from human-piloted aerobatic flight, and in Schouwenaars et al. [70], the hybrid model replaces trim states with 'linear time-invariant modes' and fixed-duration transitions.

A number of planning methodologies for UAVs have been explored that combine motion primitives with sampling-based planners. In [65], a pre-defined motion primitive set is used for 2D RRT-based path planning with a fixed-wing UAV. Pre-computed motion primitives are used in an A*-based planner in [71]. In [72], a real-time framework was developed, which incorporates a look-up table of boundary-value problem solutions into a sampling-based algorithm called Fast Marching Trees (FMT*). Trajectory funnels for robust motion planning were applied to a highly maneuverable fixed-wing aircraft in [73].

A modern and prevalent alternative for robot planning and control is model predictive control (MPC), also known as receding horizon control [74, 75]. The MPC planning method is basically equivalent to repeatedly solving an optimal control problem on-line, with obstacles as additional constraints. This method can be highly effective with linear and even nonlinear models, but real-time optimizations can be resource-intensive.

Two-phase planning techniques have been proposed for a number of fixed-wing applications. First, a high-level planner constructs a course grid or path, and then a low-level planner refines it to account for more restrictive constraints [76]. In, [77], for example, a

two-phase navigation plan is proposed, which consists of a top-level (global) planner and a local motion planner. The global planner, based on the A* algorithm [78], considers only the vehicles kinematic constraints; the local planner computes a finer trajectory for the aircraft to track, using sampling-based motion planning under differential constraints.

This thesis utilizes the approach mentioned above of combining pre-generated motion primitives with a sampling-based planner (RRT). Relative to the existing literature, the following contributions differentiate this work. The motion primitives are generated using a high-fidelity dynamics model, and thus can exploit the aircraft's full agility. The trajectory solutions are composed of the aircraft's full 12-state vector, as well as all of its control inputs. Additionally, we note that by the way in which the motion primitives are integrated into the planning algorithm, the size of the library has no effect on planning time; typically, the computational cost of RRT increases with the number of primitives [79]. Furthermore, our implementation places no constraints on the sequencing of primitives. Eliminating the need for an extremely large set of pre-defined transition maneuvers, a transition heuristic is applied to allow any primitive to follow another.

1.3.4.1 Motion Planning Under Uncertainties

There are various complications to motion planning problems that arise due to uncertainties [80]. While the specific topic of handling uncertainties falls out of the scope of this thesis, it is a pressing and relevant problem that many have considered, and thus warrants some discussion. Uncertainties, as identified by LaValle and Sharma in [81], can be classified into the following types:

- Uncertainty in vehicle dynamics
- Uncertainty in the knowledge of the environment, i.e. obstacles
- Disturbances in the operational environment (e.g. wind)
- Uncertainty in pose information (where the UAV is in relation to the map)

An approach to making the motion planner robust to modeling uncertainties in the vehicle dynamics was to implement the aircraft into an optimization as a closed-loop system, storing the resulting trajectories in the maneuver automaton [82]. This is effective to the extent that the closed-loop policy will correct for small perturbations caused by uncertainties in the model. Robust MPC is another method of accounting for uncertainties in vehicle dynamics [83].

Incremental graph search algorithms specialize in being able to account for obstacles that are spotted in real-time, as opposed to being determined as part of the environment a

priori. When obstacles arise, these algorithms work by updating their plan based on information from the previous plan [80]. Missiuro and Roy extended the PRM algorithm to incorporate environment uncertainties [84]. In this paper, they modeled a cost for collisions when travelling through uncertain regions of the environment. Reactive Planning (RP) can also deal with obstacles that can not be predicted a priori [85, 86]. RP typically resorts to allowing the local planner to take priority, temporarily ignoring the high-level planner for the sake of avoiding obstacles spotted by sensors. Evolutionary algorithms present another approach to the problem of handling uncertainties in the knowledge of the environment. Nikolos et al. employed a modified breeder genetic evolutionary algorithm, in part to solve a UAV navigation problem in an unknown three-dimensional environment [87].

Due to their size and relatively slow speed, UAVs are particularly susceptible to wind disturbances. McGee et al. explored the problem of generating optimal time paths in a two-dimensional plane for an aircraft subject to a constant wind [88]. Using the Minimum Principle they were able to draw conclusions about the shape of such optimal paths. Nelson et al. accounted for ambient winds while approaching the problem of fixed-wing UAVs following straight lines and orbits. Their intention was to guide the aircraft by a path, rather than a trajectory, making it easier to regulate the speed of the aircraft [89]. Jennings et al. tackled the presence of wind disturbances with Dynamic Programming by changing the problem of traveling to a fixed point in wind to travelling to a moving point without wind [90].

Uncertainty in pose introduces the issue of the aircrafts position and orientation being uncertain with respect to the environment. This is most often a problem when a GPS cannot be used, and can even be an issue with a GPS, when its own deficiencies are present. An adaptation of PRM, called Belief Roadmap (BRM), proposed by Prentice and Roy, uses a variant of the Kalman filter to account for uncertainties in pose [91].

1.3.5 Experimental Demonstrations

Being relatively new, the study of autonomous agile fixed-wing UAV flight has only yet produced a few experimental demonstrations. This section will describe some of the most relevant work in this area.

In [49], nonlinear feedback control designs are used for perching of a light-weight (85 g) fixed-wing glider; an airplane whose only actuator is an elevator. Experiments were performed indoors, using off-board sending and control. A Vicon motion capturing system

was used for sensing, and a ground control station performed estimation and control. The experiments were demonstrated for different initial speeds (6 - 8 $\mathrm{m\,s^{-1}}$, using a custom-made launching device. Under these conditions, the aircraft had a high rate of success performing precise perched landings.

Paranjape et al., in [47], aimed to perform fast and agile flight through dense obstacle fields. Two types of motion primitives were developed and experimentally validated: 3D circular paths between two points in space, and aggressive turn-arounds. Indoor flight tests were performed using a very small, 11 g aircraft. The tests also relied on a Vicon motion capturing system to measure the aircraft's position and orientation. The paper included motion planning simulations, but experiments were restricted to flights that terminated after the first waypoint.

In [73], the authors were interested in motion planning with guaranteed success in the face of disturbances, and uncertainties in the model and environment. Libraries of trajectory *funnels* were pre-computed for use in real-time motion planning. The indoor flight experiments were performed with a small, highly maneuverable RC airplane. A launching mechanism sent the aircraft into flight, at which point the position and geometry of all obstacles became known to the computer, thereby forcing the planner to run in real-time. Once again, these experiments made use of a Vicon motion capture arena, and computations were solely performed off-board. The indoor testing environment was small, and thus only one funnel, i.e. one obstacle-evasion maneuver, was tested at a time. Many successful runs were recorded, where the aircraft was able to avoid a small but dense obstacle field composed of complex geometries.

Bry et al. developed trajectory planning and state estimation algorithms for aggressive flight of micro aerial vehicles in known, obstacle-dense environments [92]. In contrast to the previously mentioned demonstrations, the fixed-wing experiments in this work perform all of sensing, state estimation, and control on-board. The aircraft was equipped with a laser rangefinder, an IMU, and an Intel Atom flight computer. Prior to flying, waypoints were hand-selected and *Dubins-Polynomial* trajectories were optimized to connect them. The indoor experiments demonstrated impressive closed-loop tracking of these plans, with small position and velocity errors.

In [93], pushbroom stereo was used for obstacle detection and avoidance in high-speed flight. The experiments in this work used a 664 g fixed-wing aircraft with two moving flaps - the aircraft had no conventional fuselage or tail. The computation and GPS-denied sensing was done fully on-board, using two ODROID-U3 single-board computers, an IMU

platform, a barometric altimeter, and two cameras. Outdoor flight tests showed obstacle detection and avoidance at speeds of up to 14 $\mathrm{m\,s^{-1}}$.

The flight test demonstrations presented in this thesis show tracking of highly agile maneuvers, as well as full start-to-goal motion plans. The flight tests are the first to demonstrate their level of implementation, in terms of the extent of the flight envelope utilized, and fully relied on on-board sensing and computing – including for motion planning.

1.4 Thesis Organization

The thesis is organized as follows. First, preliminaries are discussed, including an aircraft dynamics model and a feedback controller, both of which are contributions of past and present students of the AML. Also in the preliminaries chapter is a framing of a general optimization problem. The optimization problem is formalized upfront because it is subsequently used in Chapters 3 and 4. Chapter 3 discusses a slight sidetrack from the main objectives of the thesis; an investigation performed to evaluate the significance of sideslip and slipstream in extreme maneuvering. Chapter 4 describes the strategy for designing maneuvers. As a precursor to the further goal of motion planning with maneuvers, the chapter is framed as the development of a *maneuver space*. The maneuver space includes *motion primitives*, which are classified as either agile maneuvers or trim conditions. This chapter also discusses the methodology of transitioning between primitives, and how the agile maneuvers can be continuously parametrized. The main contribution of Chapter 5 is to take this maneuver space and integrate it into a real-time motion planner. We illustrate the advantages of using the maneuver space by contrasting the planning framework to a baseline approach that uses Dubins curves [94]. Chapter 6 describes the steps taken to validate the work of the previous chapters. We describe simulations and flight tests of agile maneuvers and real-time motion planning.

Figure 1.2 shows how the topics and chapters of the thesis are connected. As illustrated, the preliminary topics - the aircraft dynamics model, feedback controller, and optimization framework - relate to multiple sections. The general optimization framework, which utilizes the dynamics model as a set of constraints, is used in Chapter 3 for an investigation of the role of sideslip and slipstream in extreme maneuvering. In Chapter 4, the framework is used for generating the trajectories that make up the maneuver space, and in Chapter 5, the maneuver space is integrated into a motion planner. Simulations and flight tests are used to validate the agile maneuver design and motion planner. Both sets

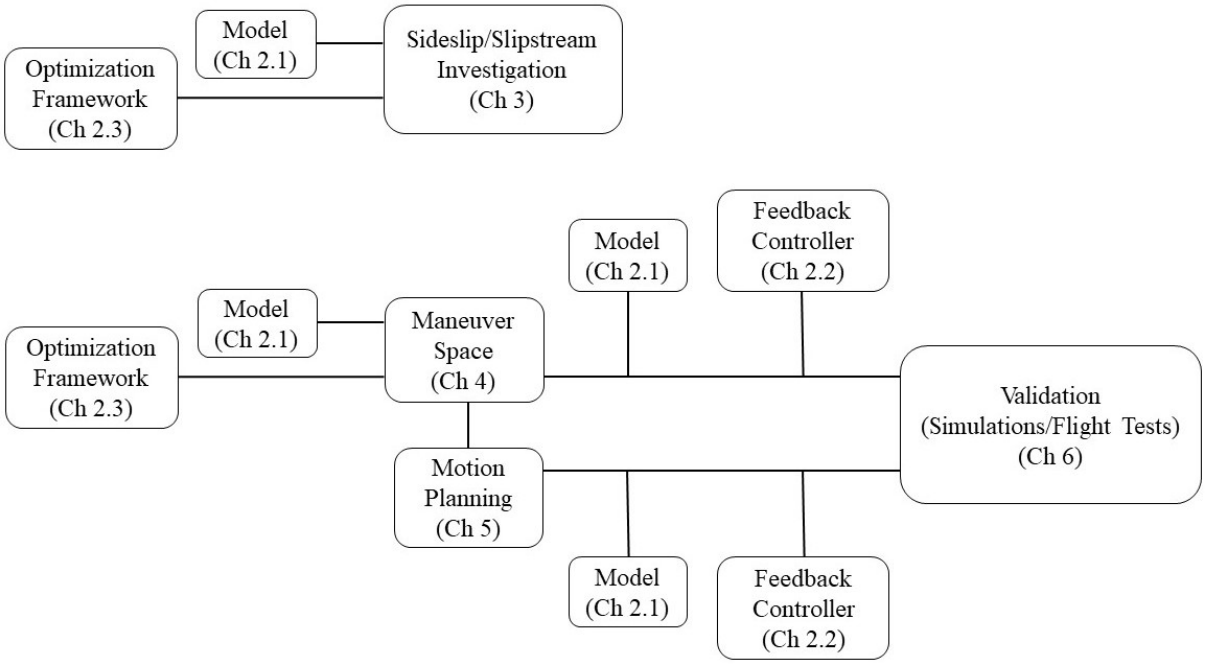


FIGURE 1.2: Organization of thesis components.

of tests make use of the feedback controller, which is an integral part of the full control system, and the simulation environment includes the aircraft dynamics model.

Chapter 2

Preliminaries

This chapter discusses preliminary work that will be exploited throughout the thesis. The first two sections of this chapter describe work that was originally authored by other members of the McGill Aerospace Mechatronics Lab. Section 2.1 summarizes an aircraft dynamics model, and Section 2.2 describes a feedback controller. Both of these components are used as tools towards achieving the objectives of the thesis, but do not represent contributions of this thesis. The final section, Section 2.3, lays out a general optimization framework that is utilized in Chapters 3 and 4.

2.1 Aircraft Dynamics Model

The aircraft dynamics model is a comprehensive representation of the dynamics of an agile fixed-wing UAV. Parts of the model are generalizable, but where appropriate, it has been tailored to the fixed-wing UAV used for experimental validations. The full flight envelope of the aircraft is captured, accounting for a ± 180 degree range in both angle-of-attack and sideslip. The model addresses the dynamics of the thruster, the slipstream effects of the propeller, and the aircraft's nonlinear aerodynamics. The most complete description of the model can be found in Waqas Khan's thesis [95]. The main components of the model are also described individually in a number of articles that will be cited throughout the summary of the model presented in this chapter.

To the best of our knowledge, this is the most detailed and accurate model to be used for motion planning and control of an agile fixed-wing UAV. The objectives of the thesis were in part motivated by the opportunity to take advantage of the model, since it accurately and uniquely captures dynamic behavior that would be difficult to harness

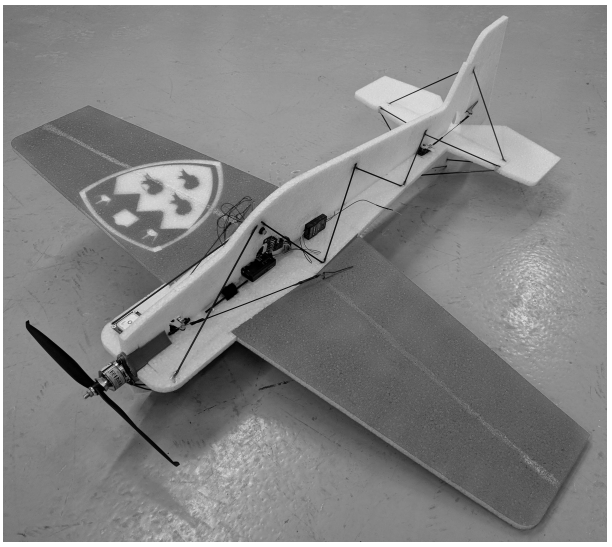


FIGURE 2.1: McFoamy: an agile fixed-wing UAV.

for autonomous flight otherwise. The model is exploited in multiple ways throughout the thesis. In Chapter 3, it is used to analyze the significance of slipstream and sideslip modeling, and in Chapter 4, it is used for trajectory generation. The model is also used for simulations throughout the thesis.

There are a number of assumptions present in the model. It is assumed that there is no swirl component of the induced velocity created by the thruster; the slipstream model has only been validated for a stationary propeller [7]; and there is no modeling of unsteady aerodynamic effects. The last assumption is based on the reduced frequency parameter, $k_{\dot{\alpha}}$, that characterizes the degree of unsteadiness during flight. For fixed-wing flight, $k_{\dot{\alpha}} = \frac{\dot{\alpha}c}{2V} > 0.05$ [28, 96] characterizes unsteady flow, which was found to very rarely and very briefly occur during any of the maneuvers investigated in this thesis.

2.1.1 Aircraft Configuration

The work of Khan was tailored to an RC plane built upon the Yak54 by Great Planes airframe. This airframe recently went out of production, and thus we switched to a similar one (for experiments in this thesis), the *McFoamy* by West Michigan Park Flyers. The two airframes are very similar, with the major difference being that the foam of the McFoamy is more resilient to crashes. The McFoamy aircraft, shown in Fig. 2.1, has a wingspan of 0.86 m. Updating the dynamics model for the new airframe involved creating a CAD model and measuring properties of the airframe's geometry.

TABLE 2.1: Properties of the McFoamy aircraft.

Parameter	**Symbol**	**Value**	**Unit**
Mass	m	0.576	kg
Moments and products of inertia	I_x	4.02×10^{-3}	$\mathrm{kg\,m^2}$
	I_y	1.44×10^{-2}	$\mathrm{kg\,m^2}$
	I_z	1.77×10^{-2}	$\mathrm{kg\,m^2}$
	I_{xz}	4.60×10^{-4}	$\mathrm{kg\,m^2}$
Location of cg	x^{cg}	-0.270	m
(measured from propeller plane)	y^{cg}	0	m
	z^{cg}	6.0×10^{-3}	m
Wing area	S	0.143	$\mathrm{m^2}$
Wing span	b	0.86	m
Mean aerodynamic chord	$\bar{c}$	0.21	m
Propeller disk area	A_{prop}	5.07×10^{-2}	$\mathrm{m^2}$
Maximum control surface deflections	$\delta_{a_{max}}$	42	deg
	$\delta_{e_{max}}$	45	deg
	$\delta_{r_{max}}$	46	deg
Control derivative coefficients	$C_{l_{\delta_a}}$	-6.78×10^{-4}	$\mathrm{deg^{-1}}$
	$C_{l_{\delta_r}}$	9.31×10^{-4}	$\mathrm{deg^{-1}}$
	$C_{m_{\delta_e}}$	-1.18×10^{-2}	$\mathrm{deg^{-1}}$
	$C_{n_{\delta_r}}$	-3.57×10^{-3}	$\mathrm{deg^{-1}}$

The airframes were chosen because they are lightweight, easily assembled, and the foam absorbs most of the impact in crashes. The mass, inertia, center of gravity location, and other properties of the UAV are listed in Table 2.1. The moments and products of inertia are determined using the CAD model. The thruster includes a RimFire 400 Outrunner brushless DC Motor by Great Planes, and an Electrifly PowerFlow 10×4.5 propeller. It is powered by an 11.1 V lithium polymer battery through an Electrifly Silver Series 25A brushless electronic speed controller. The testbed also includes sensing and computing equipment (accounted for in the values given in Table 2.1), which are described in Chapter 6.

2.1.2 Frames of Reference

There are three frames of references referred to throughout this chapter. They are the inertial frame, the body frame, and the desired frame. The former two are pictured in Fig. 2.2. The inertial frame has an arbitrary origin, fixed relative to the surface of the Earth. The x_I axis points north, the y_I axis east, and the z_I axis towards the center of the Earth. The body frame is aligned with the aircraft, with its origin at the aircraft's center of gravity. The x_B axis points out the nose of the aircraft. The z_B axis points

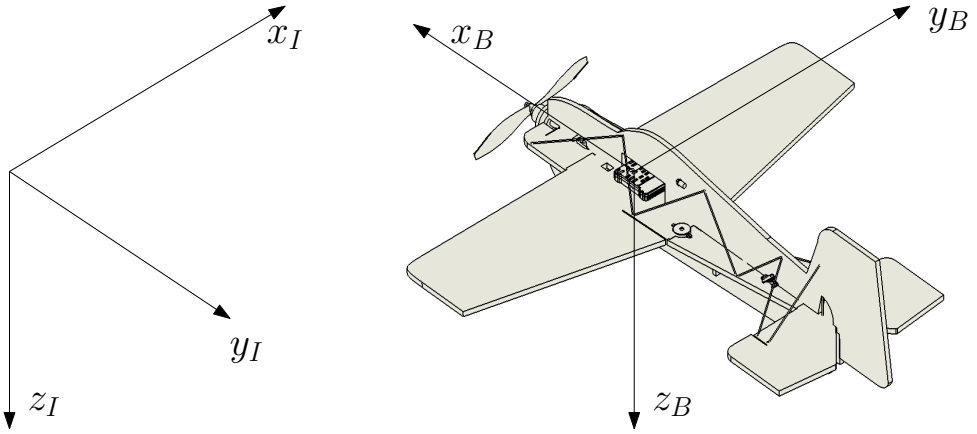


FIGURE 2.2: Frames of reference: inertial frame (x_I, y_I, z_I) and body frame (x_B, y_B, z_B).

below the aircraft, and the y_B axis points out the aircraft's right wing (from the top-down perspective). The third frame of reference, the desired frame, is aligned with the body frame as it is meant to be at a given time on a given reference trajectory. If the aircraft were tracking a reference trajectory perfectly, the desired and body frame would be aligned. The desired frame is only relevant to the feedback controller; it is used there to express position and velocity errors in terms of where the aircraft is meant to be at a given time on a given reference trajectory.

2.1.3 Equations of Motion

The full aircraft dynamics model is summarized by the equations of motion for a rigid body:

$$\begin{aligned} \dot{\mathbf{V}}_B^{\text{cg}} &= \frac{1}{m}\mathbf{F}_B - \boldsymbol{\omega}_B^{\times}\mathbf{V}_B^{\text{cg}} \\ \dot{\boldsymbol{\omega}}_B &= \mathbf{I}_B^{-1}[\mathbf{M}_B - \boldsymbol{\omega}_B^{\times}\mathbf{I}_B\boldsymbol{\omega}_B] \\ \dot{\mathbf{p}}_I &= \mathbf{C}_{BI}^{\intercal}\mathbf{V}_B^{\text{cg}} \\ \dot{\mathbf{q}} &= \frac{1}{2}\mathbf{q} \odot \boldsymbol{\omega}_B \end{aligned} \tag{2.1}$$

The translational and angular velocity of the aircraft, $\mathbf{V}_B^{\text{cg}} = [u, v, w]^{\intercal}$ and $\boldsymbol{\omega}_B = [p, q, r]^{\intercal}$, are resolved in the body frame, as are the inertia matrix, $\mathbf{I}_B$, and the net forces and moments acting on the aircraft, $\mathbf{F}_B$ and $\mathbf{M}_B$. The position vector, $\mathbf{p}_I = [x, y, z]^{\intercal}$, is resolved in the inertial frame. A quaternion, $\mathbf{q}$, is used to represent attitude, because

unlike the standard Euler angles, quaternions do not suffer from issues of singularity. The aircraft's full state vector is $\mathbf{x} = [u,\ v,\ w,\ p,\ q,\ r,\ q_1,\ q_2,\ q_3,\ q_4,\ x,\ y,\ z]^\intercal$, together with the constraint $||\mathbf{q}|| = 1$, needed for quaternion rotation. The modeling challenge is to properly characterize the net forces and moments acting on the aircraft, $\mathbf{F}_B$ and $\mathbf{M}_B$. These have three sources: gravity, propulsion, and aerodynamics, which will be discussed in Sections 2.1.4 - 2.1.6. The cross product matrix of vector $\boldsymbol{\omega}_B$ is given by:

$$\boldsymbol{\omega}_B^\times = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}, \tag{2.2}$$

and the quaternion product, $\odot$, in the last line of Eq. 2.1 produces:

$$\mathbf{q} \odot \boldsymbol{\omega}_B = \begin{bmatrix} q_2 p + q_3 q + q_4 r \\ q_1 p + q_3 r - q_4 q \\ q_1 q + q_4 p - q_2 r \\ q_1 r + q_2 q - q_3 p \end{bmatrix} \tag{2.3}$$

The term $\mathbf{C}_{BI}$ is the rotation matrix that transforms vectors from the inertial frame to the body frame. This matrix is constructed from the quaternion attitude representation, which is defined as $\mathbf{q} = q_1 + q_2\mathbf{i} + q_3\mathbf{j} + q_4\mathbf{k}$:

$$\mathbf{C}_{BI} = \begin{bmatrix} q_1^2 + q_2^2 - q_3^2 - q_4^2 & 2(q_2 q_3 + q_1 q_4) & 2(q_2 q_4 - q_1 q_3) \\ 2(q_2 q_3 - q_1 q_4) & q_1^2 - q_2^2 + q_3^2 - q_4^2 & 2(q_3 q_4 + q_1 q_2) \\ 2(q_2 q_4 + q_1 q_3) & 2(q_3 q_4 + q_1 q_2) & q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{bmatrix} \tag{2.4}$$

2.1.4 Thruster Model

The thruster model is responsible for computing the aerodynamic and gyroscopic forces and moments produced by the thruster unit, expressed in the body frame as $\mathbf{F}_{B,T}$ and $\mathbf{M}_{B,T}$. These terms are dependent on the thrust control input, as well as the aircraft's translational and angular velocities. To compute the forces and moments, blade element momentum theory is employed, and all flow conditions - static, axial, oblique, and reverse flow - are captured. The model was experimentally validated in [29], where full details of the modeling of the battery, electronic speed controller, brushless DC motor, and propeller can be found.

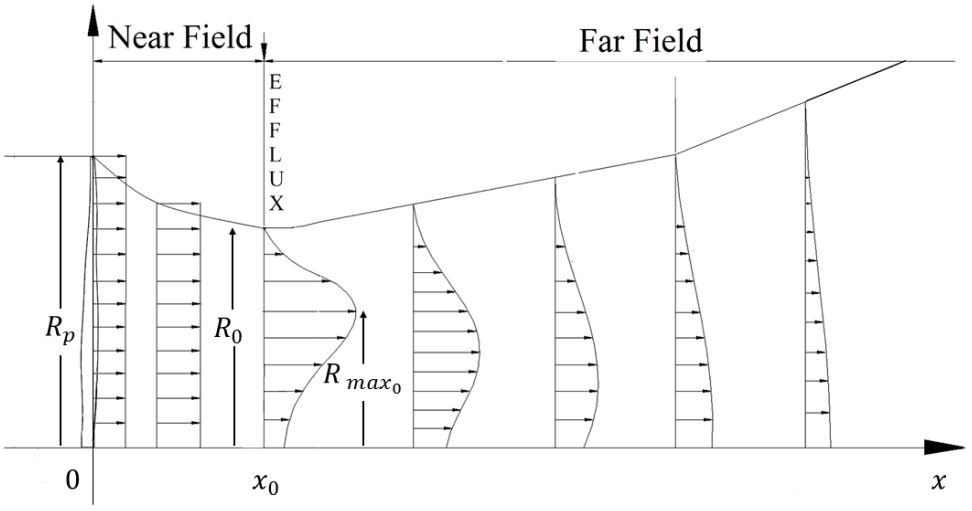


FIGURE 2.3: Propeller slipstream: near field and far field regions [7].

2.1.5 Propeller Slipstream Model

The propeller slipstream effect, also referred to as propwash, describes the additional airflow induced over the aircraft by the rotating propeller. This additional airflow adds to the effectiveness of the control surfaces to produce forces and moments. The slipstream model calculates the propwash in the form of a velocity field, denoted in general as $V_s(x, r)$. This is an axial airflow that is calculated differently for two regions: a near-field region and a far-field region, which are separated by the 'efflux plane', located roughly 20% of the fuselage length downstream of the propeller, as seen in Fig. 2.3. The front of the aircraft's fuselage and part of its main wing (that which is closest to the fuselage and inside the slipstream) reside in the near-field region. In this region, the axial airflow's motion is dominated by the pressure force created by the propeller, which induces an acceleration. The airflow here is calculated using momentum theory [97]:

$$V_s(x) = V_{i0,avg}\left[1 + \frac{x/R_p}{\sqrt{(1 + (x/R_p)^2)}}\right], \tag{2.5}$$

where $V_s(x)$ is the momentum-averaged induced velocity at an axial distance, x, from the propeller plane. The variable R_p is the propeller radius, and $V_{i0,avg}$ is the momentum-averaged induced velocity at the propeller plane ($x = 0$).

The horizontal and vertical tail surfaces and aft fuselage are located in the far-field region. In this region, air viscosity and turbulence become more prominent and cause diffusion of the slipstream radially into the ambient flow. As a result of the momentum transfer

TABLE 2.2: Slipstream parameters of the McFoamy aircraft.

Parameter	**Symbol**	**Value** [m]
Propeller radius	R_p	0.127
Slipstream radius at the efflux plane	R_0	0.0940
Radial position of maximum induced velocity at the efflux plane	R_{max_0}	0.0589
Axial position of the efflux plane	x_0	0.194

between the fast-moving slipstream and the slow-moving ambient flow, the slipstream velocity decreases and the slipstream expands radially. In the far-field region, the slipstream velocity profile is calculated in terms of the axial distance from the propeller plane, x, and the radial distance, r, via a one-term Gaussian function that accounts for the diffusion phenomenon:

$$V_s(x,r) = V_{max} \exp\left[-\left(\frac{r - R_{max}}{a_3 R_{max_0} + b_3(x - x_0 - R_0)}\right)^2\right], \tag{2.6}$$

where V_{max} and R_{max} are the maximum induced velocity and its radial position, respectively. Both are linear functions of axial distance:

$$\begin{aligned} V_{max} &= V_0\left(a_1 - b_1\frac{x - x_0}{D_0}\right) \\ R_{max} &= R_{max_0}\left(a_2 - b_2\frac{x - x_0}{D_0}\right) \end{aligned} \tag{2.7}$$

The slipstream radius, R_0, contracted diameter, $D_0 = 2R_0$, and radial position of the maximum induced velocity, R_{max_0}, are measured at the efflux plane ($x = x_0$, see Fig. 2.3). Coefficients a_1, a_2, a_3 and b_1, b_2, b_3 are determined semi-empirically based on experiments with a propeller slipstream in air. Their values, listed in [7], correspond to piecewise linear curve fits, split by axial distance. The slipstream parameters for the McFoamy testbed are listed in Table 2.2. To give a rough sense of the slipstream's magnitude, when flying straight and level at 5 $\mathrm{m\,s^{-1}}$, the additional airflow is as large as 9.5 $\mathrm{m\,s^{-1}}$ at some points on the aircraft. The slipstream is less dominant, but still significant, when the aircraft is flying at higher speeds. For example, at 13 $\mathrm{m\,s^{-1}}$ ground speed, the maximum additional airflow is 7.8 $\mathrm{m\,s^{-1}}$.

The slipstream effect also includes a swirl component. This effect is modeled as a reduction of 60% on the thruster's reaction torque, $\mathbf{M}_{B,T_x}$. Further details can be found in [7].

2.1.6 Aerodynamics

For the aerodynamic modeling, a component breakdown approach is employed. Each of the aircraft's main components (the main wing, tail, rudder, and fuselage) are split into segments, whose aerodynamics are computed individually and then summed together. The component breakdown approach is used because aerodynamic behavior varies over the surfaces of the aircraft. For instance, the propwash velocity field varies by body frame location, and segments have different ratios of chord length to flap chord length. The velocity at the aerodynamic center of each segment is calculated as follows:

$$\mathbf{V}_{\text{seg}} = \mathbf{V}_B^{\text{cg}} + \boldsymbol{\omega}_B^{\times}\mathbf{p}_{\text{seg}} + \mathbf{V}_{s,\text{seg}}, \tag{2.8}$$

where the position vector, $\mathbf{p}_{\text{seg}}$, measures from the aircraft's center of gravity to the aerodynamic center of the segment. Note that the slipstream only has an axial component, thus $\mathbf{V}_{s,\text{seg}} = [V_s,\ 0,\ 0]^\intercal$. The components of $\mathbf{V}_{\text{seg}} = [V_{\text{seg,x}}, V_{\text{seg,y}}, V_{\text{seg,z}}]^\intercal$ are used to calculate the angle of attack on the segment:

$$\alpha_{h,\text{seg}} = \arctan\frac{V_{\text{seg,z}}}{V_{\text{seg,x}}}, \quad \alpha_{v,\text{seg}} = \arctan\frac{V_{\text{seg,y}}}{V_{\text{seg,x}}} \tag{2.9}$$

The terms $\alpha_{h,\text{seg}}$ and $\alpha_{v,\text{seg}}$ apply to horizontal and vertical surfaces, respectively. The aerodynamic forces and moments on each horizontal segment are calculated as:

$$\begin{aligned}\mathbf{F}_{B,\text{seg}} &= \frac{1}{2}\rho b_{\text{seg}}\bar{c}_{\text{seg}}(V_{\text{seg,x}}^2 + V_{\text{seg,z}}^2)[CF_{\text{seg,x}}, 0, CF_{\text{seg,z}}]^\intercal \\ \mathbf{M}_{B,\text{seg}} &= \frac{1}{2}\rho b_{\text{seg}}\bar{c}_{\text{seg}}^2(V_{\text{seg,x}}^2 + V_{\text{seg,z}}^2)[0, C_{M,ac,\text{seg}}, 0]^\intercal,\end{aligned} \tag{2.10}$$

with analogous equations for the vertical segments:

$$\begin{aligned}\mathbf{F}_{B,\text{seg}} &= \frac{1}{2}\rho b_{\text{seg}}\bar{c}_{\text{seg}}(V_{\text{seg,x}}^2 + V_{\text{seg,y}}^2)[CF_{\text{seg,x}} \quad CF_{\text{seg,y}} \quad 0]^\intercal \\ \mathbf{M}_{B,\text{seg}} &= \frac{1}{2}\rho b_{\text{seg}}\bar{c}_{\text{seg}}^2(V_{\text{seg,x}}^2 + V_{\text{seg,y}}^2)[0 \quad 0 \quad -C_{M,ac,\text{seg}}]^\intercal,\end{aligned} \tag{2.11}$$

The terms b_{seg} and $\bar{c}_{\text{seg}}$ are the span and mean aerodynamic chord of the segment, and ρ is the density of air. The force coefficients are obtained by resolving lift and drag coefficients, C_L and C_D, using the segment's angle-of-attack:

$$\begin{aligned} CF_{\text{seg,x}} &= C_{L,\text{seg}} \sin \alpha_{h/v,\text{seg}} - C_{D,\text{seg}} \cos \alpha_{h/v,\text{seg}} \\ CF_{\text{seg,z/y}} &= -C_{L,\text{seg}} \cos \alpha_{h/v,\text{seg}} - C_{D,\text{seg}} \sin \alpha_{h/v,\text{seg}} \end{aligned} \tag{2.12}$$

The forces and moments are transferred to the aircraft's center of gravity and summed to give the total aerodynamic forces and moments acting on the UAV:

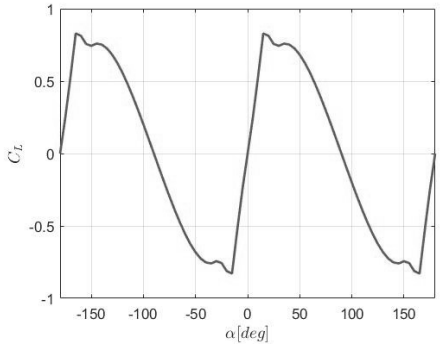
$$\begin{aligned} \mathbf{F}_{B,\text{aero}} &= \sum \mathbf{F}_{B,\text{seg}} \\ \mathbf{M}_{B,\text{aero}} &= \sum \left(\mathbf{M}_{B,\text{seg}} + \mathbf{p}_{\text{seg}}^{\times} \mathbf{F}_{B,\text{seg}} \right) \end{aligned} \tag{2.13}$$

The quasi-steady lift, drag, and moment coefficients, $C_{L,\text{seg}}$, $C_{D,\text{seg}}$, and $C_{M,ac,\text{seg}}$, are calculated with an account of whether the segment is in a low-angle-of-attack or high-angle-of-attack regime, capturing the full ± 180 degree range in both angles. Effects of aspect ratio, stall, control surface deflection, bound vortices, and trailing vortices are all modeled, as detailed in [27]. As a basic sanity check, aerodynamic coefficient curves are plotted in Fig. 2.4 for McFoamy's main wing. The plotted coefficients of lift, C_L, drag, C_D, and moment, C_M, correspond to the mean aerodynamic chord, when no flap deflection is applied. The coefficient curves are clearly nonlinear over the full angle-of-attack range. In [95], Khan plots coefficient curves for various flap deflections and notes the similarities to data that has been collected for other flat plate UAV wings [26, 98]. Figure 2.5 provides further data on the lift and drag coefficients: the drag polar, which plots the lift and drag coefficients against each other, and the lift-to-drag ratio. These plots are a good indication of the aircraft's maneuvering capabilities. For one, we can consider the amount of thrust required for the aircraft to maintain steady, straight and level flight. By combining the two equilibrium equations – thrust equals drag, $T = D$, and lift equals weight, $L = W$ – we can deduce the required thrust for a given lift-to-drag ratio:

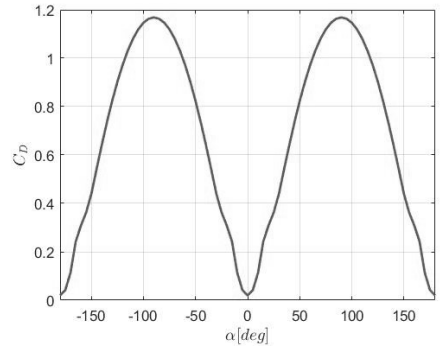
$$T_{\text{req}} = \frac{W}{\frac{L}{D}} = \frac{W}{\frac{C_L}{C_D}} \tag{2.14}$$

We note from Fig. 2.5b that the angle-of-attack for the most efficient cruising condition is approximately 5 degrees, where the lift-to-drag ratio is 5.95. From Eq. 2.14, we can calculate that the thrust required to maintain this cruise condition is 0.95 N, which is only 10% of the maximum thrust the propeller can produce, approximately 9.5 N.

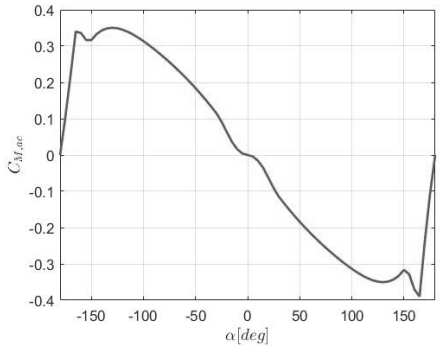
Continuing on from Eq. 2.13, we can finally compute the net forces and moments acting on the aircraft, including the effect of gravity, $\mathbf{g}_I = [0 \;\; 0 \;\; 9.81]^{\intercal} \text{m s}^{-2}$:



(a) Lift coefficient



(b) Drag coefficient

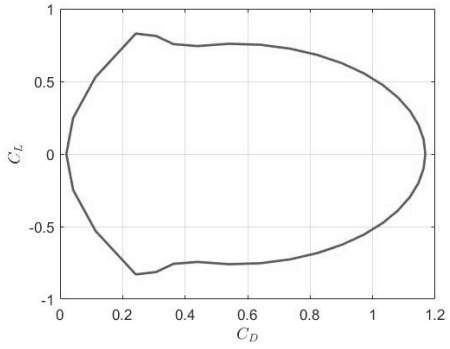


(c) Moment coefficient

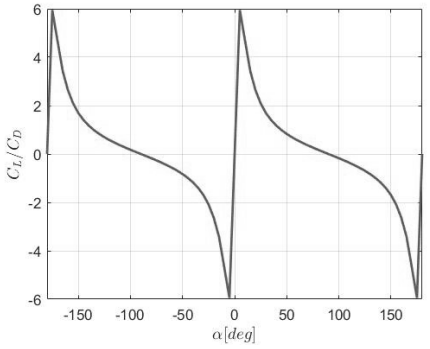
FIGURE 2.4: Aerodynamic coefficient curves for McFoamy's main wing.

$$\begin{aligned} \mathbf{F}_B &= \mathbf{C}_{BI}\mathbf{g}_I + \mathbf{F}_{B,\text{aero}} + \mathbf{F}_{B,T} \\ \mathbf{M}_B &= \mathbf{M}_{B,\text{aero}} + \mathbf{M}_{B,T} \end{aligned} \tag{2.15}$$

The full aircraft dynamics model is illustrated in block diagram form in Fig. 2.6.



(a) Drag polar



(b) Lift-to-drag ratio

FIGURE 2.5: Lift and drag coefficient data.

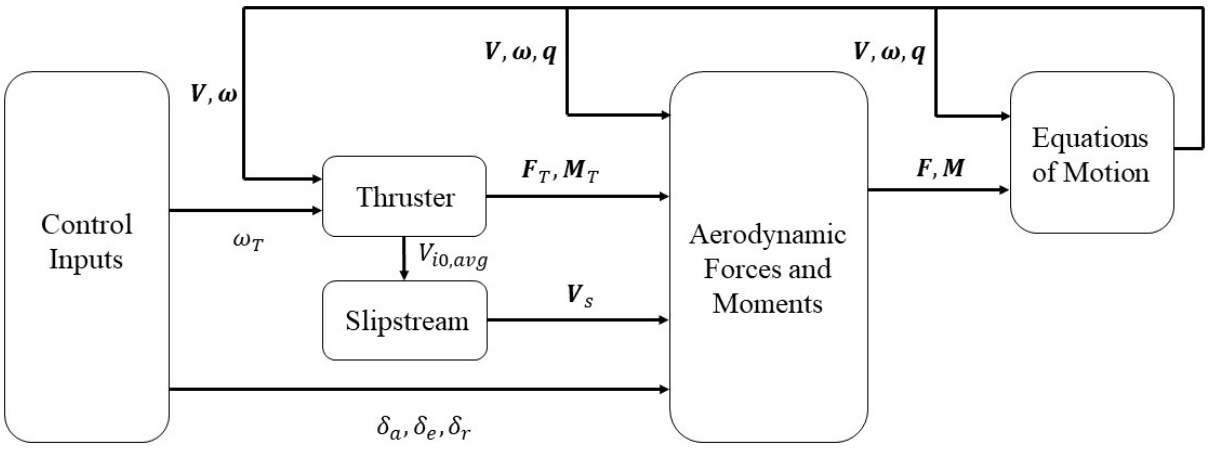


FIGURE 2.6: Block diagram of aircraft dynamics model.

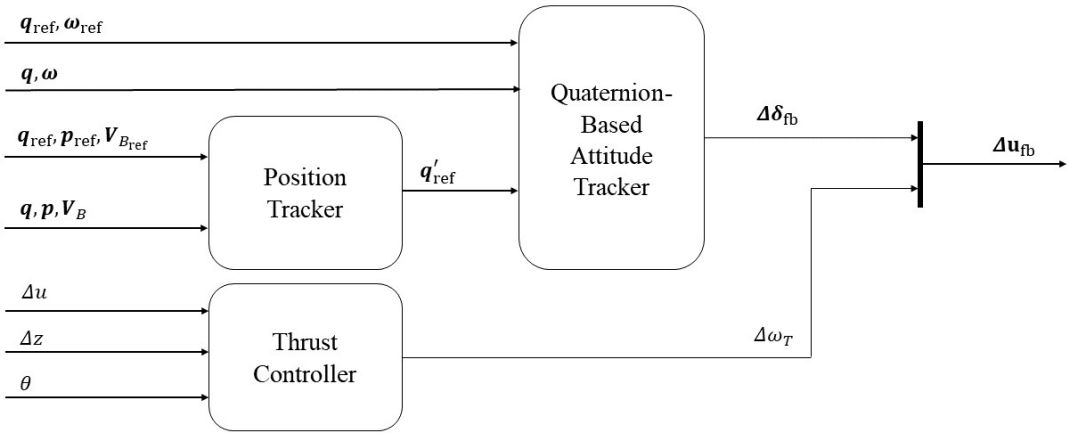


FIGURE 2.7: High-level overview of feedback controller.

2.2 Feedback Controller

In Chapter 4, feedforward control policies are generated for executing maneuvers. These feedforward control inputs must be complemented by a feedback controller to compensate for modeling inaccuracies, external disturbances, and measurement noise. This section summarizes the components of the largely physics-based feedback controller, initially developed and fully detailed in [99]. The controller was developed with the intent of creating a single set of control laws *and* gains that could be used to track any feasible trajectory. In achievement of these goals, the feedback controller is simple to tune, and enables robust transitions between maneuvers (there are no discontinuous transitions between control laws).

The controller consists of three main components: a position tracker, a quaternion-based attitude tracker, and a thrust controller. These components make up a modular architecture that addresses the inherent under-actuation problem of fixed-wing aircraft. The position tracker indirectly counteracts position errors by modifying the reference attitude, and the attitude tracker uses the control surfaces to track this modified reference attitude. The thrust controller's task is to regulate the aircraft's forward speed and altitude about their reference values. Figure 2.7 illustrates a high-level overview of the full feedback controller. The controller has been tested in hardware-in-the-loop simulations and flight text experiments for tracking a number of flight modes and agile maneuvers [99].

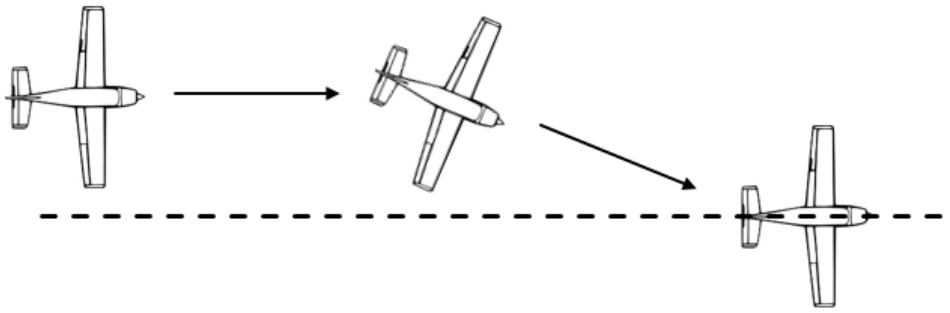


FIGURE 2.8: Feedback position tracking via reference attitude modification.

2.2.1 Position Tracker

Before being fed to the attitude tracker, the reference attitude passes through the position tracker. At any point in time, if the aircraft's actual position has deviated from the reference position, the position tracker will slightly reorient the reference attitude. In doing so, the direction of the propeller's thrust force (i.e. the direction of the body frame x axis) is used to indirectly diminish position errors. This general concept is illustrated in Fig. 2.8.

The position tracker performs two rotation operations on the reference quaternion, in order to enforce the shortest (cross product) rotation. Quaternion terms $\mathbf{q}_y$ and $\mathbf{q}_z$ are constructed from position and velocity errors, $\mathbf{e}_p$ and $\mathbf{e}_V$ (which are expressed in the desired frame, see Section 2.1.2):

$$\begin{aligned}
\mathbf{q}_y &= [\cos\frac{\Theta_y}{2},\ 0,\ -\sin\frac{\Theta_y}{2},\ 0]^\intercal \\
\mathbf{q}_z &= [\cos\frac{\Theta_z}{2},\ 0,\ 0,\ \sin\frac{\Theta_z}{2}]^\intercal \\
\Theta_y &= K_{p_p} e_{p_z} + K_{p_d} e_{V_z} \\
\Theta_z &= K_{p_p} e_{p_y} + K_{p_d} e_{V_y}
\end{aligned} \tag{2.16}$$

This is an incomplete attitude requirement, since only pitch and yaw are needed to completely define where the thrust vector points. In terms of the position tracker, the roll angle, by design, is not unique.

The reference position, $\mathbf{p}_{\text{ref}}$, and actual position of the aircraft (as measured on-board), $\mathbf{p}$, are known in the inertial frame, and thus must be converted to the desired frame:

$$\mathbf{e}_p = \mathbf{C}_{RI} \Delta \mathbf{p}_I, \tag{2.17}$$

where $\Delta\mathbf{p}_I = \mathbf{p}_{\text{ref}} - \mathbf{p}$. The rotation matrix, $\mathbf{C}_{RI}$, can be expressed in terms of the reference attitude quaternion (here, q_{rn} represents a component of the reference attitude quaternion, $\mathbf{q}_{\text{ref}}$):

$$\mathbf{C}_{RI} = \begin{bmatrix} q_{r1}^2 + q_{r2}^2 - q_{r3}^2 - q_{r4}^2 & 2(q_{r2}q_{r3} + q_{r1}q_{r4}) & 2(q_{r2}q_{r4} - q_{r1}q_{r3}) \\ 2(q_{r2}q_{r3} - q_{r1}q_{r4}) & q_{r1}^2 - q_{r2}^2 + q_{r3}^2 - q_{r4}^2 & 2(q_{r3}q_{r4} + q_{r1}q_{r2}) \\ 2(q_{r2}q_{r4} + q_{r1}q_{r3}) & 2(q_{r3}q_{r4} + q_{r1}q_{r2}) & q_{r1}^2 - q_{r2}^2 - q_{r3}^2 + q_{r4}^2 \end{bmatrix} \tag{2.18}$$

The velocity errors are computed as follows, where $\mathbf{V}_B$ is the measured body frame velocity of the aircraft:

$$\mathbf{e}_V = \mathbf{V}_{B_{\text{ref}}} - \mathbf{C}_{RI}\mathbf{C}_{IB}\mathbf{V}_B, \tag{2.19}$$

Note that $\mathbf{C}_{IB} = \mathbf{C}_{BI}^{\text{T}}$. The terms K_{p_p} and K_{p_d} in Eq. 2.16 are proportional and derivative gains. Using these rotation quaternions, a new reference attitude is generated and sent to the attitude tracker:

$$\mathbf{q}'_{\text{ref}} = \mathbf{q}_{\text{ref}} \odot \mathbf{q}_z \odot \mathbf{q}_y \tag{2.20}$$

The rotations $\mathbf{q}_y$ and $\mathbf{q}_z$ are limited to 45 degrees so that the controller will prioritize the reference attitude, as opposed to the modifications made to it by the position tracker.

2.2.2 Quaternion-Based Attitude Tracker

The quaternion-based attitude tracker actuates the ailerons, elevator, and rudder, to track the modified reference attitude, $\mathbf{q}'_{\text{ref}}$. As in [23], the attitude controller first computes an error quaternion, $\Delta\mathbf{q} = \mathbf{q}^* \odot \mathbf{q}'_{\text{ref}}$ - where $\mathbf{q}^*$ is the conjugate of $\mathbf{q}$ - and then converts it to three angular errors about the body frame axes, E_x, E_y, and E_z:

$$\begin{aligned} E_x &= 2\cos^{-1}(\Delta q_1)\frac{\Delta q_2}{||\Delta q_{2:4}||} \\ E_y &= 2\cos^{-1}(\Delta q_1)\frac{\Delta q_3}{||\Delta q_{2:4}||} \\ E_z &= 2\cos^{-1}(\Delta q_1)\frac{\Delta q_4}{||\Delta q_{2:4}||}, \end{aligned} \tag{2.21}$$

In Eq. 2.21, the nonlinear function on the vector component of the quaternion error linearizes around the angle variable of the axis-angle parametrization, which means that the controller reacts proportionally to angular errors.

Here, we introduce first-order angular errors about the body frame axes, which are computed from the commanded and actual angular rates, $\boldsymbol{\omega}_{B_{\text{ref}}}$ and $\boldsymbol{\omega}_B$:

$$\dot{\mathbf{E}} = \mathbf{C}_{IB}^{\mathsf{T}} \mathbf{C}_{RI}^{\mathsf{T}} \boldsymbol{\omega}_{B_{\text{ref}}} - \boldsymbol{\omega}_B \tag{2.22}$$

In much of the literature, the commanded control surface deflections are directly proportional to angular errors. This method neglects the dependency of the resulting control input on the local airflow over the flap's surface, i.e. that the control surfaces are much more effective at high speeds than at low speeds. To address this issue, an additional step is taken here. Instead of mapping control surface deflections directly to angular errors (and angular rate errors), we first calculate desired moment terms. Desired moments, ΔL, ΔM, and ΔN, are found from:

$$\begin{aligned} \Delta L &= (K_{a_p} E_x + K_{a_d} \dot{E}_x) I_x \\ \Delta M &= (K_{a_p} E_y + K_{a_d} \dot{E}_y) I_y \\ \Delta N &= (K_{a_p} E_z + K_{a_d} \dot{E}_z) I_z \end{aligned} \tag{2.23}$$

where K_{a_p} and K_{a_d} are proportional and derivative control gains, while I_x, I_y, and I_z are the aircraft's moments of inertia about its body axis. Equation (2.23) considers that more moment is needed to correct a given angular error about an axis with a large moment of inertia. Introducing the inertia terms into the equation makes it possible for the one proportional and one derivative gain to be appropriate for all axes.

In order to determine the control surface deflections that would be needed to obtain the desired moments, we first consider the basic aerodynamic relations between these variables [100]:

$$\begin{aligned} \Delta L &= \frac{1}{2}\rho v_s^2 Sb(C_{l_{\delta_a}}\Delta\delta_a + C_{l_{\delta_r}}\Delta\delta_r) \\ \Delta M &= \frac{1}{2}\rho v_s^2 Sc C_{m_{\delta_e}}\Delta\delta_e \\ \Delta N &= \frac{1}{2}\rho v_s^2 Sb C_{n_{\delta_r}}\Delta\delta_r, \end{aligned} \tag{2.24}$$

where v_s is the airspeed over the wing and S is the wing area. The first of these equations accounts for the effect of rudder deflection on the roll moment, since the rudder is not centered on the aircraft's roll axis. Rearranging the above gives the following:

$$\begin{aligned} \delta_{a_{\text{fb}}} &= \left(\frac{\Delta L}{\frac{1}{2}\rho v_s^2 Sb} - C_{l_{\delta_r}}\Delta\delta_r\right)\frac{1}{C_{l_{\delta_a}}} \\ \delta_{e_{\text{fb}}} &= \frac{\Delta M}{\frac{1}{2}\rho v_s^2 S\bar{c}C_{m_{\delta_e}}} \\ \delta_{r_{\text{fb}}} &= \frac{\Delta N}{\frac{1}{2}\rho v_s^2 Sb C_{n_{\delta_r}}}, \end{aligned} \tag{2.25}$$

where $\delta_{a_{\text{fb}}}$, $\delta_{e_{\text{fb}}}$, and $\delta_{r_{\text{fb}}}$ have been substituted in to denote that the deflections are feedback terms. The control derivative coefficients, $C_{l_{\delta_a}}$, $C_{l_{\delta_r}}$, $C_{m_{\delta_e}}$, and $C_{n_{\delta_r}}$ can be retrieved from the aircraft dynamics model. While these terms are not explicitly calculated in [95], they are determined for use here based on the experimental findings of that work.

The airspeed over the wing accounts for the slipstream effect. To be able to perform at real-time speeds, the slipstream model used in the control laws is simpler than that of Section 2.1.5. For this purpose, the slipstream is estimated using momentum theory [97]. Combined with the forward speed of the aircraft, the airspeed over the wing is calculated as:

$$v_s = \sqrt{u^2 + \frac{2T}{\rho A_{\text{prop}}}}, \tag{2.26}$$

where the propeller disk area is denoted by A_{prop}, and the thrust, T, is approximated as the commanded value (the combination of feedforward and feedback inputs). Recall that the aircraft's properties are listed in Table 2.1.

TABLE 2.3: Feedback controller gains.

Symbol	Value	Unit
K_{p_p}	0.08	$\mathrm{rad\,m^{-1}}$
K_{p_d}	0.1	$\mathrm{rad\,m^{-1}\,s^{-1}}$
K_{a_p}	180	$\mathrm{s^{-2}}$
K_{a_d}	8	$\mathrm{s^{-1}}$
K_{u_p}	3	$\mathrm{s^{-1}}$
K_{z_p}	5	$\mathrm{s^{-2}}$
K_{z_i}	0.5	$\mathrm{s^{-1}}$

2.2.3 Thrust Controller

The thrust feedback term, T_{fb}, is used to regulate the aircraft's forward speed, u, and altitude, z, about the desired trajectory. The following PI control law acknowledges that only the vertical component of thrust can influence the aircraft's altitude:

$$T_{\mathrm{fb}} = m\left(K_{u_p}\Delta u + (K_{z_p}\Delta z + K_{z_i}\int \Delta z\ dt)\sin\theta\right), \tag{2.27}$$

where the PI gains are K_{u_p}, K_{z_p}, and K_{z_i}. The thrust force must be converted to the control input: the rotational speed of the motor, ω_T. Because of the quadratic relationship between thrust and motor speed, the conversion can only be done after the feedforward thrust, T_{ff}, is computed (in Chapter 4):

$$\omega_T = \sqrt{\frac{T_{\mathrm{ff}} + T_{\mathrm{fb}}}{k_T}} \tag{2.28}$$

The thrust coefficient, k_T, is a function of the advance ratio, as detailed in [99]. All feedback gains were initially tuned in the hardware-in-the-loop simulator, mainly through trial and error, and then refined in flight tests. Some details on the tuning process are described in [99]. The gain values are listed in Table 2.3.

While it does have a position-tracking component, the feedback controller is not a path-following scheme, and has no intrinsic mechanism to adjust the reference path the aircraft is commanded. The control system as a whole attempts to achieve trajectory tracking, rather than path-following, and the rest of the thesis is concerned with creating time-dependent trajectories that the control system will track.

2.3 Optimization Framework

Chapters 3 and 4 make use of an optimal control framework that will be presented here in its most general form. In Chapter 3, such problems are solved to gain insight into the dynamics of the aircraft, and in Chapter 4, the problems are formulated to produce dynamically feasible trajectories and associated feedforward control inputs. In both cases, the optimization framework allows the full aircraft dynamics model to be exploited.

The general optimal control problem is stated as follows. The continuous time optimization solves for the time-dependent state and control vectors, $\mathbf{x}(t)$ and $\mathbf{u}(t)$, respectively:

$$\min J \triangleq \Phi[\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f] + \int_{t_0}^{t_f} \mathcal{L}[\mathbf{x}(t), \mathbf{u}(t), t]dt \tag{2.29}$$

subject to the dynamics constraints

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \tag{2.30}$$

the inequality path constraints

$$\mathbf{\Gamma}[\mathbf{x}(t), \mathbf{u}(t), t] \leq 0, \tag{2.31}$$

and the boundary conditions

$$\mathbf{\Theta}[\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f] = 0, \tag{2.32}$$

where Φ and $\mathcal{L}$ are the boundary and Lagrangian path cost functions.

The aircraft's state vector is $\mathbf{x} = [u,\ v,\ w,\ p,\ q,\ r,\ q_1,\ q_2,\ q_3,\ q_4,\ x,\ y,\ z]^\intercal$, and its control inputs are: ailerons, δ_a, elevator, δ_e, rudder, δ_r, and thrust, ω_T. For the optimizations, the aircraft's actual control inputs are added to the state vector, $\mathbf{x}$, and their derivatives make up the optimization problem's 'control vector', $\mathbf{u} = [\dot{\delta}_a, \dot{\delta}_e, \dot{\delta}_r, \dot{\omega}_T]^\intercal$. This is done because it allows the control input rates - in addition to the control inputs themselves - to be constrained in the problem definition. By imposing rate limits, we are also, at least partially, accounting for actuator dynamics, i.e. their lag in response to commanded inputs. The control inputs commanded to the aircraft in simulations and

flight tests are still the ailerons, elevator, rudder, and thrust (not their derivatives). As the aircraft sees it, the open-loop control policy determined via optimization is: $\mathbf{u}(t) = [\delta_a(t), \delta_e(t), \delta_r(t), \omega_T(t)]^\intercal$.

The optimization framework is used in various ways throughout Chapters 3 and 4, but the dynamic constraints never change, and they are equivalent to the aircraft dynamics model. Equation 2.30 can be expanded as:

$$
\begin{aligned}
\dot{u} &= \frac{F_x}{m} + 2g(q_2q_4 - q_1q_3) + rv - qw \\
\dot{v} &= \frac{F_y}{m} + 2g(q_3q_4 + q_1q_2) + pw - ru \\
\dot{w} &= \frac{F_z}{m} + g(q_1^2 - q_2^2 - q_3^2 + q_4^2) + qu - pv \\
\dot{p} &= \frac{I_zM_x + I_{xz}M_z - I_{xz}(I_y - I_x - I_z)pq - (I_{xz}^2 + I_z(I_z - I_y))qr}{I_xI_z - I_{xz}^2} \\
\dot{q} &= \frac{M_y - (I_x - I_z)pr - I_{xz}(p^2 - r^2)}{I_y} \\
\dot{r} &= \frac{I_{xz}M_x + I_xM_z - I_{xz}(I_y - I_x - I_z)qr - (I_{xz}^2 + I_x(I_x - I_y))pq}{I_xI_z - I_{xz}^2} \\
\dot{q}_1 &= -\frac{1}{2}(q_2p + q_3q + q_4r) \\
\dot{q}_2 &= \frac{1}{2}(q_1p + q_3r - q_4q) \\
\dot{q}_3 &= \frac{1}{2}(q_1q + q_4p - q_2r) \\
\dot{q}_4 &= \frac{1}{2}(q_1r + q_2q - q_3p) \\
\dot{x} &= u(q_1^2 + q_2^2 - q_3^2 - q_4^2) + 2v(q_2q_3 - q_1q_4) + 2w(q_1q_3 + q_2q_4) \\
\dot{y} &= 2u(q_2q_3 + q_1q_4) + v(q_1^2 - q_2^2 + q_3^2 - q_4^2) + 2w(q_3q_4 - q_1q_2) \\
\dot{z} &= 2u(q_2q_4 - q_1q_3) + 2v(q_3q_4 + q_1q_2) + w(q_1^2 - q_2^2 - q_3^2 + q_4^2) \\
\dot{\delta}_a &= u_{\delta_a} \\
\dot{\delta}_e &= u_{\delta_e} \\
\dot{\delta}_r &= u_{\delta_r} \\
\dot{\omega}_T &= u_{\omega_T},
\end{aligned}
\tag{2.33}
$$

where the thruster, slipstream, and aerodynamic components of the model are all housed in the force and moment terms, $\mathbf{F}_b = [F_x, F_y, F_z]^\intercal$, and $\mathbf{M}_b = [M_x, M_y, M_z]^\intercal$. Including the first-order dynamics of the aircraft's true control inputs allows constraints to be placed

TABLE 2.4: Control input saturation values.

Symbol	Value	Unit
$\delta_{a_{max}}$	42	deg
$\delta_{e_{max}}$	45	deg
$\delta_{r_{max}}$	46	deg
$\omega_{T_{min}}$	1716	rpm
$\omega_{T_{max}}$	6710	rpm
$\dot{\delta}_{a_{max}}$	258	deg/s
$\dot{\delta}_{e_{max}}$	430	deg/s
$\dot{\delta}_{r_{max}}$	430	deg/s
$\dot{\omega}_{T_{max}}$	10000	rpm/s

on the optimization's control vector: $\mathbf{u} = [u_{\delta_a}, u_{\delta_e}, u_{\delta_r}, u_{\omega_T}]^\intercal$. Unless otherwise specified, the control inputs and their derivatives are constrained by the physical limits of the aircraft's actuators. Each control surface is actuated using a servomechanism, and thus the maximum deflections and deflection rates correlate with the servo operation angular and rate limits. The motor's rotational speed limits correspond to the user-commanded pulse width modulation (PWM) signal range, and the motor rate limits employed were arrived at using conservative knowledge of the thruster's capabilities. These limits are all implemented as path constraints:

$$\begin{aligned} \delta_a &\in [-\delta_{a_{max}}, \delta_{a_{max}}], & \dot{\delta}_a &\in [-\dot{\delta}_{a_{max}}, \dot{\delta}_{a_{max}}] \\ \delta_e &\in [-\delta_{e_{max}}, \delta_{e_{max}}], & \dot{\delta}_e &\in [-\dot{\delta}_{e_{max}}, \dot{\delta}_{e_{max}}] \\ \delta_r &\in [-\delta_{r_{max}}, \delta_{r_{max}}], & \dot{\delta}_r &\in [-\dot{\delta}_{r_{max}}, \dot{\delta}_{r_{max}}] \\ \omega_T &\in [\omega_{T_{min}}, \omega_{T_{max}}], & \dot{\omega}_T &\in [-\dot{\omega}_{T_{max}}, \dot{\omega}_{T_{max}}], \end{aligned} \tag{2.34}$$

The saturation values are listed in Table 2.4. The thrust input, ω_T, measured in revolutions per minute (rpm), maps approximately quadratically to Newtons of thrust, where 1716 rpm $\approx$ 0 N and 6710 rpm $\approx$ 9.5 N for a stationary propeller. Unlike some other aerobatic UAVs, the thruster used here does not have the ability to reverse the propeller's direction of rotation.

At times, Euler angles are used to present a more intuitive formulation of a problem, however, it should be understood that their respective constraint equations are actually nonlinear functions of the quaternion, as per the following conversions:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan \frac{2(q_1 q_2 + q_3 q_4)}{1-2(q_2^2+q_3^2)} \\ \arcsin(2(q_1 q_3 - q_4 q_2)) \\ \arctan \frac{2(q_1 q_4 - q_2 q_3)}{1-2(q_3^2+q_4^2)} \end{bmatrix} \tag{2.35}$$

The unity norm constraint on the quaternion attitude representation, $q_1^2+q_2^2+q_3^2+q_4^2 = 1$, is also present, as a path constraint, in all versions of the optimization framework. What differs between optimal control problems presented throughout this thesis are the cost function, additional path constraints, and boundary conditions.

Apart from Chapters 3 and 4, there is one other way in which optimal control was used in this thesis. Appendix A describes a distinct reformulation of the problem that incorporates a feedback controller. This was done to optimize gains, and evaluate how effectively feedback control laws could be used to perform agile maneuvers. Valuable insight was gained from this work, but for reasons detailed in the appendix, it was not pursued further.

2.3.1 Optimal Control Problem Solver

There are a number of methods for solving optimal control problems. Our problems offer no analytical solutions, and must be solved numerically. Numerical methods for solving optimal control problems are split into indirect and direct methods. The indirect method employs the calculus of variations to solve boundary value problems. There has been software developed to implement the indirect method, however, solving boundary problems is normally extremely difficult, and thus direct methods have been more widely adopted. In the direct method, the states and controls are approximated by appropriate functions, and the problem is converted to a nonlinear programming problem (NLP) of the form:

$$\begin{aligned} &\min f(x) \\ &\text{subject to:} \\ &g(x) = 0 \\ &h(x) \leq 0 \end{aligned} \tag{2.36}$$

An explanation of the conversion can be found in [101]. The advantage of converting the problem to an NLP is that many well-developed software packages can easily solve these problems.

Every optimal control problem solved in this thesis is done so using GPOPS-II [102], which is MATLAB-based general purpose optimal control software. It employs a variable-order adaptive orthogonal collocation method. It first approximates the continuous-time optimal control problem as an NLP, using a number of points in the domain (called *collocation points*), and then calls upon the well-known NLP solver, SNOPT [103]. GPOPS-II was favored over other direct method software because of its ease of use, and that it is written in MATLAB. The software is general and works well for solving all the optimal control problems found throughout this thesis, which all adhere to the format expressed by Eqs. 2.29 - 2.32. The benefit of the software being MATLAB-based is that the aircraft model, and updates to the model, can easily be integrated into the problem. The major functions of the optimal control problems (for computing forces and moments) are converted to C MATLAB executable (MEX) files, which run almost two orders of magnitude faster than MATLAB script files. Most optimal control problems presented in this thesis are solved - using GPOPS-II - in a matter of seconds or minutes.

Chapter 3

Investigation of Sideslip and Slipstream in Extreme Maneuvering

This chapter uses the aircraft dynamics model of Section 2.1 to investigate the role of sideslip and propeller slipstream in the extreme maneuvering capabilities of agile fixed-wing UAVs. We identify the cost, in terms of performance loss, if either of these two phenomena are not accounted for in maneuver design. The entire investigation is the author's original work.

3.1 Motivation

Among other characteristics of the design of agile fixed-wing UAVs that allow for improved agility and extreme maneuvering, is the effect of the propeller's slipstream, and the ability to induce large sideslip angles. The most impressive demonstrations of the maneuvering capabilities of these aircraft are exhibited during RC aerobatics competitions, where these two phenomena are clearly seen at play. These phenomena have been exploited in some of the research on planning and control, even if not explicitly modeled. For instance, once in a (stationary) nose-up hover, the control mechanism of an agile fixed-wing UAV becomes similar to that of a ducted fan aircraft (e.g. the RQ-16 T-Hawk [104]), wherein all control authority of the flaps would be lost if not for the slipstream's energetic flow. Many fixed-wing maneuvers that have been researched involve sideslip angles, such as the skid-to-turn [105], and even an aggressive perching variant [106]. For some maneuvers, such as knife-edge flight [52], the inclusion of sideslip is obvious and inevitable. For others, such as aggressive turns [47], the utility of sideslip is less apparent.

This chapter evaluates the effect of sideslip and gathers insight as to its utility, or lack thereof, in extreme maneuvering. We investigate whether performance is benefited by not constraining sideslip to be zero - which might, for instance, be done in the interest of controllability and stability. To the best of our knowledge, the literature on these aircraft have yet to include a systematic study of the significance of sideslip and slipstream in maneuvers. Having access to a dynamics model that comprehensively includes these two phenomena presented a unique opportunity to gather valuable insight.

3.2 Analysis Methodology

Considering the number of extreme (e.g. aerobatic) maneuvers that agile fixed-wing UAVs are able to perform, a comprehensive analysis of the use of sideslip and slipstream in each maneuver is unrealistic. Instead, a few distinct maneuvers are investigated. The analysis is framed as a series of questions relating to maneuver performance, wherein each question is answered by means of trajectory optimization. The generic optimization framework is described in Section 2.3, and here we further specify the problem definitions. For each optimization problem, we first compute the solution using the complete dynamics model, and then use this result as an initial guess in solving for the cases where sideslip and/or the slipstream effect are neglected. The investigation is conducted by comparing the maneuvers that result from accounting for these phenomena versus those that result from ignoring them. We quantify the significance of sideslip and slipstream by analyzing the disparity in answers depending on which of the two phenomena are incorporated into the optimization problem.

Neglecting sideslip means to constrain it to be zero in the optimization, i.e. $\beta = 0$, or equivalently $v = 0$. The swirling flow from the thruster is modeled as a reduction in reaction torque, and this reduction in torque is retained in the zero-sideslip case, since the swirling flow is assumed to be unaffected. The slipstream can be neglected by removing the axial airflow term, V_s (see Eq. 2.8), from the dynamics model, and removing the reduction in the thruster reaction torque that was introduced to account for the thruster-induced swirling flow.

3.3 Maneuver Performance Investigation

In this investigation we tailor the optimization framework of Section 2.3 to address a number of problems relating to maneuver performance, and discuss the results.

3.3.1 What is the slowest speed the aircraft can fly while maintaining a constant altitude?

The cost function for this problem integrates speed over time:

$$J = \int_0^{t_f} V dt \tag{3.1}$$

The following path constraints are implemented to enforce steady straight and level flight:

$$[\dot{u}, \dot{v}, \dot{w}, p, q, r, \phi, \dot{\theta}, \psi, y, z]^\intercal = \mathbf{0} \tag{3.2}$$

Constant control input action, $\mathbf{u} = \mathbf{0}$, is also enforced. Recall that $\mathbf{u}$ represents the rates of the actual control inputs. Where state derivatives appear in constraints, they denote their respective nonlinear functions from the equations of motion, Eq. (2.33). The only boundary condition is on the final time, $t_f = 10$ s. Specifying the final time helps the solver converge on a solution, but the exact value of 10 s is ultimately arbitrary.

It should be noted that because we have solved for equilibrium conditions rather than time-varying trajectories, the solutions are independent of the cost function, i.e the solution depends only on the dynamics, and thus a lower speed could not be obtained by employing a different cost function.

Using the full dynamics model to solve the problem, results showed that the aircraft is effectively able to fly as slowly as is desired, operating well into the post-stall region. For instance, at an 80° pitch angle, the aircraft will fly straight and level at 0.8 $\mathrm{m\,s^{-1}}$. At this speed, the elevator deflection is approximately -10°, and the motor speed is also far from saturation, at 4730 rpm, producing 4.95 N of thrust. The slowest speed that the aircraft could fly without accounting for the slipstream was 5.62 $\mathrm{m\,s^{-1}}$, with a motor speed of 4190 rpm producing 3.02 N of thrust. In this condition, the elevator is fully saturated and holding a 40° angle-of-attack. Without the additional flow of the slipstream, the aerodynamic forces generated by the elevator deflection are lesser. The aircraft cannot maintain steady-state flight at any slower of a speed because the elevator would be unable to balance out the aerodynamic moments to zero.

This case exemplifies the very large benefit that can be derived from the propeller slipstream; allowing the aircraft to perform a very extreme maneuver; and such maneuvers have been routinely demonstrated by RC pilots.

TABLE 3.1: Final times and costs of heading reversals

Case	$\mathbf{t_f}$**[s]**	**Cost**
Full model	1.90	18.18
No sideslip	2.03	21.19
No slipstream	4.00	284.81
Neither	4.01	302.61

3.3.2 How much space and time is required to reverse the aircraft's heading?

The cost function now penalizes the final time and displacements in all three dimensions of space:

$$J = w_1 t_f + w_2 \int_0^{t_f} (x^2 + y^2 + z^2) dt, \tag{3.3}$$

where $w_1 = 1\ \mathrm{s}^{-1}$ and $w_2 = 1\ \mathrm{m}^{-2}\,\mathrm{s}^{-1}$. The weighting terms are set to unity because we started with these values and found no reason to alter them. This rationale extends to other parts of the thesis in which weights are set to one. The cost function separates x, y, and z instead of calculating a volume, because a one- or two-dimensional maneuver (with zero volume) would still take up space and should therefore not have zero cost. The only path constraints are for the general saturation of the control inputs and their rates, see Eq. 2.34. Boundary conditions at $t = 0$ define straight and level flight at the desired speed, which in this case is $V = 7\ \mathrm{m\,s}^{-1}$ (a typical cruising speed for the testbed aircraft). Therefore, at $t = 0$,

$$[v, p, q, r, \phi, \psi, x, y, z, \delta_a, \delta_r]^\intercal = \mathbf{0}, \tag{3.4}$$

and u, w, θ, δ_e, and ω_T take on their trim conditions for straight and level flight. The boundary conditions at $t = t_f$, where t_f is a free variable, are the same (including $V_f = 7\ \mathrm{m\,s}^{-1}$) except that $\psi(t_f) = \pi$. Note that the aircraft must end the maneuver at the same position it began from. In Chapter 4, we discuss how useful a turn-around maneuver like this is for motion planning in obstacle-dense environments.

The path solutions for the four cases are plotted in Fig. 3.1: the full model, no sideslip, no slipstream, and neither sideslip nor slipstream. Additionally, Table 3.1 lists the values of the final time and cost for each trajectory. Even though the final times of each maneuver represent a relatively small portion of the cost, it is noteworthy that there is more than a

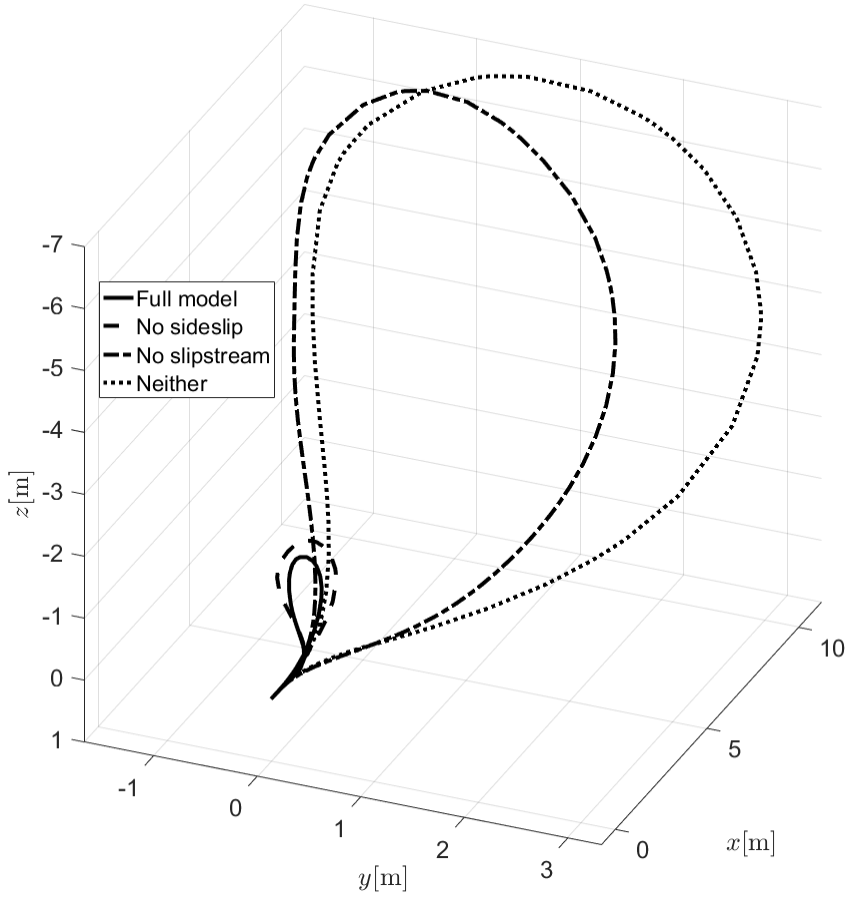


FIGURE 3.1: Heading reversal paths.

twofold difference between the first and last cases. The results show that a small penalty to the optimality of the maneuver is incurred by constraining sideslip, and a much larger penalty arises from neglecting the modeling of the slipstream.

The utility of sideslip in this maneuver is two-fold. First, the sideslip angle points the nose inwards of the turn, creating a component of the thruster force that acts laterally, in the direction the aircraft is turning. Sideslip also has the effect of generating aerodynamic forces off of the aircraft's body, which, about halfway through the maneuver, help it slow down and change heading.

The effect of the slipstream is noticeably more significant. For one, the additional airflow of the slipstream greatly increases the maximum aerodynamic forces each control surface can produce; thus increasing their control authority. A closer investigation of the results also reveals that the aircraft is having to fly faster, on average, without the slipstream. The mean speed of the maneuver generated with the full model is 4.7 $\mathrm{m\,s^{-1}}$, compared to 8.1 $\mathrm{m\,s^{-1}}$ without the slipstream. An increase in speed is needed to produce more airflow

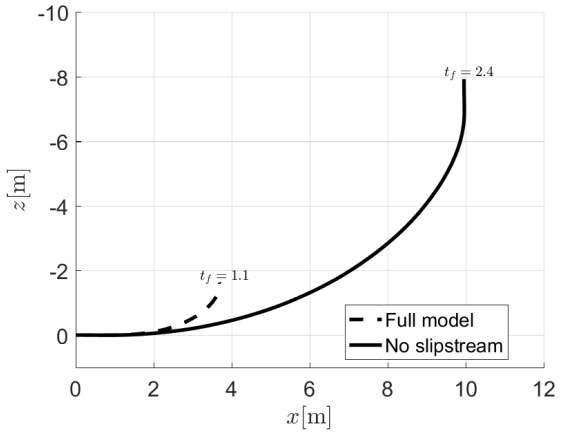


FIGURE 3.2: Transition paths from wings-level to hover.

over the control surfaces. Each control surface becomes saturated at one point or another during the maneuver, and needs enough airflow to produce aerodynamic forces capable of aggressively turning the aircraft around.

3.3.3 How much space and time is required to put the aircraft into a hover?

The cost function is

$$J = w_1 t_f + w_2 \int_0^{t_f} (x^2 + z^2) dt \tag{3.5}$$

Once again, $w_1 = 1\ \mathrm{s}^{-1}$ and $w_2 = 1\ \mathrm{m}^{-2}\,\mathrm{s}^{-1}$. Lateral displacement, y, is now placed in the path constraints, $[v, p, r, \phi, \psi, y] = \mathbf{0}$. These constraints force the aircraft to operate in the vertical plane, keeping the problem tractable and relevant to the existing literature on pitch up transitions into hover. The initial boundary conditions again define straight and level flight at $V = 7\ \mathrm{m\,s}^{-1}$. The final boundary condition is set to a stationary nose-up hover, i.e. $[u, w, q, \theta] = [0, 0, 0, 90°]$.

Given the nature of the hover transition as per the problem definition, sideslip is irrelevant ($v = 0$). The use of the slipstream, however, is clearly not, as illustrated in Fig. 3.2. We see that both space and time needed to transition into a hover are penalized when the slipstream is neglected. The cost function value associated with the full model trajectory is only 12.06, in contrast to a cost of 200.37 in the case without slipstream. The reasons for these disparities are very similar to those stated for the previous maneuver. Without

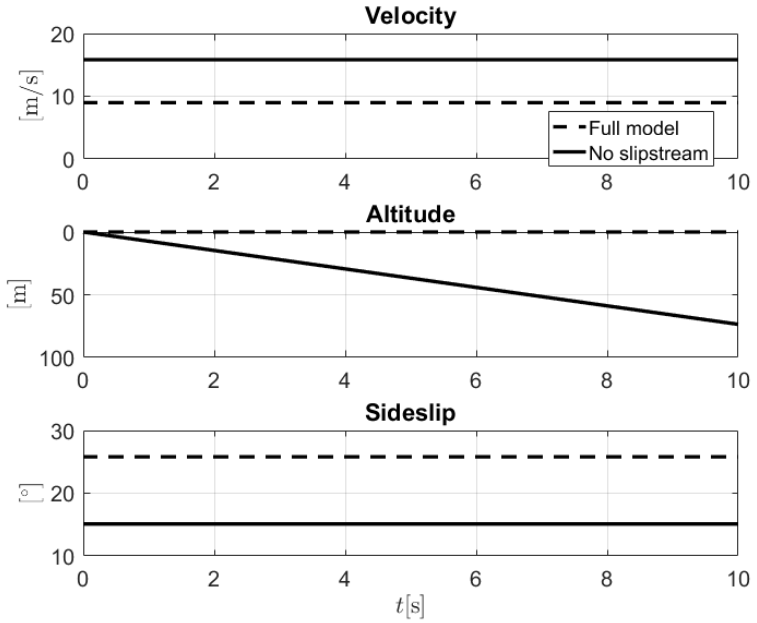


FIGURE 3.3: Velocity, altitude, and sideslip angle in knife-edge flight.

the slipstream accounted for, the aircraft first increases its speed as it begins to pitch up (thus causing a large gain of altitude), whereas the speed only decreases when the full model is employed. In either case, full elevator deflection is used throughout most of the maneuver, which is less effective at pitching the aircraft upwards when the slipstream is omitted.

3.3.4 Can the aircraft hold a constant altitude in knife-edge flight?

We first ask whether such a maneuver is possible at any (constant) velocity. The cost function penalizes deviations in altitude:

$$J = \int_0^{t_f} z^2 dt \tag{3.6}$$

Constant control input action is forced in the path constraints, as in Section 3.3.1. The other path constraints force straight and steady knife-edge flight: $\phi = 90°$ and $[\dot{u}, \dot{v}, \dot{w}, p, q, r, \psi, y] = \mathbf{0}$. Also as in Section 3.3.1, the only boundary condition is on the final time: $t_f = 10$ s.

To fly a knife-edge (90° roll angle) at a constant altitude, the aircraft must counterbalance its weight with aerodynamic and/or thruster forces. By virtue of its conventional layout, our aircraft can only generate vertical components from these forces by tilting its nose upwards. In doing so, the velocity vector of the aircraft, which remains in the horizontal plane to maintain altitude, will have components along the body-frame x- and y-axes. The maneuver, therefore, necessarily utilizes sideslip.

Figure 3.3 plots the results of solving with and without the slipstream modeled. Without the slipstream modeled, the aircraft cannot maintain a constant altitude, even without constraints on the velocity. The rudder saturates, and without the additional control authority provided by the slipstream, is unable to keep the aircraft from descending. This observation is analogous to the case of straight and level flight in Section 3.3.1, in that no rudder input is enough to zero the net moment on the aircraft. With the full model, however, the aircraft can in fact perform a straight, constant velocity and altitude knife-edge maneuver. In this case, the aircraft is flying at $V = 9.0\ \mathrm{m\,s^{-1}}$, using full rudder but less than maximum throttle, $\omega_T = 4990$ rpm. Note that the 26° sideslip angle is equivalent to the aircraft's pitch, since altitude is constant. Being able to maintain straight and level knife-edge flight expands the scope of utility of the maneuver. In Section 5.2, we exploit this maneuverability to integrate the knife-edge maneuver into motion planning.

With this knowledge that the knife-edge maneuver can be performed at *some* velocity (using the full model), we then re-frame the problem to identify the full velocity range for which it is feasible. The altitude is moved from the cost function into the path constraints, such that $z = 0$. Two problems are then solved, one to find the minimum velocity, with

$$J = \int_0^{t_f} V dt, \tag{3.7}$$

and another to find the maximum velocity, with

$$J = \int_0^{t_f} \frac{1}{V} dt \tag{3.8}$$

As in Section 3.3.1, the minimum and maximum speeds are inherent to the aircraft's dynamics, and independent of the cost function used to find them. The results reveal that the full model solution in Fig. 3.3, where the rudder is saturated and $V = 9.0\ \mathrm{m\,s^{-1}}$, is the minimum velocity case. The maximum velocity for which straight and constant altitude knife-edge flight can be maintained is $V = 10.8\ \mathrm{m\,s^{-1}}$. In this case, the rudder

deflection has room to spare, at 33°, but maximum throttle is used. The sideslip angle is 21°.

3.4 General Remarks

All findings of the analysis showed that ignoring sideslip and slipstream in the modeling will lead to sub-optimal results, and that this is especially true of the slipstream effect. While the comprehensive aircraft model could be used to determine steady level flight at any slow speed, the version of the model that omitted the slipstream could not be used to find an equilibrium state at any speed under 5.62 $\mathrm{m\,s^{-1}}$. An aggressive heading-reversal maneuver took up approximately twice as much space when sideslip was constrained to be zero, but over 100 times more space when the slipstream model was left out. To transition into a hover without the slipstream required twice the forward distance and time, and four times as much vertical displacement. With and without the slipstream effect, these two aggressive maneuvers saw 15- and 16-fold differences, respectively, in cost function values (the metric used to evaluate performance). It was also found to be impossible to generate a constant altitude knife-edge condition without modeling the slipstream; at best it could maintain a high velocity of 15.8 $\mathrm{m\,s^{-1}}$ while descending at 7.4 $\mathrm{m\,s^{-1}}$. With the slipstream modeled, however, the maneuver was found to be possible within a range of speeds from 9.0 $\mathrm{m\,s^{-1}}$ to 10.8 $\mathrm{m\,s^{-1}}$. When designing maneuvers, one should be aware of the significance of these phenomena, and of the penalties incurred if the methods are reliant on sideslip constraints or simplified modeling techniques that neglect slipstream effects.

Chapter 4

Maneuver Space and Motion Primitives

This chapter presents the aircraft's maneuver space, which acts as a hybrid representation of the aircraft's dynamics, in place of the nonlinear ordinary differential equations. It is constructed off-line, using the aircraft dynamics model of Section 2.1, as a collection of maneuvers that can be accessed by the motion planner. Its purpose is to allow the planning algorithm to generate dynamically feasible trajectories without having to solve complex dynamic constraints in real-time. Instead, it simply has to choose maneuvers out of the set.

The maneuver space consists of a finite number of motion primitives, where motion primitives are dynamically feasible trajectories and their associated feedforward control inputs. The set of primitives was designed to be large enough to represent a significant portion of the aircraft's flight envelope, yet suitably compact to fit in the limited computational resources available on the aircraft's autopilot system (as described in Section 6.1).

Motion primitives are classified here as either trim primitives or agile maneuver primitives. Trim primitives are steady maneuvers with constant control inputs that can be coasted along indefinitely. Agile maneuver primitives are finite-time transitions that accomplish a specific, purposeful change in the aircraft's pose.

The motion primitives are generated via optimization, solving optimal control problems to produce reference state trajectories and their corresponding feedforward control inputs. This approach to trajectory generation avoids the issues associated with learning from piloted flight (e.g. [14, 39]): relying on the competence of a pilot, the inconsistency inherent in manual flight, and the sub-optimality. Additionally, the trajectories are not

corrupted by external disturbances (e.g. from wind) or state estimation errors. A further benefit of this method is that specific functional objectives can be incorporated into the maneuver design by minimizing a cost function, and imposing constraints and boundary conditions.

The general optimization framework is set up and solved in the same way as was described in Section 2.3. The specifics of the optimal control problem formulations - cost function, path constraints, and boundary conditions - are particular to each motion primitive and will be described here.

The maneuver space is bound to a specific flight velocity, and this is consistent with the motion planner. Many applications of automated flight would not require speed to change throughout the task. Incorporating velocity into the planning problem would make it considerably more complex, by increasing the dimension of the configuration space by one. With that said, we do see the value in future work exploring more complex planning algorithms that can choose from maneuvers with different flight speeds. It may be useful, for example, to proportion the flight speed to the density of obstacles in an area; slowing the aircraft down in highly constrained areas, and allowing it to fly at the most energy-efficient speed otherwise. A move was made in this direction by developing a methodology for parametrizing maneuvers by speed, using a novel application of Dynamic Time Warping. The methodology makes the maneuver space more robust while incurring a negligible cost to the computational load needed in flight. It is strategically split into off-line and on-line portions, such that the storage requirements and computational burden on the flight controller are minimal.

As this chapter proceeds, the generation of trim primitives, agile maneuver primitives, transitions, and a knife-edge maneuver are discussed in detail. The velocity parametrization methodology is then described, and a final note on the implementation of the maneuver space is given.

4.1 Trim Primitives

Trim primitives are steady maneuvers ($[\dot{u}, \dot{v}, \dot{w}, \dot{p}, \dot{q}, \dot{r}, \dot{\phi}, \dot{\theta}] = \mathbf{0}$) with constant control inputs that can be held indefinitely. They are the aircraft's basic flight modes, and the set of trim primitives make up the greater portion of the maneuver space. These primitives can be used by the motion planner for any length of time, since they are steady, and the feedforward control inputs needed to hold them remain constant. The trim primitives included in the maneuver space are:

TABLE 4.1: Control input saturation values that are 80% of the aircraft's actual limits

Symbol	**Value**	**Unit**
$\delta_{a_{max}}$	34	deg
$\delta_{e_{max}}$	36	deg
$\delta_{r_{max}}$	37	deg
$\omega_{T_{min}}$	1716	rpm
$\omega_{T_{max}}$	5368	rpm

- straight and level flight
- climbs/descents
- banked turns
- helical banked turns
- hover

Trim primitives can be used by the planner for any length of time, since the feedforward control inputs needed to hold them remain constant. Trim primitives can be solved for in various ways; for example, using MATLAB's *fsolve* function, or a bifurcation analysis [107]. With the exception of hover, all trim primitives are found here by solving a trajectory optimization problem. The exact form of the cost function used to solve for the trim primitives is not relevant, except to minimize actuation if there are multiple sets of control inputs that can be used to achieve the same trim condition.

$$\min \; J \triangleq \int_0^{t_f} \left(w_1 \delta_a^2 + w_2 \delta_e^2 + w_3 \delta_r^2 + w_4 \omega_T^2 \right) dt$$

subject to the first-order dynamics of the aircraft (Eq. 2.1),

and the path constraints:

$$\begin{aligned} V = V_d, \quad \dot{\phi} = \dot{\theta} = 0, \quad \dot{\psi} = \dot{\psi}_d, \quad \dot{z} = \dot{z}_d \\ \delta_a \in [-\delta_{a_{max}}, \delta_{a_{max}}], \qquad \dot{\delta}_a = 0^\circ \, \mathrm{s}^{-1} \\ \delta_e \in [-\delta_{e_{max}}, \delta_{e_{max}}], \qquad \dot{\delta}_e = 0^\circ \, \mathrm{s}^{-1} \\ \delta_r \in [-\delta_{r_{max}}, \delta_{r_{max}}], \qquad \dot{\delta}_r = 0^\circ \, \mathrm{s}^{-1} \\ \omega_T \in [\omega_{T_{min}}, \omega_{T_{max}}], \qquad \dot{\omega}_T = 0 \text{ rpm/s} \end{aligned} \tag{4.1}$$

The weights in the cost function are set to $w_1 = w_2 = w_3 = 1$ rad^{-2} s^{-1} and $w_4 = 1.56 \times 10^{-8}$ rpm^{-2}s^{-1}]. The last weight is much smaller than the others due to the mismatch in units; we want the penalty on thrust to be roughly similar in magnitude to that on the control surface inputs so that it does not dominate the cost.

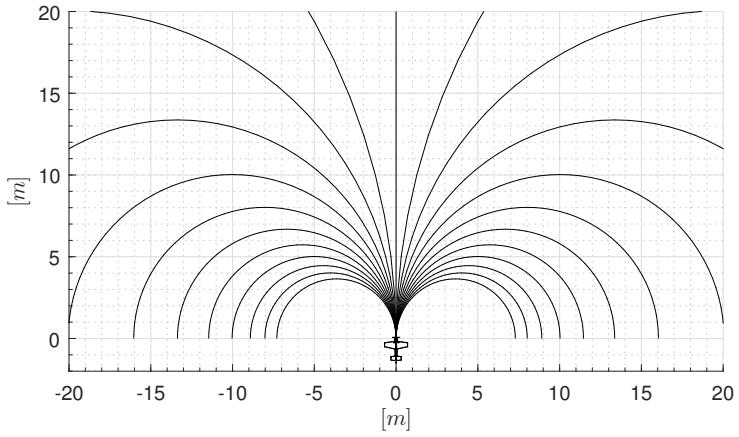
The control inputs may take on any value within their physical limits, but constraints are put on their derivatives so that they remain constant. The saturation values of the control inputs themselves are set to 80% of the actual limits of the McFoamy aircraft (see Table 2.4), to ensure there is room for feedback control. The new maximum values are listed in Table 4.1. The desired speed, V_d, is set to a constant 7 $\mathrm{m\,s^{-1}}$, which is a normal cruising speed for the aircraft. The planner is developed for constant speed flight, except when hovering and when performing agile maneuvers. The roll and pitch rates, $\dot{\phi}$ and $\dot{\theta}$, are set to be zero so that the maneuvers are steady. Because the maneuvers are steady, there are no boundary conditions to satisfy. With the exception of hover, each trim primitive is obtained for different combinations of the desired yaw rate, $\dot{\psi}_d$, and climb/descent rate, $\dot{z}_d$, as per the following, where $\dot{\psi}_c$ and $\dot{z}_c$ represent non-zero constants:

- straight and level flight: $\dot{\psi}_d = 0$, $\dot{z}_d = 0$
- climbs/descents: $\dot{\psi}_d = 0$, $\dot{z}_d = \dot{z}_c$
- banked turns: $\dot{\psi}_d = \dot{\psi}_c$, $\dot{z}_d = 0$
- helical banked turns: $\dot{\psi}_d = \dot{\psi}_c$, $\dot{z}_d = \dot{z}_c$

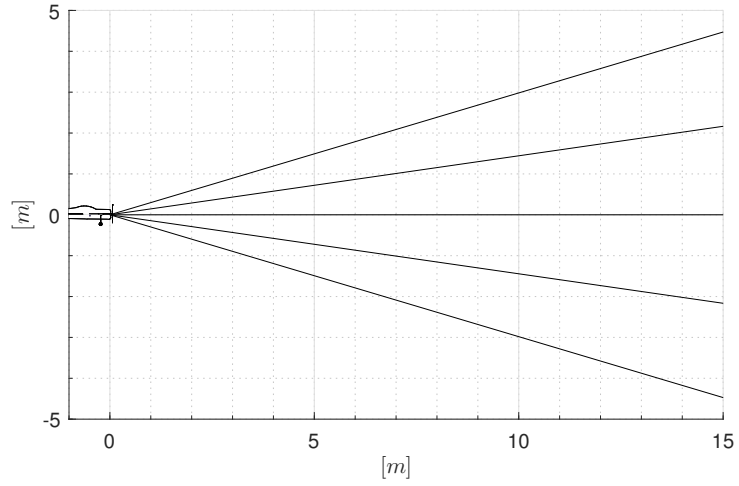
As mentioned, the planner is developed for constant speed flight, except when hovering and when performing agile maneuvers. Fixed-wing aircraft can maximize their range and endurance predictably at specific flight speeds, and the aircraft would navigate close to some such suitable speed. Operating at a specific flight speed is consistent with the objective of the motion planner, which is to guide the aircraft to a desired goal region. The objective does not, for instance, include arrival time constraints. While further agility could be harnessed by loosening constraints on the flight speed, doing so would increase the size of the maneuver space and presumably result in a more difficult trajectory tracking problem.

For the sake of preserving computational resources, we keep the maneuver space compact by solving for primitives in incremental values. For the yaw rate, $\dot{\psi}_c$, we find primitives from -110° $\mathrm{s^{-1}}$ to 110° $\mathrm{s^{-1}}$ in increments of 10° $\mathrm{s^{-1}}$, and for the climb/descent rate, $\dot{z}_c$, we sample from -2 $\mathrm{m\,s^{-1}}$ to 2 $\mathrm{m\,s^{-1}}$ in increments of 1 $\mathrm{m\,s^{-1}}$. There are a total of 116 trim primitives; straight and level flight at 7 $\mathrm{m\,s^{-1}}$, 4 climbs/descents, 22 banked turns, 88 helical banked turns, and the hover. While smaller incremental values could have been used at the cost of increased memory consumption, the simulations and flight test experiments of Chapter 6 confirmed that this number of primitives sufficiently covers the aircraft's flight envelope such that it can effectively traverse obstacle-dense environments.

The resulting maneuver space is depicted in Fig. 4.1. The lines in the figures represent the dynamically feasible paths associated with each trim primitive, which can extend



(a) Top-down view, each line corresponds to a different value of $\dot{\psi}$



(b) Side view, each line corresponds to a different value of $\dot{z}$

FIGURE 4.1: Trim primitive maneuver space.

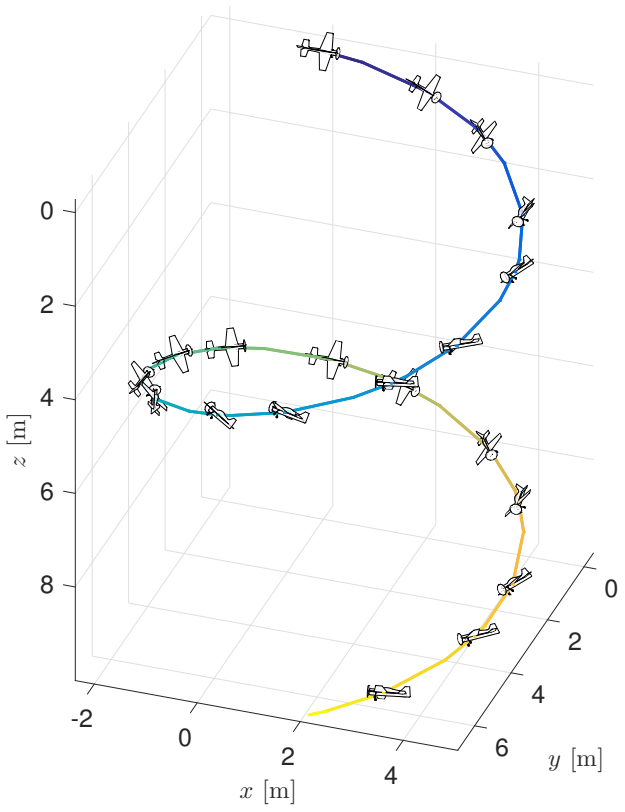


FIGURE 4.2: A helical turn trim primitive, where $\dot{\psi} = 110^\circ\,\mathrm{s}^{-1}$ and $\dot{z} = 2\ \mathrm{m\,s}^{-1}$. The aircraft is drawn approximately to scale.

indefinitely. For banked turns and helical banked turns - which are circular - the smallest turn radius available in the maneuver space is 3.65 m, and the largest turn radius is 40.1 m. We can compare these turn radii to other minimum turning radii found in the literature on simulations and experiments of automated flight with small fixed-wing UAVs. In [108], a path-following algorithm implements a minimum turning radius of 45 m in simulations. In [77], simulations of a small fixed-wing UAV include turns with a minimum radius of 15 m. Fixed-wing simulation results for a trajectory generation and control methodology used 8 m radius Dubin's turns in [92].

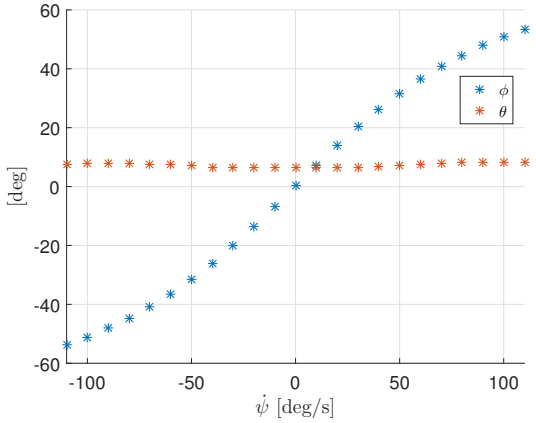
To showcase an example, Fig. 4.2 illustrates the trajectory of one of the most aggressive trim primitives, for which $\dot{\psi} = 110^\circ\,\mathrm{s}^{-1}$ and $\dot{z} = 2\ \mathrm{m\,s}^{-1}$. The plot shows the trajectory spanning over 5 seconds - the direction illustrated by the path transitioning in colour from blue to yellow - but the trim primitives all have arbitrary end times that can be dictated by the motion planning algorithm. The steady-state roll and pitch angles are 50.4° and 0.96°, respectively. The constant control inputs are $[\delta_a, \delta_e, \delta_r, \omega_T] = [-2.58^\circ, -33.45^\circ, -5.08^\circ, 5336\ \mathrm{rpm}]$. Whilst the aileron and rudder input

are relatively small, the elevator and thrust input are nearing saturation (i.e. 80% of the aircraft's actual limits).

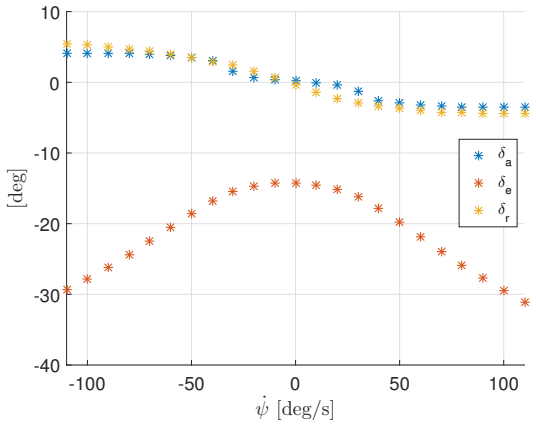
Figure 4.3 uses the trim primitive data to show how the states and controls evolve as the aircraft turns more aggressively in either direction. Each marker on the plots represents a trim primitive associated with the corresponding turn rate (on the x axis), and zero climb rate. The findings are similar for the other, non-zero climb/descent rates. From one extreme turn to its opposite, the roll angle varies greatly, from approximately -55 to 55 degrees, while the pitch angle always hovers between 13 and 15 degrees. As observed in Fig. 4.3b, very little rudder and aileron are needed to hold a banked turn. Even the most aggressive turns need barely more than a 5 degree input in either control surface. As discussed in Chapter 3, the slipstream has a significant impact on the effectiveness of the control surfaces. This is especially true of the ailerons, because they are closest to the propeller and have a large surface area. To maintain the steady state maneuvers, the elevator deflections range more vastly during turns, from approximately -15 degrees during straight and level flight to -30 degrees during either of the most aggressive turns. The thrust also notably increases with sharper turns. All these trends remain true for the cases where the aircraft is turning with non-zero climb/descent rates.

Assembling the information in these plots uncovers an interesting observation. As the aircraft deviates from $\dot{\psi} = 0$, the ailerons and rudder change slightly, while the elevator and thrust change more significantly, but in a symmetrical fashion, i.e. a -20 degree turn requires approximately the same elevator and thrust input as a 20 degree turn. This means that greater changes in roll angle and turn rate do not necessarily correlate with greater changes in all the control inputs.

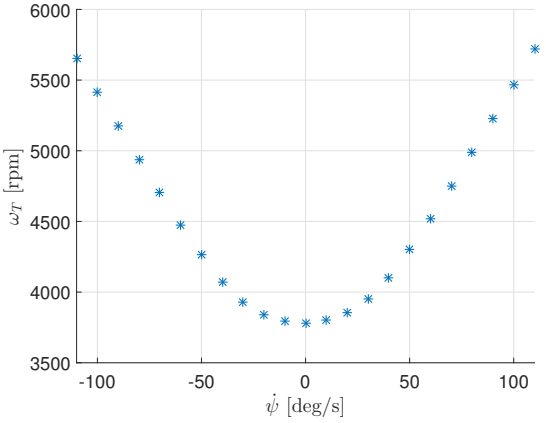
The one trim primitive yet to be discussed is the hover. The hover primitive is determined via solution of the aircraft's equations of motion. The only variable we solve for is the value of thrust that achieves an equilibrium state when the pitch angle is 90 degrees. For the McFoamy aircraft, this value happens to be 5334 rpm. We note that, as per the dynamics model, the equilibrium condition is *actually* achieved with a combination of all control inputs, not just thrust. This is because the aircraft isn't perfectly symmetrical, and the spinning propeller produces an aerodynamic torque and a swirl effect. However, the control surface inputs that theoretically produce the equilibrium state are negligibly small (less than 3 degrees). Combining this with the fact that the hover equilibrium is unstable, and thus in practice requires significant feedback control to maintain, the feedforward control surface inputs for this primitive are simply set to zero to avoid needless complications.



(a) Roll and pitch angles



(b) Control surface deflections



(c) Thrust

FIGURE 4.3: Trim primitive states and control inputs for each turn rate, $\dot{\psi}$.

4.2 Agile Maneuver Primitives

Agile maneuver primitives enhance the maneuver space - and hence the motion planner - with functional changes of the aircraft's pose, which could not otherwise be achieved, or at least not as effectively, using trim primitives. In contrast to trim primitives, agile maneuvers are executed over a pre-computed finite amount of time (determined by the free-terminal-time problems), and the states and control inputs are time-dependent. Three agile maneuvers were developed for use in the maneuver space: an aggressive turn-around (ATA), a cruise-to-hover transition (CTH), and a hover-to-cruise transition (HTC). The aggressive turn-around maneuver rapidly reverses the aircraft's heading. With this maneuver in its repertoire, the aircraft can turn away from dead-ends using only one primitive, and while occupying less space than would be required by piecing together even the most aggressive trim primitives. The cruise-to-hover transition, as its name suggests, transitions the aircraft from straight and level flight (cruise) to a hover. The hover-to-cruise transition performs the reverse maneuver. Together, these two primitives allow the aircraft to start and stop in a hover.

The agile maneuver primitives are also found by solving trajectory optimization problems. While this method does not preclude well-known maneuver sequences from appearing as a solution, such an outcome is not forced.

$$\min \; J \triangleq w_1 t_f + \int_0^{t_f} \left(w_2 \dot{\delta}_a^2 + w_3 \dot{\delta}_e^2 + w_4 \dot{\delta}_r^2 + w_5 \dot{\omega}_T^2 \right) dt$$

subject to the first-order dynamics of the aircraft (Eq. 2.1),

the path constraints:

$$\begin{aligned}
\delta_a &\in [-\delta_{a_{max}}, \delta_{a_{max}}], & \dot{\delta}_a &\in [-\dot{\delta}_{a_{max}}, \dot{\delta}_{a_{max}}] \\
\delta_e &\in [-\delta_{e_{max}}, \delta_{e_{max}}], & \dot{\delta}_e &\in [-\dot{\delta}_{e_{max}}, \dot{\delta}_{e_{max}}] \\
\delta_r &\in [-\delta_{r_{max}}, \delta_{r_{max}}], & \dot{\delta}_r &\in [-\dot{\delta}_{r_{max}}, \dot{\delta}_{r_{max}}] \\
\omega_T &\in [\omega_{T_{min}}, \omega_{T_{max}}], & \dot{\omega}_T &\in [-\dot{\omega}_{T_{max}}, \dot{\omega}_{T_{max}}]
\end{aligned} \tag{4.2}$$

and the boundary conditions (see Table 4.2):

$$\begin{aligned}
\mathbf{x}(0) &= \mathbf{x}_0, & \mathbf{u}(0) &= \mathbf{u}_0 \\
\mathbf{x}(t_f) &= \mathbf{x}_f, & \mathbf{u}(t_f) &= \mathbf{u}_f
\end{aligned}$$

Equation 4.2 includes a minimum-time cost function that additionally penalizes control inputs rates, to produce smooth control input time histories. It is well known that abrupt changes in control inputs can be detrimental to a mechanical system, and smooth inputs

TABLE 4.2: Boundary conditions for agile maneuver primitives. Straight and level trim conditions denoted by subscript $_{SL}$, hover trim conditions denoted by subscript $_{H}$.

States and Controls	**Boundary Conditions**					
	ATA		**CTH**		**HTC**	
	$t = 0$	$t = t_f$	$t = 0$	$t = t_f$	$t = 0$	$t = t_f$
$\boldsymbol{u}$	u_{SL}	u_{SL}	u_{SL}	0	0	u_{SL}
$\boldsymbol{v}$	0	0	0	0	0	0
$\boldsymbol{w}$	w_{SL}	w_{SL}	w_{SL}	0	0	w_{SL}
$\boldsymbol{p, q, r, \phi}$	0	0	0	0	0	0
$\boldsymbol{\theta}$	θ_{SL}	θ_{SL}	θ_{SL}	90°	90°	θ_{SL}
$\boldsymbol{\psi}$	0	π	0	0	0	0
$\boldsymbol{x}$	0	0	0	–	0	–
$\boldsymbol{y}$	0	0	0	0	0	0
$\boldsymbol{z}$	0	0	0	–	0	–
$\boldsymbol{\delta}_a$	0	0	0	0	0	0
$\boldsymbol{\delta}_e$	$\delta_{e_{SL}}$	$\delta_{e_{SL}}$	$\delta_{e_{SL}}$	0	0	$\delta_{e_{SL}}$
$\boldsymbol{\delta}_r$	0	0	0	0	0	0
$\boldsymbol{\omega}_T$	$\omega_{T_{SL}}$	$\omega_{T_{SL}}$	$\omega_{T_{SL}}$	ω_{T_H}	ω_{T_H}	$\omega_{T_{SL}}$

tend to result in more robust trajectories. The weights in the cost function are again proportional to units and aim to balance the penalties on each control input. They are set to $w_1 = 4$ s^{-1}, $w_2 = w_3 = w_4 = 0.01$ rad^{-2} s, and $w_5 = 2.5 \times 10^{-7}$ rpm^{-2}s. The minimization of time (t_f is a free variable) leads to aggressive maneuvers that require little space. The maneuvers are no longer steady, but the control inputs and their derivatives are constrained by the physical limits of the aircraft, namely, the limits and rate limits of the motor and control surface servomechanisms. Additionally, the maneuvers are finite-time transitions and thus must satisfy boundary conditions on the state and control vectors, $\mathbf{x} = [u,\ v,\ w,\ p,\ q,\ r,\ q_1,\ q_2,\ q_3,\ q_4,\ x,\ y,\ z]^\intercal$ and $\mathbf{u} = [\delta_a, \delta_e, \delta_r, \omega_T]^\intercal$, at $t = 0$ and $t = t_f$. The boundary conditions for each maneuver are listed in Table 4.2. The solutions to each agile maneuver problem come in the form of time-dependent reference trajectories and feedforward control inputs. Recall that Euler angles map onto quaternion attitude representations in the actual problem formulations, as per Eq. 2.35.

4.2.1 Aggressive Turn-Around

In full, the formulation for the ATA maneuver is a minimum time problem for reversing heading, in which the aircraft must begin and end in straight and level flight, with the same velocity and at the same position. The maneuver is similar to that of Section 3.3.2, however, a different cost function is used here to make the maneuver suitable for

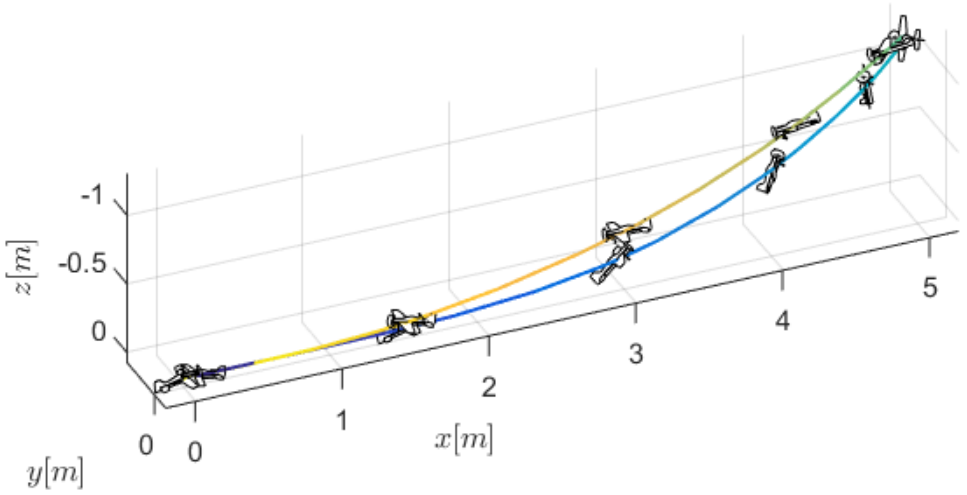
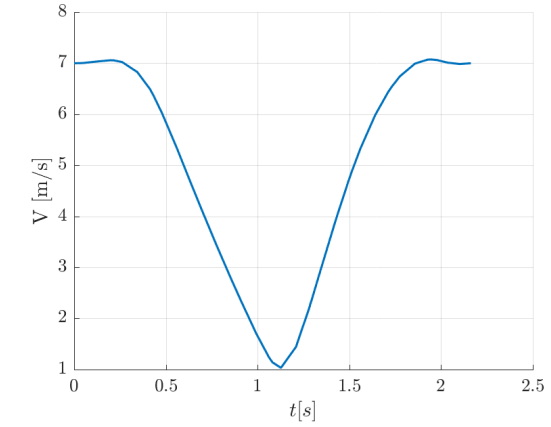


FIGURE 4.4: 3D visualization of ATA reference trajectory, $V_0 = 7\ \mathrm{m\,s^{-1}}$. The aircraft is scaled down by a factor of approximately four and the drawings of the aircraft are spaced out by a consistent time interval.

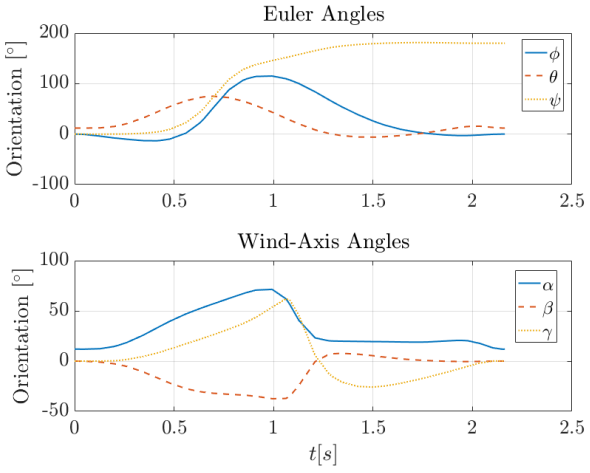
integration within the motion planner. Having the ATA in the planner's repertoire makes the fixed-wing UAV useful for a number of applications for which fixed-wings are currently ill-suited. One possible application of the maneuver is the situation in which the aircraft encounters a rapidly-approaching dead-end and must back-track. The ATA is not unlike the well-known Herbst maneuver for combat aircraft [17], which is designed as an evasive technique to minimize the time for reversing the heading. Unlike the ATA, which is found via optimization, the Herbst maneuver is a concatenation of well-known combat maneuver sequences.

The boundary conditions for the ATA, summarized in Table 4.2, represent trimmed straight and level flight, denoted by the subscript $_{SL}$, at the beginning and end of the maneuver. The trim conditions correspond to the initial speed of $V_0 = 7\ \mathrm{m\,s^{-1}}$. The heading reversal is enforced here, and the aircraft must also terminate at the same position and with the same velocity as when it began (another way in which the ATA differs from the Herbst maneuver). Relaxing the boundary conditions and constraints could result in a lower cost, however, their strict enforcement makes for an easier implementation of a highly functional maneuver.

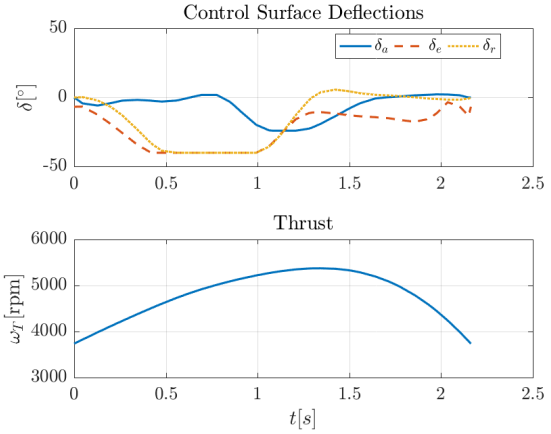
Figures 4.4 and 4.5 plot the reference trajectory of the aggressive turn-around maneuver, with $V_0 = 7\ \mathrm{m\,s^{-1}}$. We see in Fig. 4.5a that the velocity indeed begins and ends at $7\ \mathrm{m\,s^{-1}}$, dropping to approximately $1\ \mathrm{m\,s^{-1}}$ in between, transferring kinetic energy to potential energy as the aircraft changes direction. Figure 4.5b shows the smooth heading reversal, and that large pitch and roll angles are used in the process. The angle-of-attack, α, exceeds 70 degrees, well into the post-stall region, and large values of sideslip, β,



(a) Velocity



(b) Attitude



(c) Control inputs

FIGURE 4.5: ATA reference trajectory and feedforward control, $V_0 = 7\ \mathrm{m\,s^{-1}}$.

are used as well. We also note that the flight path angle, γ, begins and ends at zero, consistent with the boundary conditions for straight and level flight. As shown in Fig. 4.5c, all control inputs are smooth. We see that each control surface deflection saturates for some time during the maneuver. Recall, however, that the saturation values in the optimization are only 80% of the aircraft's actual limits. Finally, we note that all control inputs end at the values they began with, to trim the aircraft at straight and level flight with the desired velocity of 7 $\mathrm{m\,s^{-1}}$.

4.2.2 Cruise-to-Hover

The ability to transition into and out of a hover is arguably the most useful innovation of these novel fixed-wings, since it achieves a key advantage of rotorcraft over traditional fixed-wing aircraft: the ability to stop mid-flight. For easy integration into the motion planning framework, the two maneuvers were restricted to the vertical plane. The following path constraints were added to the optimization framework:

$$y = \phi = 0 \tag{4.3}$$

The boundary conditions for the two maneuvers are listed in Table 4.2. The two maneuvers use essentially the same two boundary conditions, only in opposite order. Straight and level flight is again consistent with $V_0 = 7\ \mathrm{m\,s^{-1}}$. The final forward and vertical position, x_{t_f} and z_{t_f}, are free variables.

The cruise-to-hover maneuver is visualized in Fig. 4.6, and the relevant states and control inputs are plotted in Fig. 4.7. We see in Fig. 4.6 that the maneuver requires approximately 5 m of forward distance and 2 m vertically to complete. Distance flown was indirectly penalized in the cost function via the time to complete, t_f. As depicted in Fig. 4.7a, the initial cruising speed of 7 $\mathrm{m\,s^{-1}}$ is reduced to zero in just over 2 seconds. During the maneuver, the pitch angle exceeds the terminal condition of 90°, such that the nose of the aircraft points backwards with respect to its velocity for approximately half a second. This phenomenon occurs because the direction of the thrust force, pointing backwards with the nose, acts to reduce the aircraft's forward velocity. Figure 4.7c shows that the elevator deflection is saturated for the first second of the maneuver. It is interesting to note that the thrust is mainly increasing during this time, even as the aircraft is slowing down. The reason for this is that the increased thrust enhances the control authority of the maxed-out elevator deflection. With the increased airflow over the aircraft's tail, the elevator can more effectively pitch the aircraft upwards.

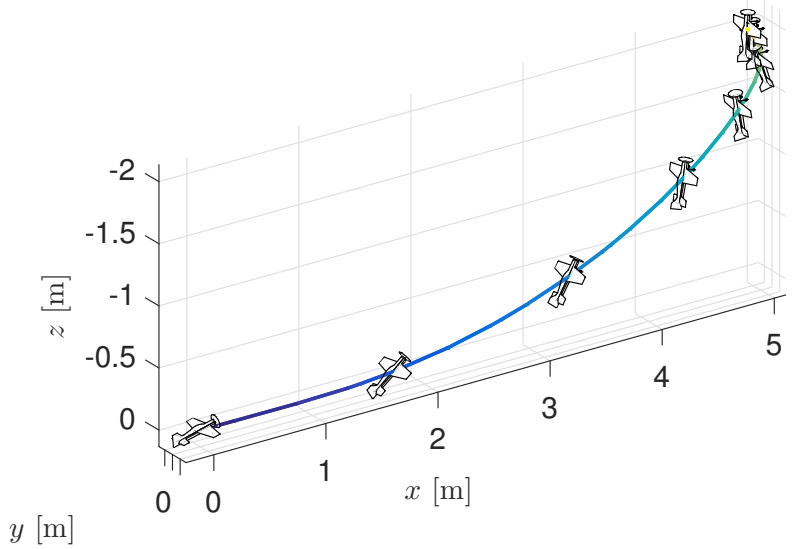


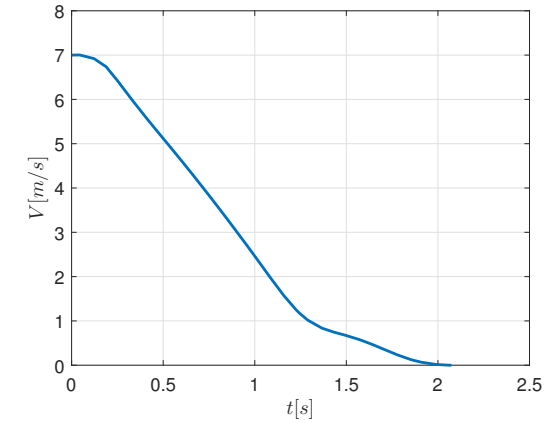
FIGURE 4.6: 3D visualization of CTH reference trajectory, $V_0 = 7\ \mathrm{m\,s^{-1}}$.

4.2.3 Hover-to-Cruise

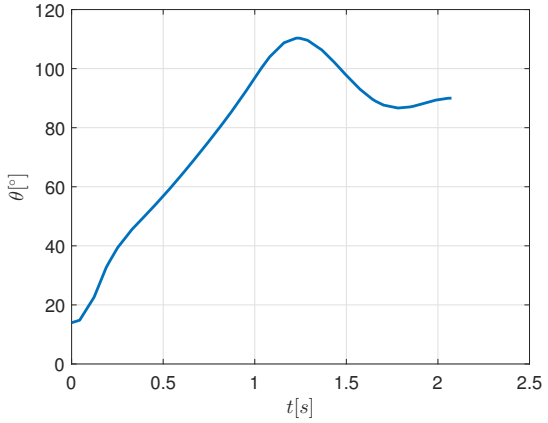
The hover-to-cruise trajectory is seen in Fig. 4.8. In this figure, we see that the maneuver takes approximately the same forward distance as the cruise-to-hover to accomplish, but uses very little vertical space. The states and controls are plotted in Fig. 4.9. As seen in Fig. 4.9a, the maneuver takes less than 1.5 s to reach the cruise condition at $V_f = 7\ \mathrm{m\,s^{-1}}$. The pitch angle, see Fig. 4.9b, rapidly decreases over the first 0.6 s and then stabilizes to the trim straight and level condition. In Fig. 4.9c, we see the elevator initially and briefly rise to its maximum value, and then continue to descend as the pitch angle decreases. The thrust input increases for the first second, then smoothly reduces to the rpm value corresponding to the straight and level flight condition.

4.3 Transitioning Between Primitives

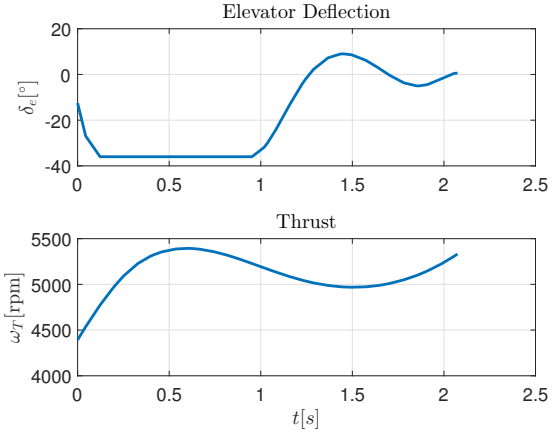
The aircraft is not infinitely agile and thus requires time to transition from one primitive to another. In theory, finite-time transition primitives could be used, but to connect the 116 trim primitives alone would require a massive and impractical expansion of the maneuver space. Instead, we implement a transition maneuver heuristic in the planner and control system.



(a) Velocity



(b) Pitch angle



(c) Control inputs

FIGURE 4.7: CTH reference trajectory and feedforward control, $V_0 = 7\ \mathrm{m\,s^{-1}}$.

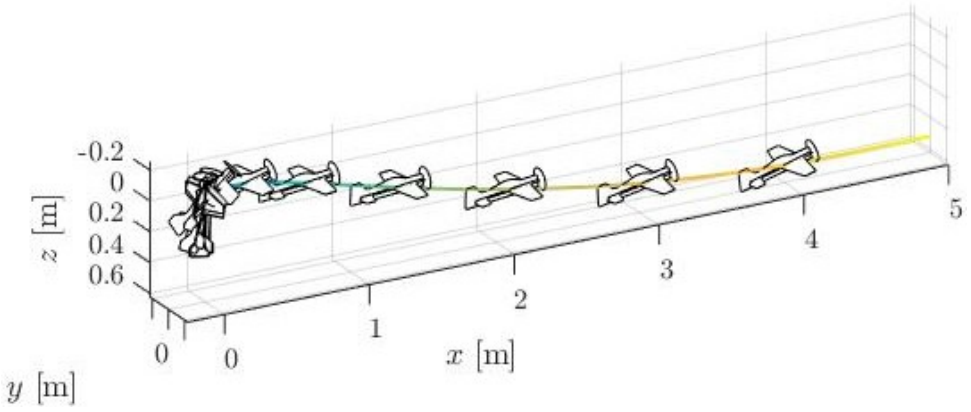
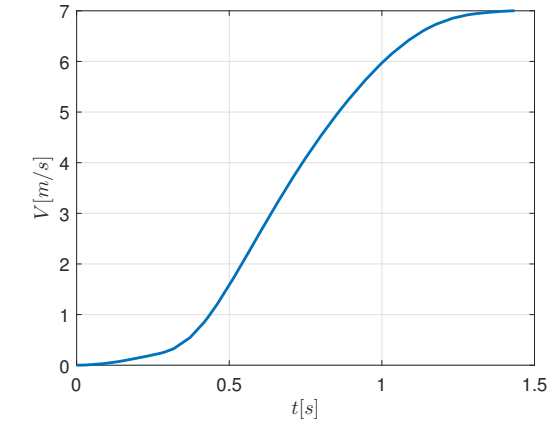


FIGURE 4.8: 3D visualization of HTC reference trajectory, $V_0 = 7\ \mathrm{m\,s^{-1}}$.

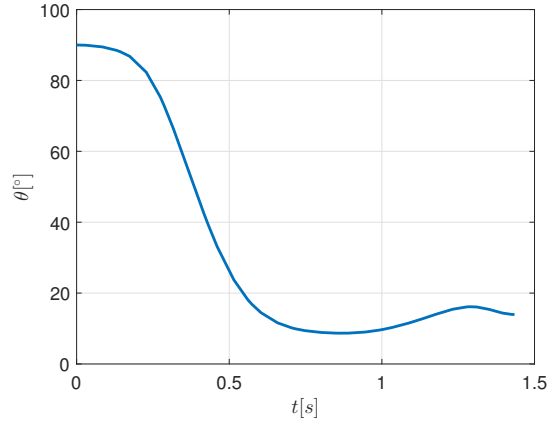
To smooth the transition between any two primitives (trim or agile), a time-delay heuristic is implemented. For the duration of the time-delay, the feedforward inputs and reference trajectory of the subsequent primitive are commanded, *except* for the path and heading, which are extended from the previous primitive. Consider Fig. 4.10, in which A to B is one trim primitive, and B' to C is another. The trajectory from B to B' is the transition maneuver. The maneuver extends the curvature of the path and heading of the A to B primitive, while all steady states tracked and control inputs commanded are that of the B' to C primitive.

The rationale behind the transition maneuver is based on the time-scale separation principle, as it applies to the physics of fixed-wing flight [109]; in particular, the modal time-scales of the fast rotational and slow translational dynamics. Accordingly, we diminish position tracking errors by allowing the aircraft a short amount of time to continue along the path predicted by its current motion as it begins to transition into the steady states of the next primitive. What this enables is a transition that avoids sudden changes in variables that cannot react fast enough, while preparing the fast variables for the next state. The time-scale separation principle is mirrored in the design of the feedback controller, which has independent inner (attitude) and outer (position) control loops.

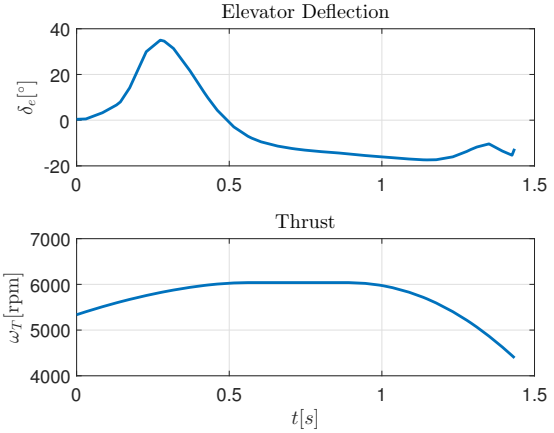
An analysis of the aircraft's dynamics is used to determine the duration of the delay. Essentially, we want to determine the time it typically takes for the roll angle to reach a commanded value. It is the roll dynamics which should dominate the calculation of the duration because the roll angles are undergoing large discontinuous changes. Pitch angles are also discontinuously commanded, but the changes are relatively small. Under the control system (the combination of feedforward inputs and feedback control laws), the roll dynamics behave similarly to a low-pass filter of the form $\frac{1}{\tau s+1}$. This low-pass filter may be viewed as the first-order Padé approximation of a time-delayed input, t_d [110, p.183]. For step input commands (as is the case for the change in commanded roll



(a) Velocity



(b) Pitch angle



(c) Control inputs

FIGURE 4.9: HTC reference trajectory and feedforward control, $V_0 = 7\ \mathrm{m\,s^{-1}}$.

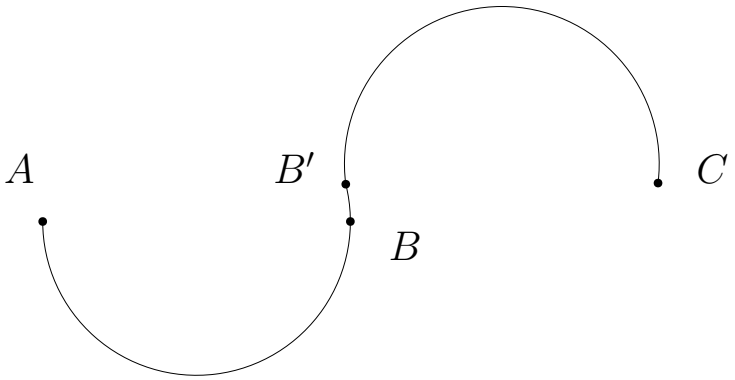


FIGURE 4.10: Structure of a transition maneuver.

angle between primitives), it can be shown that τ is a suitable value of the time-delayed input, t_d [47].

To determine the value of τ, simulations of the aircraft model and control system were run in Simulink. Step input commands for roll were given, and the outputs of these commands superimposed with low-pass filters was observed. The value of τ in the low-pass filter was tuned until the output of the filter closely matched the aircraft's actual dynamics under the control system. The value of τ found this way, 0.23 s (at $V = 7\text{m}\,\text{s}^{-1}$), was given to the transition trajectory time-delay constant, t_d. Figure 4.11 shows the simulated roll dynamics as the controller tracks a pre-defined motion plan that involves three trim primitives. During the first 10 seconds, the reference trajectory is the straight and level primitive. During the next 20 seconds, a banked turn at a rate of $\dot{\psi} = -60^\circ\,\text{s}^{-1}$ is commanded; and finally a banked turn with $\dot{\psi} = 60^\circ\,\text{s}^{-1}$ is commanded. The blue line is the commanded roll angle throughout the plan, and the black line shows the actual roll, under the control system. The final line, in magenta, plots the reference roll angle having gone through the low-pass filter. As can be seen, with $\tau = 0.23$ s, the actual roll dynamics are approximated closely by the low-pass filter.

The transition maneuver heuristic was also validated in trajectory tracking simulations. A series of primitives were sequenced in the following order: straight and level flight, a banked turn to the left, and a banked turn to the right. Using the feedback controller and simulation environment, the primitive sequence was tracked; first with and then without including the transition maneuver. Two sets of trials were conducted, one using turns of $\dot{\psi} = 30^\circ\,\text{s}^{-1}$, and another using turns of $\dot{\psi} = 60^\circ\,\text{s}^{-1}$. In both cases, the transition maneuver reduced the overall position tracking errors by approximately 35%. With the $30^\circ\,\text{s}^{-1}$ turns, the root-mean-square error (RMSE) on position was reduced from 0.33 m to 0.21 m, and with $60^\circ\,\text{s}^{-1}$ turns, the error was reduced from 0.44 m to 0.28 m. The position tracking errors for the first case, $\dot{\psi} = \pm 30^\circ\,\text{s}^{-1}$, are plotted in Fig. 4.12. The plot

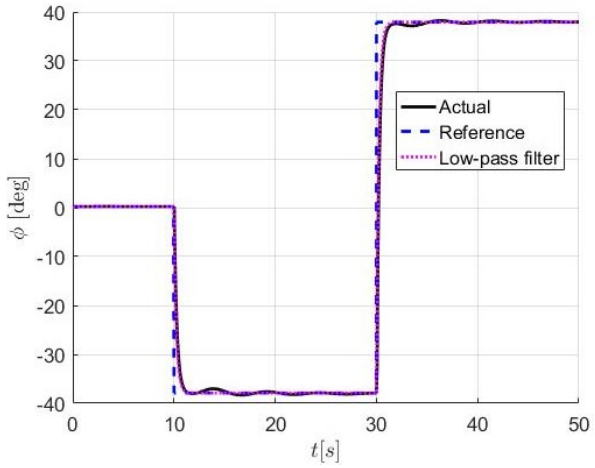


FIGURE 4.11: Comparison of actual roll dynamics to low-pass filter.

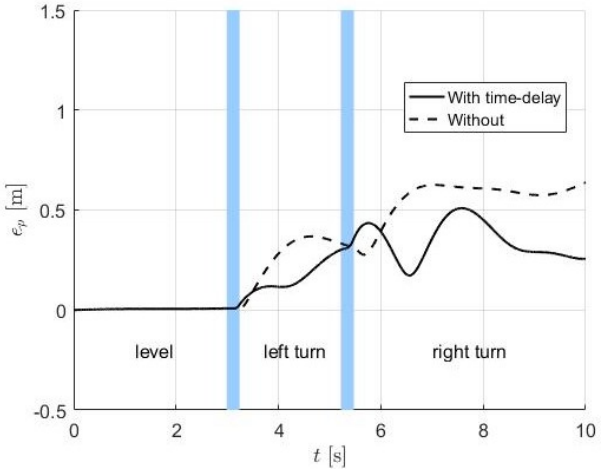


FIGURE 4.12: Position tracking errors with and without using the time-delay transition maneuvers, for turns using $\dot{\psi} = \pm 30^\circ\,\mathrm{s}^{-1}$. The background is colored light blue during the time periods when the transition maneuver is being executed.

illustrates how the position tracking performance is improved by using the time-delayed transitions. Notice as well that the position error, e_p, remains stable throughout the transition maneuvers (the light blue sections).

4.4 Knife-Edge Maneuver

A final maneuver, called the knife-edge, was generated using a similar approach to the others. The knife-edge maneuver does not fit the description of either a trim primitive *or* an agile maneuver primitive, because it is a combination of both. The maneuver is composed of three phases. First is a transition from level flight to knife-edge flight, a 90° roll. Knife-edge flight is then held for a desired duration before transitioning back to

TABLE 4.3: Steady knife-edge states and control inputs

Symbol	**Value**	**Unit**
θ_{KE}	42.6	deg
$\delta_{a_{KE}}$	0.626	deg
$\delta_{e_{KE}}$	-1.49	deg
$\delta_{r_{KE}}$	37.0	deg
$\omega_{T_{KE}}$	4434	rpm

level flight. The maneuver differs from the knife-edges designed in [52] and [51] in terms of the highly transient nature of the transitions and that the knife-edge roll can be held indefinitely as a trim condition. The differences in the design of the maneuver from our other agile maneuvers make it ill-suited for the maneuver space, however, the knife-edge is incorporated into a planning problem that is presented in Chapter 5.

An optimal control problem is solved for each stage of the maneuver: one to determine a steady 90° roll trim condition, and two others connecting this trim condition to level flight. First formulated is the problem of holding knife-edge, the 90° roll, steady. The cost function was defined in such a way as to minimize the maneuver's deviation from a straight-line trajectory. Changes in velocity and deviations from the straight path were penalized:

$$\min \; J \triangleq \int_0^{t_f} \left(w_1 (V - V_0)^2 + w_2 y^2 + w_3 z^2 \right) dt, \tag{4.4}$$

where $w_1 = 1 \text{ m}^{-2}\,\text{s}$ and $w_2 = w_3 = 1 \text{ m}^{-2}\,\text{s}^{-1}$. The velocity selected here and for the motion planner in Section 5.2 is $V_0 = 5 \text{ m}\,\text{s}^{-1}$, which is also within the normal cruising speed range of the aircraft. The slower speed, relative to the other agile maneuvers, was chosen to be slightly more appropriate for the size of the environments used for motion planning in Section 5.2. Path constraints on constant control input action are enforced - just as they were for the trim primitives - and an additional path constraint is placed on the roll angle, $\phi = 90°$. Finally, a boundary condition is placed on the final time, $t_f = 10$ s, to ensure the knife-edge can in fact be held steadily.

The results of this problem are summarized in Table 4.3 and Fig. 4.13, which confirms that velocity, altitude, and lateral position are steady. It follows that the solution to this optimization problem can similarly be generated using the trim condition cost function of Eq. 4.1.

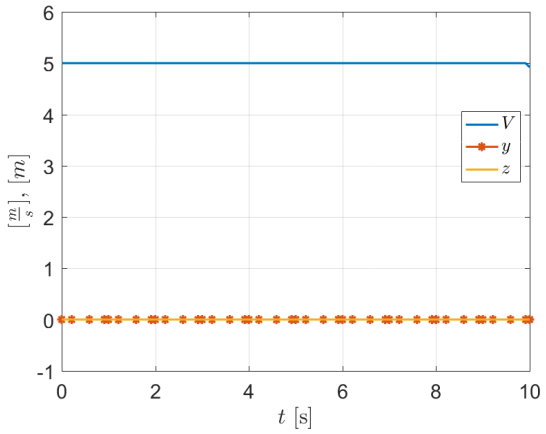


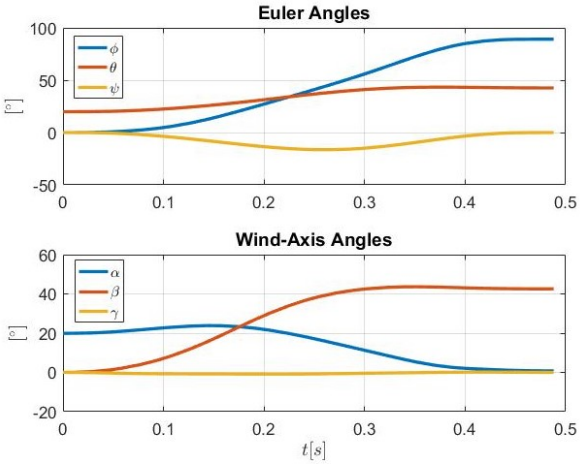
FIGURE 4.13: Steady knife-edge flight reference trajectory, $V_0 = 5\ \mathrm{m\,s^{-1}}$.

The cost function for the transition maneuvers to and from knife-edge flight is similar to the previous agile maneuver primitives, except that it also penalizes deviations in velocity and deviations from the straight-line path:

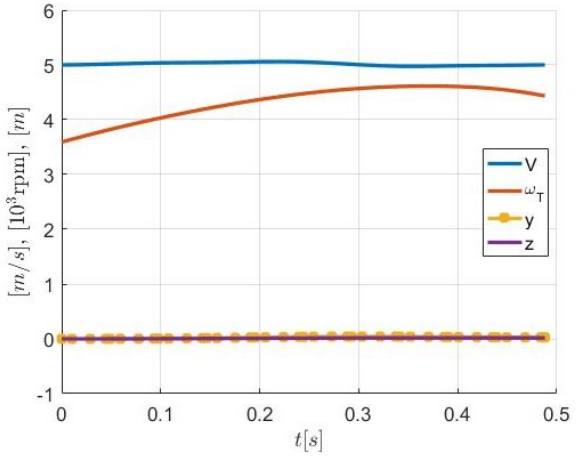
$$\begin{aligned} \min\ J \triangleq w_1 t_f + \int_0^{t_f} \Big(& w_2(V - V_0)^2 + w_3 y^2 + w_4 z^2 + \\ & w_5 \dot{\delta}_a^2 + w_6 \dot{\delta}_e^2 + w_7 \dot{\delta}_r^2 + w_8 \dot{\omega}_T^2 \Big) dt \end{aligned} \tag{4.5}$$

The penalties on velocity and path deviations are included in this optimization as a deliberate design choice that makes the maneuver easy to integrate into the planner described in Section 5.2.1.1. The weights in the cost function, which combine those from Eqs. 4.2 and 4.4, are listed in Table 4.4. Like the other agile maneuvers, the knife-edge transitions must also enforce boundary conditions. The two sets of boundary conditions, for wings-level and knife-edge flight, are summarized in Table 4.5. The level flight boundary conditions are the trim conditions corresponding to the desired speed, $V = 5\ \mathrm{m\,s^{-1}}$. The knife-edge boundary conditions are the solution to the first optimal control problem (Table 4.3). Note that the body-frame velocity components, u_{KE} and v_{KE}, follow directly from θ_{KE} and V_0. Assigning the sets of boundary conditions, one as the initial condition and the other as the final condition, is the only difference in problem formulation between the transition maneuvers to and from knife-edge flight.

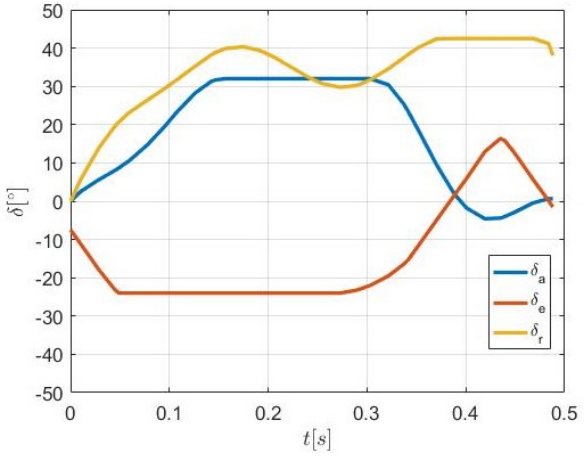
Figure 4.14 displays the solution for transitioning into knife-edge flight from the wings-level condition. At this speed, it takes less than 0.5 s and 2.5 m to complete the transition. As seen in Fig. 4.14a, the roll is rapid and smooth, the initial angle-of-attack, α, is traded



(a) Attitude



(b) Velocity, thrust, lateral displacement and altitude



(c) Control surface deflections

FIGURE 4.14: Reference trajectory for transition from level to knife-edge flight, $V_0 = 5\ \mathrm{m\,s^{-1}}$.

TABLE 4.4: Weights in the cost function for transition maneuvers to and from knife-edge flight.

Term	**Value**	**Unit**
w_1	4	s^{-1}
w_2	1	$\mathrm{m}^{-2}\,\mathrm{s}$
w_3	1	$\mathrm{m}^{-2}\,\mathrm{s}^{-1}$
w_4	1	$\mathrm{m}^{-2}\,\mathrm{s}^{-1}$
w_5	0.01	$\mathrm{rad}^{-2}\,\mathrm{s}$
w_6	0.01	$\mathrm{rad}^{-2}\,\mathrm{s}$
w_7	0.01	$\mathrm{rad}^{-2}\,\mathrm{s}$
w_8	2.5×10^{-7}	$\mathrm{rpm}^{-2}\mathrm{s}$

TABLE 4.5: Boundary conditions for knife-edge transition maneuvers. Straight and level trim conditions denoted by subscript $_{SL}$, knife-edge flight conditions denoted by subscript $_{KE}$

States and Controls	**Boundary Conditions**	
	Wings Level	Knife-Edge
$\boldsymbol{u}$	u_{SL}	u_{KE}
$\boldsymbol{v}$	0	v_{KE}
$\boldsymbol{w}$	w_{SL}	0
$\boldsymbol{p, q, r}$	0	0
$\boldsymbol{\phi}$	0	90°
$\boldsymbol{\theta}$	θ_{SL}	θ_{KE}
$\boldsymbol{\psi}$	0	0
$\boldsymbol{\delta}_a$	0	$\delta_{a_{KE}}$
$\boldsymbol{\delta}_e$	$\delta_{e_{SL}}$	$\delta_{e_{KE}}$
$\boldsymbol{\delta}_r$	0	$\delta_{r_{KE}}$
$\boldsymbol{\omega}_T$	$\omega_{T_{SL}}$	$\omega_{T_{KE}}$

for positive sideslip, β, and the flight path angle, γ, stays near zero. Figure 4.14b confirms that the velocity remains nearly constant throughout the maneuver, as does the lateral displacement and altitude. The feedforward control surface deflections are seen in Fig. 4.14c. Each control surface hits its saturation limit constraint at some point in time (80% of the actual saturation limits). Plots of the transition from knife-edge to wings-level flight are omitted, since the problem definition only differs by the reversal of the sets of boundary conditions, and thus the solution's characteristics are largely analogous.

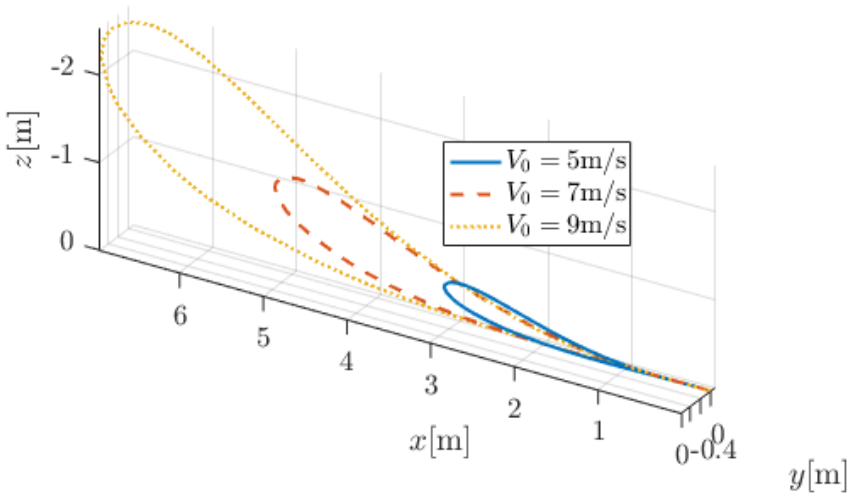
4.5 Velocity Parametrization

Up to this point, the maneuver space has been developed for flight at a constant speed (other than the temporary changes that occur during agile maneuvers). While this may be sufficient for many applications of autonomous flight, there is certainly value to be found in extending this maneuver space to allow for speed changes during flight. One of the impediments to doing so is the need for additional computational resources. In this section, we present a methodology that was developed to, at least partially, address this concern. The methodology is used to continuously parametrize agile maneuvers - or any finite-time transition - by initial speed. In effect, only two maneuvers need to be generated and stored on-board the aircraft for a maneuver class, e.g. aggressive turn-arounds, to be continuously parametrized by speed.

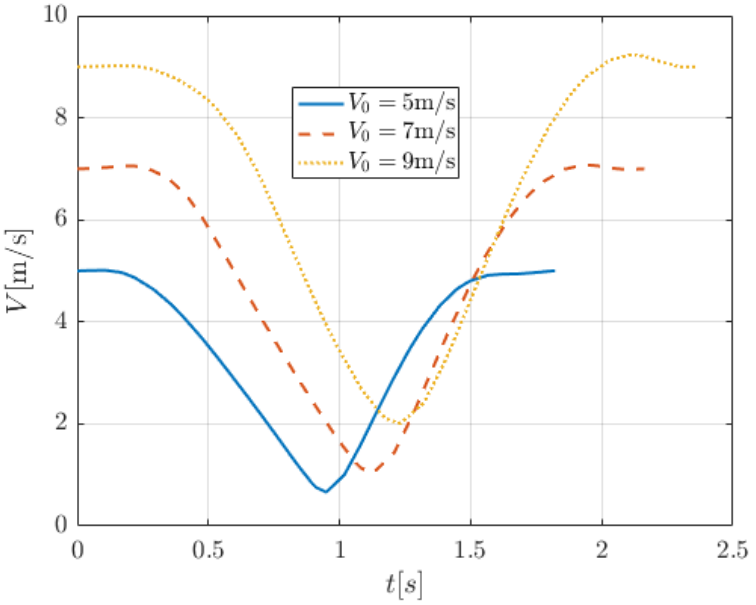
Throughout this section, the aggressive turn-around maneuver will be used as an example to explain the methodology. In Section 4.2.1, we solved for the ATA with an initial speed of $V_0 = 7\mathrm{m\,s^{-1}}$. Here, we use the resulting solution as an initial guess in GPOPS-II to solve the problem for $V_0 = 5\mathrm{m\,s^{-1}}$ and then for $V_0 = 9\mathrm{m\,s^{-1}}$. The results from the three cases show very similar patterns in the trajectories and control input time histories. Figure 4.15 presents the path and velocity time histories. The two results associated with $V_0 = 5\mathrm{m\,s^{-1}}$ and $V_0 = 9\mathrm{m\,s^{-1}}$ are kept for use in the trajectory parametrization process that allows maneuvers to be generated in real-time over the range of intermediate velocities.

4.5.1 Dynamic Time Warping for Interpolation of Agile Maneuver Primitives

Dynamic Time Warping (DTW) is an algorithm used to find an optimal alignment between two time-varying sequences which need not necessarily be of the same duration. The two sequences' time axes are *warped* so that the sequences match each other based on similarities, as shown conceptually in Figure 4.16. DTW has been applied to speech recognition [111], image matching [112], and recognition of human walking patterns [113], but its use is general and extends well to measuring similarities between any two temporal sequences, such as our agile maneuver primitives. In Section 4.2, the terminal time of the agile maneuvers is left as a free variable, and therefore the maneuver duration will be different for each optimal control solution pertaining to a particular initial velocity. For more information on DTW, the reader is referred to [48]. In [114], a similar use of DTW is employed for apprenticeship learning of aerobatic helicopter trajectories.



(a) Paths



(b) Velocities

FIGURE 4.15: Trajectories for ATA's generated from different initial velocities.

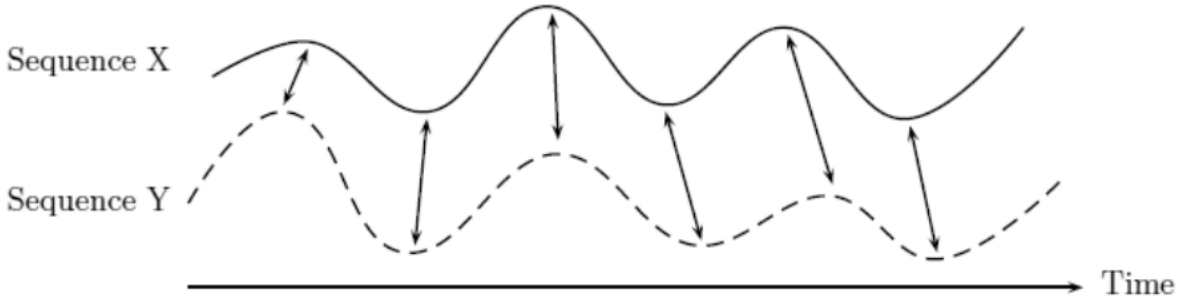


FIGURE 4.16: DTW alignment of two time-varying sequences [8].

Here, we use DTW to prepare pairs of vectors with different durations for linear interpolation. The DTW-based interpolation methodology we employ can be strategically separated into an off-line and an on-line portion.

4.5.2 Off-Line

As described by the process in Section 4.2, we generate two sets of trajectories and control inputs for the maneuver, each a solution to an optimal control problem at a particular initial velocity. Let vectors $\mathbf{x}_i \in \mathbb{R}^S, i = 1, ..., N$ and $\mathbf{y}_j \in \mathbb{R}^S, j = 1, ..., M$ be the state and control vectors at the i^{th} and j^{th} point in time, for initial velocities V_{0_1} and V_{0_2} respectively, where $S = 17$ is the number of states (13) plus the number of control inputs (4), and N, M are the number of collocation points for each trajectory. We define two time series matrices that store all vectors of each solution: $\boldsymbol{X} := [\mathbf{x}_1, ..., \mathbf{x}_N] \in \mathbb{R}^{S \times N}$ and $\boldsymbol{Y} := [\mathbf{y}_1, ..., \mathbf{y}_M] \in \mathbb{R}^{S \times M}$, where $\boldsymbol{X}$ is the matrix for V_{0_1} and $\boldsymbol{Y}$ is the matrix for V_{0_2}.

The rows of $\boldsymbol{X}$ and $\boldsymbol{Y}$ describe states that do not all have the same units, e.g. the states u and p have different units. In preparation for the DTW alignment, we compute their respective normalized matrices, $\boldsymbol{X}_n$ and $\boldsymbol{Y}_n$, by the following procedure. Consider the first entry (in row and column) of $\boldsymbol{X}$, denoted as u_{X_1} since it is represents a value of the body-frame x-axis speed, u. The normalized value of u_{X_1} is computed as follows:

$$u_{X_{1_n}} = \frac{u_{X_1} - u_{\min}}{u_{\max} - u_{\min}}, \tag{4.6}$$

where $u_{\min}$ and $u_{\max}$ are the minimum and maximum of all u values in both $\boldsymbol{X}$ and $\boldsymbol{Y}$. All values in $\boldsymbol{X}_n$ and $\boldsymbol{Y}_n$ thus range between 0 and 1.

The standard DTW algorithm [48] is performed off-line, to calculate an optimal match between the two normalized matrices. The algorithm returns the *warping paths* $\boldsymbol{i}_1$ and $\boldsymbol{i}_2$. The two warping paths are vectors of column indices, and they have the same length, because one or both of $\boldsymbol{i}_1$ and $\boldsymbol{i}_2$ will contain repeated indices. The warping paths are calculated so that two new matrices, $\boldsymbol{X}'$ and $\boldsymbol{Y}'$, have the smallest total *cost* (to be defined) between them. These matrices, $\boldsymbol{X}' \in \mathbb{R}^{S \times L}$ and $\boldsymbol{Y}' \in \mathbb{R}^{S \times L}$, are stretched versions of $\boldsymbol{X}$ and $\boldsymbol{Y}$ such that some of the columns of $\boldsymbol{X}$ and $\boldsymbol{Y}$ may be repeated as necessary in $\boldsymbol{X}'$ and $\boldsymbol{Y}'$ – as per $\boldsymbol{i}_1$ and $\boldsymbol{i}_2$ – to minimize the total cost. Hence, $L \geq \max(N, M)$. Note that the normalized matrices are used to generate the warping paths, but it is the original matrices that are stretched to construct $\boldsymbol{X}'$ and $\boldsymbol{Y}'$.

DTW allows the user to define precisely how the cost computed at each time instant, c_k, is calculated. Here we use the Euclidean distance:

$$c_k := \sqrt{(x'_{1k} - y'_{1k})^2 + (x'_{2k} - y'_{2k})^2 + ... + (x'_{Sk} - y'_{Sk})^2}, \tag{4.7}$$

where x'_{ik} and y'_{ik} are the i, k elements of $\boldsymbol{X}'$ and $\boldsymbol{Y}'$, respectively. Since $\boldsymbol{X}'$ and $\boldsymbol{Y}'$ will necessarily be the same size - and optimally aligned - they can be stored on-board the aircraft's autopilot in memory and used for the following process.

4.5.3 On-Line

All that is left to do in real-time is linear interpolation. First, a weight, σ, is generated based on the aircraft's actual velocity at the start of the maneuver, $V_{0_{\text{int}}}$, and the initial velocities of the two pre-computed trajectories, where $V_{0_1} = 5\text{m}\,\text{s}^{-1}$ and $V_{0_2} = 9\text{m}\,\text{s}^{-1}$:

$$\sigma = \frac{V_{0_{\text{int}}} - V_{0_2}}{V_{0_1} - V_{0_2}} \tag{4.8}$$

Using this weight, we interpolate between the pairs of columns of $\boldsymbol{X}'$ and $\boldsymbol{Y}'$ to get state and control input time histories associated with $V_{0_{\text{int}}}$, denoted in general here as $\mathbf{s}_{\text{int}}$:

$$\mathbf{s}_{\text{int}} = \sigma\mathbf{s}'_1 + (1 - \sigma)\mathbf{s}'_2, \tag{4.9}$$

where the vectors $\mathbf{s}'_1$ and $\mathbf{s}'_2$ represent a warped state or control input time history (i.e. rows of $\boldsymbol{X}'$ and $\boldsymbol{Y}'$) associated with $V_{0_1} = 5\text{m}\,\text{s}^{-1}$ and $V_{0_2} = 9\text{m}\,\text{s}^{-1}$.

An amendment is made to the above when interpolating quaternions, since Eq. (4.9) would not produce unit quaternions in general. Instead, we interpolate quaternions using spherical linear quaternion interpolation, or *Slerp* [115]:

$$\mathbf{q}_{\text{int}} = \mathbf{q}'_1 \odot (\mathbf{q}'^*_1 \odot \mathbf{q}'_2)^\sigma, \tag{4.10}$$

where $\mathbf{q}'_1$ and $\mathbf{q}'_2$ are the warped quaternions, and in general, $\mathbf{q}^*$ is the conjugate of a quaternion. The entire interpolation step is performed immediately before the maneuver begins to generate a reference trajectory and feedforward control policy.

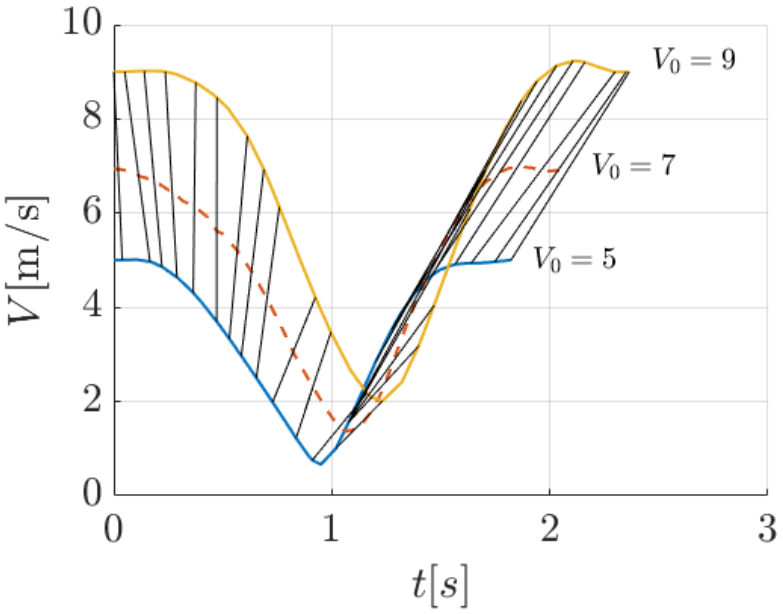
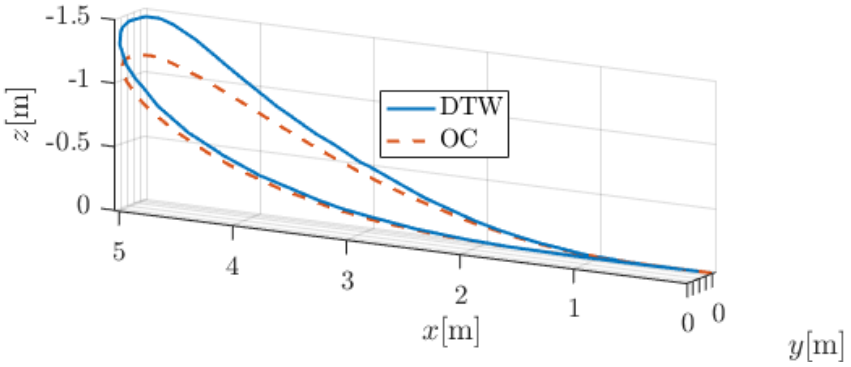


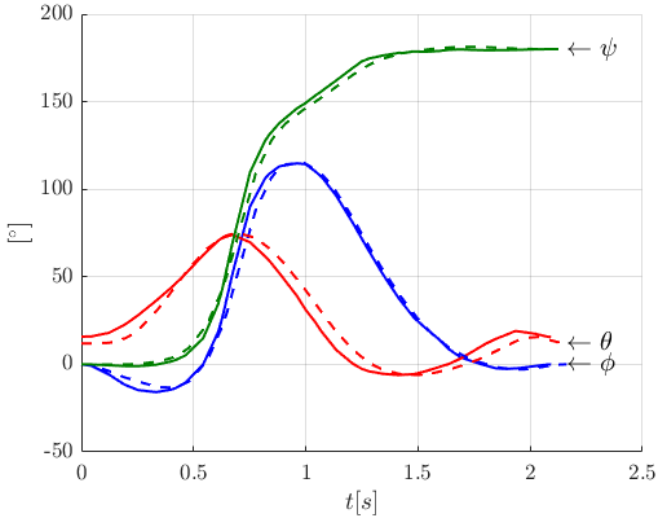
FIGURE 4.17: Alignment of original velocity vectors generated by optimal control (solid curves) and velocity vector found from interpolation (dashed curve); solid near-vertical lines are the alignment lines.

Figure 4.17 is a visual representation of the alignment between the velocity profiles of the two original bounding ATA trajectories ($V_0 = 5\mathrm{m\,s^{-1}}$ and $V_0 = 9\mathrm{m\,s^{-1}}$) generated by optimal control, as well as a velocity profile found through interpolation for $V_0 = 7\mathrm{m\,s^{-1}}$. Notice how some of the alignment lines joining the collocation points on the two bounding velocity profiles start or end at the same point in time, indicating repeated indices in $\boldsymbol{i}_1$ and $\boldsymbol{i}_2$. For clarity, only velocity is displayed, however, the alignment extends consistently to all states and control inputs such that all values in a column of $\boldsymbol{X}'$ or $\boldsymbol{Y}'$ correspond to the same value of time.

To verify that the algorithm produces feasible reference trajectories and suitable control policies, we compare the ATA optimal control solution for the case where $V_0 = 7\mathrm{m\,s^{-1}}$ to the output of the interpolation for $V_{0_{\mathrm{int}}} = 7\mathrm{m\,s^{-1}}$, again performed using reference trajectories with $V_0 = 5\mathrm{m\,s^{-1}}$ and $V_0 = 9\mathrm{m\,s^{-1}}$. As represented by the path and attitude time histories in Fig. 4.18, the two results nearly align, implying that the proposed scheme will yield a result very similar to the optimal control solution. Note that the velocity parametrization method does not explicitly ensure dynamic feasibility of interpolated trajectories. Engineering oversight is needed to ensure that the bounding trajectories are near and similar to each other, such that it can reasonably be inferred that no physical constraints will be violated by an interpolation.



(a) Path



(b) Body-frame attitude

FIGURE 4.18: DTW interpolated trajectory (solid lines) versus optimal control trajectory (dashed lines).

4.6 Storage of Maneuver Space

The maneuver space is compactly stored in memory on-board the aircraft's autopilot, discussed in Chapter 6. There are 116 trim primitives, but each one of them only requires a few values to be stored: the steady states and constant control inputs. While the path and heading do evolve during the execution of a trim primitive, they can be calculated from simple algebraic equations based on the starting time, position, and heading of the primitive. The agile maneuver primitives are time-dependent and thus are each stored as matrices. As an agile maneuver is being flown, the control system is interpolating the matrix by time. The optimal control solutions for each agile primitive contain many points in time along the maneuver, so the interpolations are highly accurate. No space in memory is required to implement the transitioning maneuver heuristic. To implement the velocity parametrization method, each agile maneuver primitive must be replaced by

its respective $\boldsymbol{X}'$ and $\boldsymbol{Y}'$ matrices, as described in Section 4.5.2. Chapter 5 goes into more detail about the implementation of the maneuver space.

Chapter 5

Motion Planning

The objective of the work presented in this chapter is to integrate the motion primitives developed in Chapter 4 into a motion planning framework. While they share many similarities, two separate motion planning strategies are presented here. At the time of investigating agile maneuvers, the knife-edge appeared to serve a valuable purpose for autonomous applications, as it would allow the aircraft to pass through gaps more narrow than its wingspan. As described in Section 4.4, though, the knife-edge maneuver is different than the other agile maneuvers because it has multiple phases, and thus could not be incorporated into the maneuver space or the planning framework that uses it. Rather than disregard it entirely, we integrated the knife-edge into a slightly different planner that could accommodate the maneuver.

The two planning frameworks share fundamental similarities. For one, they are both based on an RRT-variant algorithm, originally described in [45]. Additionally, both frameworks utilize the full six degrees-of-freedom of the aircraft's motion. In other respects, the two planners are different. One main difference is that the planner that uses the knife-edge – which we will call *Planner 1* – employs a two step process: first generating a two-dimensional straight-line path, and then transforming it into a time-dependent trajectory. The other planner, *Planner 2*, builds the tree using dynamically feasibly trajectories; it was Planner 2 that was pursued for flight tests. Another significant difference between the two is that Planner 2 runs in real-time, while Planner 1 only works off-line. There are other, more minor differences that will become apparent as the two planners are described in detail.

With respect to the real-time capabilities of *Planner 2*, there is a line of communication between the on-board motion planning computer and the on-board autopilot system

(which includes the sensor suite and controller). When planning is commenced, the aircraft's pose is measured and sent to the planner for initialization. As the motion planning algorithm runs, it iteratively sends back to the autopilot trajectories to be executed. For the sake of the 're-planning' portion of the algorithm, which will be discussed in Section 5.3.5.1, the autopilot sends estimates of the aircraft's true position back to the motion planning computer for integration within the algorithm.

This chapter continues with a brief description of the RRT algorithm. Planner 1 is first discussed in Section 5.2, since it is more basic in nature. Section 5.3 will then describe the more general motion planning framework, Planner 2, which builds off of the development of the maneuver space from Chapter 4. This section includes a comparison of Planner 2 to a baseline approach that uses Dubins curves.

5.1 Rapidly-Exploring Random Trees

The motion planner is based on the RRT algorithm, which is a single-query planning method that efficiently explores an environment such that a feasible path to the goal region can be constructed rapidly. The RRT algorithm works by extending a tree - starting from a predefined initial node - by sampling random nodes in the environment and connecting them to their nearest node in the tree. The connection is discarded if it happens to incur a collision with an obstacle in the environment. This method biases the search into the largest Voronoi regions, i.e. the unexplored areas [60]; in our case, in the three-dimensional Cartesian space, $\mathcal{C} = \mathbb{R}^3$. This concept is known as the *Voronoi bias*, which is the key aspect of the RRT algorithm and what makes it efficient at exploration. The algorithm used here varies from the standard algorithm in the way it attempts to connect random nodes to the nearest node in the tree. In the standard algorithm, a connection is attempted only with the one nearest node in the tree. If a collision results, the random node is rejected and a new one is generated. In the algorithm implemented here, the random node goes partially through the sorted list of nearest tree nodes, checking for one it can connect to collision-free. This helps the tree build through corridors and narrow passages [45].

5.2 Planner 1: Off-Line Planning with the Knife-Edge Maneuver

In this section we demonstrate how the knife-edge maneuver was incorporated into a motion planning framework. In brief, this is accomplished with a top-level planner generating a trajectory for conventional path following, which afterwards is overlaid with the knife-edge maneuver where appropriate. In the motion plan, we prioritize smoothness and feasibility over optimality. We assume that the aircraft is equipped with a map of the environment surrounding the initial position and goal region, and can recognize the locations of narrow passages. We command the aircraft to maintain a constant velocity and altitude as it navigates the environment. As a departure from most of the rest of the thesis, where we use a cruising speed of $V = 7\text{m}\,\text{s}^{-1}$, a slower speed of $V = 5\text{m}\,\text{s}^{-1}$ is used in this section. While there is nothing particularly unique about this speed, we found the slower speed to be slightly more appropriate for planning with the knife-edge maneuver. Unlike the other agile primitives, the knife-edge begins and ends with transition maneuvers, which for the sake of agile motion planning, can be made shorter in distance using the slower speed.

5.2.1 Top-Level Planning

The objective we set for the motion planner is to generate a smooth and feasible trajectory to traverse a planar environment with obstacles. This is accomplished in three stages. First, the RRT-variant algorithm constructs a collision-free path composed of straight segments. The planned path has a constant altitude - the nodes of the RRT tree store 2D Cartesian coordinates. Next, the path is automatically smoothed by *shortcutting*, i.e. getting rid of redundant nodes. A node is unnecessary if it can be discarded without the resulting path encountering a collision or violating the accessible space condition that will soon be defined. Lastly, the path is algebraically converted into a time-dependent trajectory and augmented with the states used in the feedback control laws. Although the path we consider is two-dimensional, the actual motion of the aircraft utilizes its full six degrees of freedom; the dynamics model - used for optimal control and simulations - is never constrained to 2-D motion.

To ensure the geometric path can feasibly be tracked, we impose a constraint on turning (heading) rate by only constructing nodes within an 'accessible space' relative to the previous (parent) node. By enforcing this constraint in the path planning process, the solution is guaranteed to meet our goals of smoothness and feasibility. This eliminates

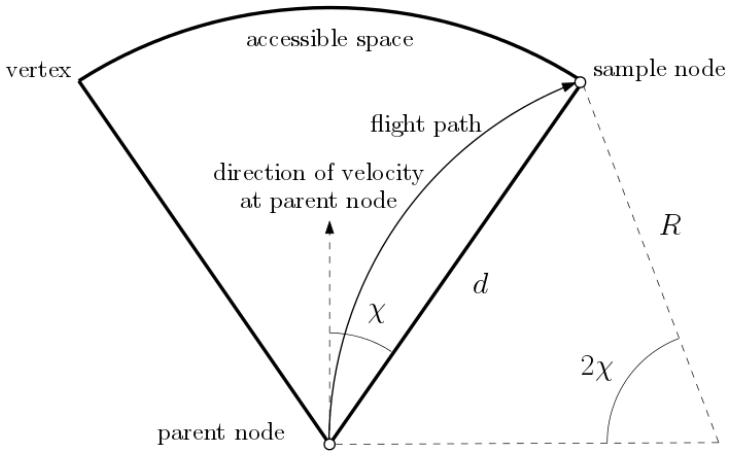


FIGURE 5.1: Accessible space.

the need for post-process trajectory optimization, which would be cumbersome given our complex, high-dimensional dynamics model (Section 2.1). As illustrated in Fig. 5.1, the flight path turn radius, R, is related to the accessible space by the distance between the parent and sample nodes, d, and the line-of-sight angle, χ:

$$R = \frac{d \cos \chi}{\sin 2\chi} = \frac{d}{2 \sin \chi} \tag{5.1}$$

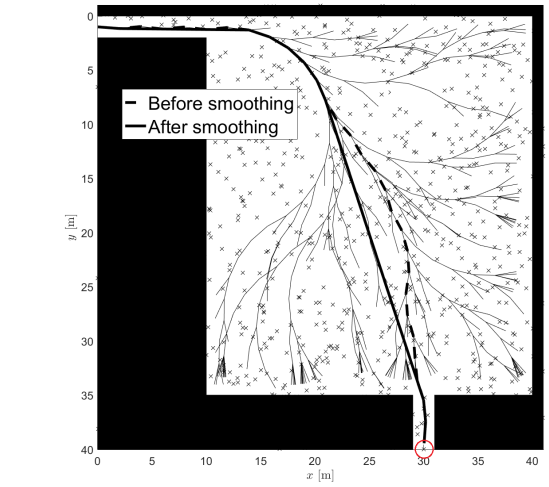
Generically, a turn radius is also given by $R = V/\dot{\psi}$, and thus it follows that the term we are interested in, the heading rate, is a function of d and χ:

$$\dot{\psi} = \frac{2V \sin \chi}{d} \tag{5.2}$$

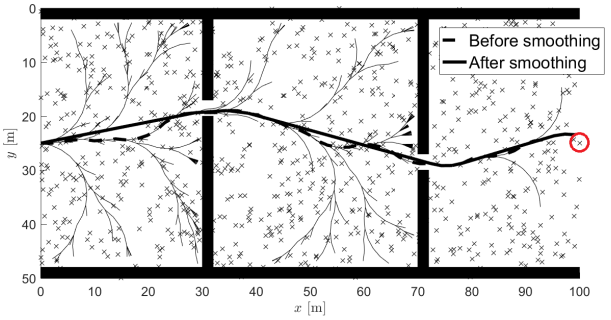
We conservatively limit the heading rate to $|\dot{\psi}|_{max} = 60^\circ\,\mathrm{s}^{-1}$, with $d = 2\mathrm{m}$ and $\chi = 12^\circ$. If a new node falls outside of its parent node's accessible space, the closest vertex of the accessible space is considered instead. This decision making process replaces the NEW_STATE function in the standard RRT algorithm [60], which returns a potential new node.

Once the random sampling algorithm has constructed a path, the automatic smoothing process samples pairs of nodes to shortcut unnecessary ones. Figure 5.2 shows generated paths through two predefined maps before and after smoothing.

The feedback controller in Section 2.2 was not designed for path-following, but rather to track a time-dependent trajectory of multiple states. Therefore, we next assign a time history to the nodes, and augment them with reference attitudes and a constant velocity.



(a) Map 1



(b) Map 2

FIGURE 5.2: Path plan before and after smoothing. Red circles represent goal regions.

This additional information can be derived from the path itself. A time history follows from the distance between points and the constant velocity. The reference heading profile is determined based on the direction between each pair of successive nodes. The reference roll angle is determined as a function of speed, and the distance and change in heading between successive nodes:

$$\tan\phi = \frac{2V^2 \sin\chi}{gd} \tag{5.3}$$

A derivation of the above expression, based on aircraft outer-state dynamics, can be found in [47]. We use an empirical formula for the pitch angle, calculated as a function of speed and roll:

$$\theta = c_1 + \frac{c_2}{V^2 \cos\phi}, \tag{5.4}$$

where c_1 and c_2 are coefficients tuned in simulations for maintaining constant altitude flight.

5.2.1.1 Integration of Knife-Edge Maneuver

Once the top-level motion plan has generated a conventional trajectory, the knife-edge maneuver is integrated into the plan. It is assumed here that the aircraft can identify narrow passages in the map, and place landmarks at the positions in the plan where knife-edge flight should begin and end. It is known from Section 4.4 that it takes 2.5m to transition into knife-edge while flying at $V = 5\text{m}\,\text{s}^{-1}$, therefore the aircraft requires at least 2.5m in advance of the passage to begin the maneuver. A deliberate feature of the maneuver design is that knife-edge flight can be superimposed on straight paths without compromising the original motion plan; the knife-edge stays on the straight-line path at a constant velocity.

As the aircraft is in flight and begins to execute the knife-edge maneuver, two things happen. For one, the precomputed feedforward control policies are activated and summed with the feedback control inputs. Additionally, the reference trajectory to the feedback controller is overridden. The aircraft will continue to track the same path, but the body-frame x-axis speed, pitch, and roll angle references are switched to the optimal control solutions for the duration of the maneuver. Due to the boundary conditions imposed in the optimal control problem formulation for the knife edge maneuver, see Table 4.5, these transitions are smooth.

5.2.1.2 Simulations

Simulations are run to test the entire proposed methodology of motion planning and control. The dynamics model of Section 2.1 is built into Simulink and simulations are conducted wherein the aircraft is tasked with tracking the motion plan.

For this demonstration, we use the smoothed paths displayed in Fig. 5.2. For Map 1, the aircraft starts and ends in wings-level flight before and after the entrance and exit to the environment. For both maps, tracking performance is illustrated in Fig. 5.3, for velocity and altitude, and Fig. 5.4, for the planar path. Figure 5.3 shows nearly constant altitude held throughout both maps, but the feedback controller does not perfectly stabilize about the desired velocity. Nevertheless, even during the most dynamic portions of the aircraft's flight - transitioning to and from the knife-edge maneuver - its actual motion rarely deviates more than $1\text{m}\,\text{s}^{-1}$ from the reference velocity. Figure 5.4 confirms that the

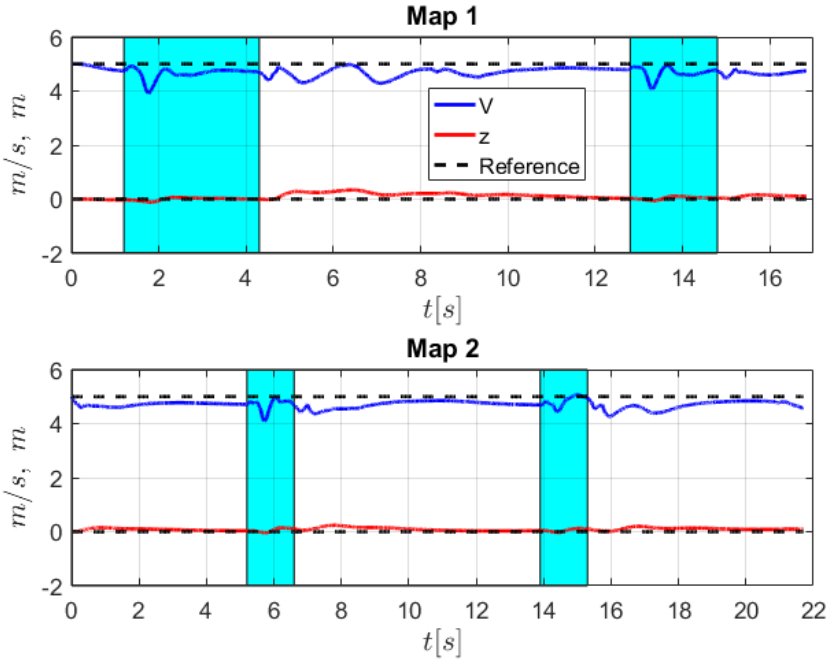
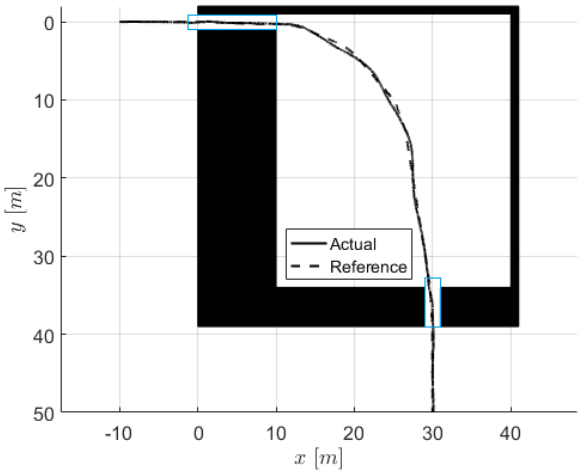


FIGURE 5.3: Velocity and altitude tracking for Maps 1 and 2 of Fig. 5.2. Knife-edge maneuver highlighted in light blue.

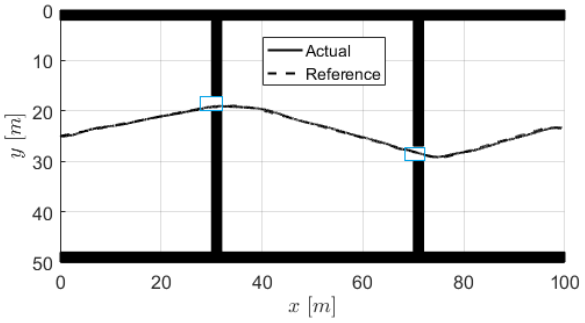
deviations in velocity have a negligible effect on the aircraft's ability to stay on the desired planar path, which is of greatest concern, given the presence of narrow passages. Furthermore, we can see from the different layout of Maps 1 and 2 that the control system is able to track the desired planar path whether the aircraft remains in the knife-edge trim condition through longer passages (Map 1), or transitions more immediately in and out of knife-edge flight through shorter passages (Map 2). Figure 5.5 shows a 3D visualization of the full flight through Map 1; the size of the aircraft has been scaled up by a factor of approximately 4.

5.3 Planner 2: Real-Time Planning with the Maneuver Space

The planner detailed in this section is more general and sophisticated in a number of ways: the maneuver space is more loosely constrained, the configuration space of the planner is three-dimensional, it can run in real-time, and the RRT tree is built using dynamically feasible motion primitives. For these reasons, when it came to flight testing, we focused on validating this planner.



(a) Map 1



(b) Map 2

FIGURE 5.4: Planar path following. Blue boxes outline where knife-edge is performed.

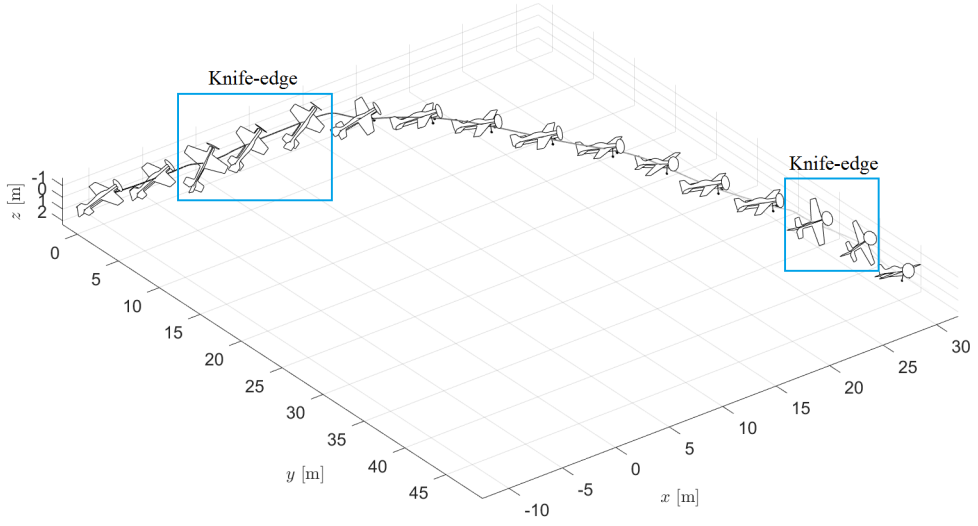


FIGURE 5.5: 3D visualization of flight through Map 1. Blue boxes outline where knife-edge is performed.

The planning algorithm is applicable to the case of a known, three-dimensional environment with static obstacles. The assumptions therein, that the environment is known and that the obstacles are static, exist to limit the scope of the problem to one that could realistically be addressed in this thesis. One may naturally wonder why we are concerned with real-time planning when the environment is known and static; and there are two reasons for this. First, a real-time application of RRT can build from the vehicle's initial condition as measured at the time. This is important for our application, since a fixed-wing UAV, unlike a robotic arm or ground vehicle, cannot easily be placed in a desired initial state. Secondly is that the planning methodology was developed with future research in mind. We sought to construct a framework that could be extended to more complex problems, involving environments that can only be mapped, or fully mapped, in real-time. Recommendations to this effect are noted in the conclusion.

Just like the previous planner, a tree of nodes is built by steering towards randomly generated points until the goal region is reached. The major difference here is that steering is done in three dimensions, and uses the dynamically feasible trajectories of the maneuver space (as opposed to straight lines). The focus of this section will be to describe the ways in which this motion planner deviates from the standard RRT algorithm. The deviations center mainly around the incorporation of the maneuver space, using the agile maneuvers intelligently, and the real-time functionality. A pseudo-code version of the high-level algorithm is presented in Algorithm 1. Note that the algorithm is specifically set up to guide the aircraft from an initial hover to a hover in the goal region.

Algorithm 1: RRT

input: Map, initial configuration ($\mathbf{x}_i$)

Initialize tree with $\mathbf{x}_i$
Generate hover-to-cruise primitive from $\mathbf{x}_i$ via `SteerAgile` (hover-to-cruise)
while *the goal region has not been reached* **do**
    while *time interval has not elapsed* **do**
        Generate a random point in the map, $\mathbf{p}_{rand}$
        `ExtendTree` towards $\mathbf{p}_{rand}$ (Algorithm 2)
    end
    `UpdateTree` (Algorithm 3)
end

5.3.1 Tree Data Structure

The tree that is built by the planner consists of nodes, each of which defines the state of the aircraft and the type of motion primitive that precedes it. In Fig. 5.6, the node n_2,

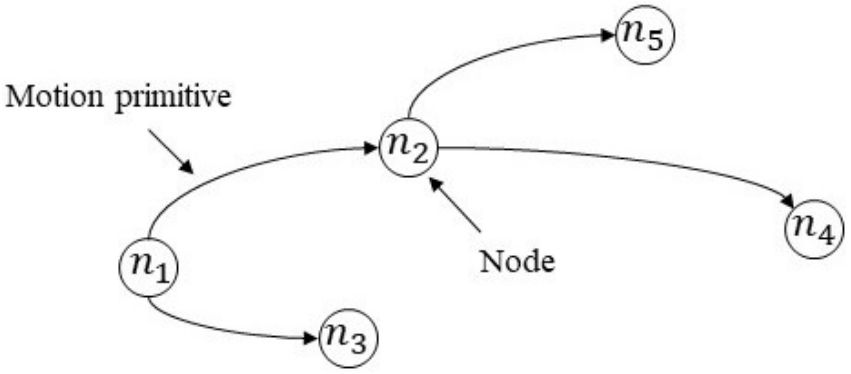


FIGURE 5.6: Tree nodes and motion primitives.

for example, contains not only the pose and time when the aircraft should reach it, but the edge (or motion primitive) that connects it to n_1. The full list of information stored in each node is as follows:

- Position (in Cartesian coordinates)
- Heading
- Time
- Type of preceding trajectory

The state of the aircraft at the node is defined by the position, heading, and time. The type of preceding trajectory denotes the type of primitive (trim or agile maneuver) that was used to arrive at that state from the previous node. If the preceding trajectory is a trim primitive, the type of preceding trajectory will include the yaw rate and climb/descent rate. In the case of an agile maneuver primitive, the type simply defines which of the agile maneuver primitives it is.

5.3.2 Extend Tree

The *Extend Tree* function aims to grow the tree by generating random nodes in the map and connecting them to the nearest in the existing tree. During the generation of random nodes, we introduce a slight goal node bias, sampling the end point instead of a random one every 40 iterations. By default, the algorithm searches the environment uniformly, and thus sampling the goal node every so often helps balances exploration with movement towards the desired end region. If the primitive extended from the nearest node ends up colliding with an obstacle, a new attempt is made with the next nearest node, and so forth for five iterations. These few iterations help build through narrow corridors and around walls [45]. The chosen value of five was arrived at via manual tuning. A value

too high results in the aforementioned benefit of the endeavor being lost, while too high a value needlessly slows down the algorithm while searching for connections in hopeless dead ends.

The *Extend Tree* function terminates in any of the following cases: a collision-free primitive is found, the list of tree nodes has been exhausted, or the maximum number of iterations through the list has been reached. Upon completion, the function outputs the node that is being extended away from, the primitive used for steering, and, in the case of a trim primitive, the time to remain along it.

The planning algorithm makes use of the agile maneuver primitives in specific ways. The plan is designed to begin from a hover, and thus the first primitive generated is always a hover-to-cruise maneuver. The cruise-to-hover maneuver is attempted every time it would land the aircraft in the goal region. The algorithm, therefore, always terminates with this maneuver, and thus with the aircraft in a hover. The aggressive turn-around maneuver is generated if a trim primitive extended from the nearest node results in a collision. This signals that the tree is headed towards an obstacle, and the aggressive turn-around maneuver can be used to immediately steer away from it, in a minimal amount of space. The maneuver is connected to the nearest node to which the random sample failed to connect. Note that the maneuver is only attempted after the first of the five iterations mentioned above. Although sequential turn-around maneuvers would be unlikely to occur anyway – because the end of the maneuver points the aircraft back into previously charted, obstacle-free territory – a simple amendment to the algorithm eliminates the possibility of this occurring. The basics of the *Extend Tree* logic are described in Algorithm 2.

Algorithm 2: ExtendTree

input: $\mathbf{p}_{rand}$

List nodes in order of nearness to $\mathbf{p}_{rand}$
foreach *node, η, in the list, if maximum number of iterations have not been reached* **do**
  if *a cruise-to-hover maneuver would land the aircraft in the goal region* **then**
    Generate maneuver primitive ρ_{prim} from η via `SteerAgile` (cruise-to-hover)
  else
    Generate ρ_{prim} from η via `Steer` ($\mathbf{p}_{rand}$)
    if *η is the first node in the list and ρ_{prim} results in a collision* **then**
      Generate ρ_{prim} from η via `SteerAgile` (aggressive turn-around)
    end
  end
  if *ρ_{prim} is collision-free* **then**
    break
  end
end

5.3.3 Steer

There are two steer functions, one for trim primitives, *Steer*, and the other for agile maneuver primitives, *SteerAgile*. These functions determine the connections of new nodes to the tree. The primary goal of the steer functions designed here is to efficiently expand the tree. The primary steer function, that for trim primitives, analytically determines which single trim primitive to use for the connection, and for how long to coast along it, i.e. the optimal cost-to-go. The function solves for yaw rate, $\dot{\psi}$, climb/descent rate, $\dot{z}$, and coasting time, Δt (as will be described in Eq. 5.5). This approach to steering highlights a salient feature of our methodology, which is that the size of the motion primitive library can be increased indefinitely without having any effect on the time spent creating connections. We found that in practice, for our purposes, nothing apparent is lost by failing to extend the tree to exactly meet the randomly generated nodes, nor neglecting to consider a larger subset of available connections (i.e. piecing together multiple primitives to connect nodes). Results demonstrating this observation are provided in Section 5.3.6.1.

The steer function for trim primitives searches in the neighborhood of circular arc parameters. It takes as input the configuration of the node it is steering away from, $\mathbf{p}_1(x_1, y_1, z_1)$ and ψ_1, and the point it is steering towards, $\mathbf{p}_2(x_2, y_2, z_2) = \mathbf{p}_{rand}$. Determining which trim primitive to use and for how long to coast along it is solved for analytically. This information is derived from the geometry of the circular arc connecting the two node points, as seen in Fig. 5.7. The point $\mathbf{p}_1$ is not in fact the node being steered away from; the node being steered away from is the one which comes after it by way of the time-delay. Referring back to Fig. 4.10, $\mathbf{p}_1$ would correspond with B' and $\mathbf{p}_2$ with C. The equations relating to Fig. 5.7 solve for the trim primitive (yaw rate, $\dot{\psi}$, and climb/descent rate, $\dot{z}$) and coasting time, Δt, that bring the aircraft as close as possible to $\mathbf{p}_2$:

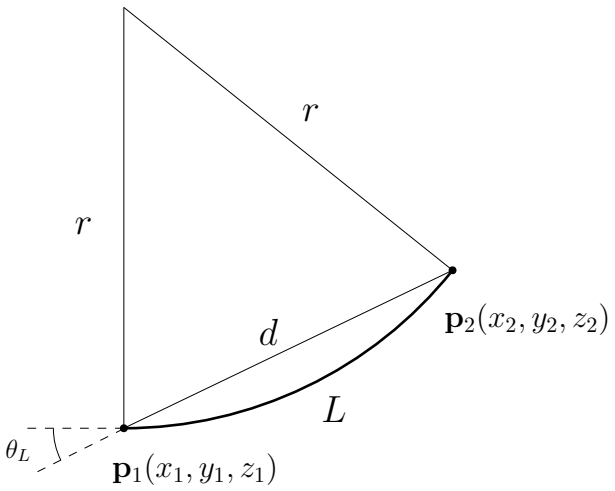


FIGURE 5.7: Top view of 3D circular arc defining trim primitive geometry.

$$
\begin{aligned}
d &= ||\mathbf{p}_2 - \mathbf{p}_1|| \\
\theta_L &= \arctan(\frac{y_2 - y_1}{x_2 - x_1}) - \psi_1 \\
r_{x,y} &= \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{2 \sin \theta_L} \\
L &= \frac{d\theta_L}{\sin \theta_L} \\
\dot{\psi} &= \frac{V}{r_{x,y}} \\
\Delta t &= \frac{L}{V} \\
\dot{z} &= \frac{z_2 - z_1}{\Delta t},
\end{aligned}
\tag{5.5}
$$

where θ_L measures the difference between the vector d and the heading, ψ_1, of the node at $\mathbf{p}_1$. The term $r_{x,y}$ is the projection of r on the horizontal plane. The desired constant speed, $V = 7\text{m}\,\text{s}^{-1}$, appears in these equations to calculate the coasting time. Given that there are a finite number of trim primitives in the maneuver space, the yaw rate and climb/descent rate calculated in Eq. 5.5 are each approximated to the closest available rates. In addition to the end node of the trim primitive, intermediate nodes are also returned by this function. This is useful to the planner in that it generates more tree node options to be steered away from in the subsequent *Extend Tree* phases.

The steer function that handles agile maneuver primitives requires as input only the type of agile maneuver (of the three) and the node to steer away from. The function uses this information to output the end node of the maneuver, the data of which is precomputed.

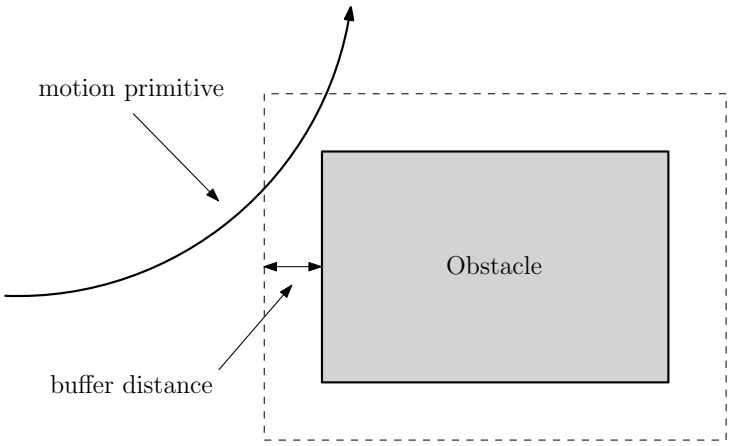


FIGURE 5.8: Collision check with obstacle.

No intermediate nodes are returned from this steer function because the planner and controller are not designed to exit agile maneuvers partway through.

5.3.4 Collision Check

The collision checking function detects if a primitive is outside the bounds of the environment or overlapping an obstacle. The function checks in intervals along the primitive and discards the primitive as a whole if any segment has a collision. The displacements of each of the three agile maneuvers are nearly restricted to the vertical plane and are pre-computed. Therefore, the collision check on agile maneuvers is trivial; it looks for any collision along a path with the same forward and vertical displacements.

As implemented here, the function detects collisions with obstacles that are rectangular prisms, based on geometric constraints. It could presumably be replaced with a function that can handle more complex obstacle geometries, so long as they can be approximated by polyhedrons. For an alternative approach based on circular or cylindrical obstacles and circular trajectories, see Paranjape et al. [47].

To account for the aircraft's geometry (i.e. that it is not a point mass) and non-ideal tracking performance from the controller, a buffer distance is added to all obstacles and environment boundaries. In effect, obstacles are inflated in the collision checker so that the aircraft stays a safe distance away from them. Figure 5.8 shows a motion primitive that would be discarded as having a collision because it falls within the buffer around the obstacle. This figure also shows the importance of checking for collisions in small intervals along the primitive. If the primitive were only checked for collisions at its ends, for instance, it would be deemed collision-free.

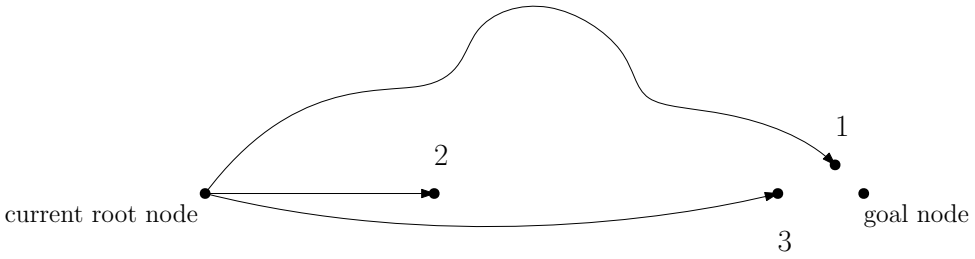


FIGURE 5.9: Nearness quantity.

5.3.5 Update Tree

The *Update Tree* function allows the planner to run in real-time, and is the point of communication between the planner and the control system. Its jobs are to choose which nodes of the tree the aircraft should follow until the next time the function is called, and to update the tree of nodes to account for the aircraft's real-time motion. The function is called iteratively as the aircraft moves up the tree.

The Update Tree function first determines the optimal node - of those available at the time - to guide the aircraft towards. It checks all nodes to find out which is 'nearest' (as will be defined) to the goal. It then determines how far along the tree to move in the direction of that node for the current iteration, i.e. how many nodes to commit to for one time interval, given the aircraft's dynamics. The aircraft commits to following these nodes, and they are sent to the control system to be tracked. In the unlikely scenario that the optimal node itself would be reached before the end of the time interval (and is not within the goal region), the node's children will be followed. If there are no such children to follow, the algorithm sends a cruise-to-hover primitive to the controller. While the aircraft is hovering, the algorithm can rebuild a tree, starting over again with the hover-to-cruise maneuver.

The 'nearness' quantity is calculated based on the length of the path to the node, the straight-line distance between the node and the goal, and the distance between the current root and the node, see Eq. 5.6. In the hypothetical situation depicted in Fig. 5.9, node number 3 would be 'nearest' to the goal of the three options. It is not as close to the goal as node 1, nor is the path to it from the root node as short as node 2, but the balance of these quantities - evaluated by Eq. 5.6 - makes it 'nearest' the goal.

$$nearness = \frac{length\ of\ path\ to\ node + distance\ from\ node\ to\ goal}{distance\ from\ current\ root\ to\ node} \tag{5.6}$$

Algorithm 3: UpdateTree

input: η_{root}, $\mathbf{p}_{goal}$, e_p

Find tree node, η_{opt}, nearest $\mathbf{p}_{goal}$
Get list of nodes connecting η_{root} to η_{opt}
Initialize list of nodes, l, to send to controller **for** $\eta_{temp} \leftarrow \eta_{root}$ **to** η_{opt} **do**
    if η_{temp} *is in goal region* **or** η_{temp} *exceeds time interval* **then**
        break
    end
    if η_{temp} *is* η_{opt} **then**
        if η_{opt} *has children* **then**
            Continue along children of η_{opt}
        else
            Add cruise-to-hover node to l
        end
    else
        Add η_{temp} to l
    end
end
if $e_p > \epsilon$ **then**
    Re-orient l to align with aircraft's actual pose
    Prune tree of all nodes other than l
end
Send l to controller
Prune tree of infeasible nodes

To account for the aircraft's real-time motion, the Update Tree function also prunes the tree of the nodes that become infeasible as a result of the commitment (nodes that will be 'behind' the aircraft in time - and their children - as it moves 'up' the tree). The Update Tree function is presented in Algorithm 3.

5.3.5.1 Re-planning

The steps in Algorithm 3 under the conditional statement, **if** $e_p > \epsilon$, constitute a re-planning operation. The re-planning operation exists to account for the fact that the aircraft may not track the planned trajectory as closely as is needed or desired. If the position error of the aircraft, e_p becomes too large (greater than a constant magnitude, ϵ), the re-planning step in the update phase is triggered. This function performs two actions, the first of which is to modify the nodes being sent to the controller so that they align with the actual position and heading of the aircraft. Re-orienting the nodes to align with the aircraft's actual position involves revisiting Eq. 5.5. The motion primitives themselves remain the same, i.e. $\dot{\psi}$, $\dot{z}$, and Δt are known, but the positions and headings of each node must be recalculated. This is done through rearrangement of Eq. 5.5:

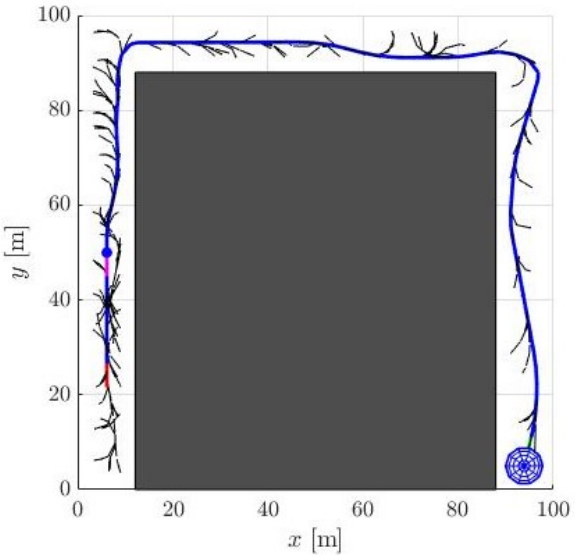
$$
\begin{aligned}
x_2 &= x_1 + \left(\frac{V}{\dot{\psi}} \sin(\psi_1 + \dot{\psi}\Delta t) - \frac{V}{\dot{\psi}} \sin\psi_1 \cos(\arcsin\frac{\dot{z}}{V}) \right) \\
y_2 &= y_1 + \left(-\frac{V}{\dot{\psi}} \cos(\psi_1 + \dot{\psi}\Delta t) + \frac{V}{\dot{\psi}} \cos\psi_1 \cos(\arcsin\frac{\dot{z}}{V}) \right) \\
z_2 &= z_1 + \dot{z}\Delta t \\
\psi_2 &= \psi_1 + \dot{\psi}\Delta t
\end{aligned}
\tag{5.7}
$$

If there happens to be an agile maneuver in the list of nodes to be modified, the modification is simply a coordinate transformation, since the planner only deals with the end points of agile maneuvers.

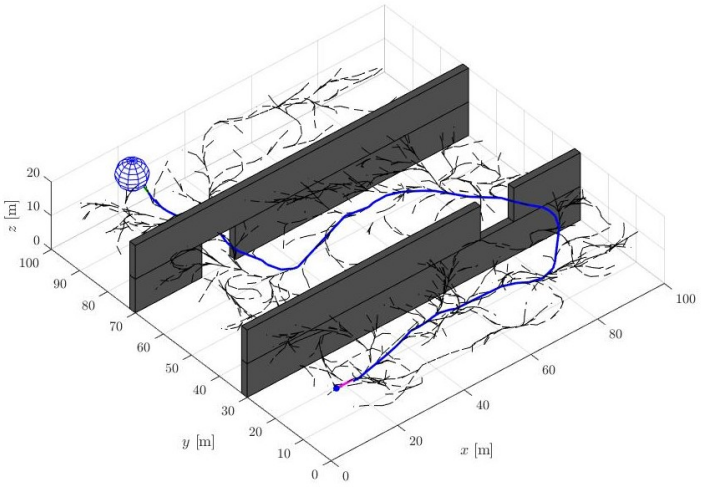
The other action taken during re-planning is to prune the tree of all nodes other than the ones in the list being sent to the aircraft. While it may seem detrimental to throw away these previously generated nodes, the algorithm is very efficient at building (or re-building) a tree in real-time - this is tested in Section 6.1.3. The alternative, to keep the tree nodes, would require translating each node and re-checking each primitive for collisions. This is a much more costly process (namely, the collision checking), that would not be of any great benefit, given how quickly the tree can be re-built. The worst case scenario of the Update Tree function remains, in that the aircraft will be commanded to perform the cruise-to-hover maneuver if no other collision-free options for pursuing the goal node exist.

5.3.6 Simulations

Simulations were run to validate the motion planner. Figure 5.10 displays two sample cases of planning in highly-constrained environments. In the figures, the axes stretch the length of the environment and the grey objects represent obstacles. The blue dots are the aircraft's initial points, and the blue spheres are the desired goal regions. The trajectory flown is also colored blue, except for the agile maneuvers, where magenta is the hover-to-cruise maneuver, red is the aggressive turn-around, and green is the cruise-to-hover maneuver. The remaining parts of the tree, which were not flown, are in black. Figure 5.10a shows a top view of a case where the aircraft had to fly through corridors. When the aircraft starts at $(x, y) = (5, 50)$, its heading is pointing downwards with respect to the figure, and thus the aggressive turn-around maneuver ends up being used to change direction and move towards the goal. Figure 5.10b shows a 3D environment in which the



(a) Map with corridors



(b) Map with narrow passages

FIGURE 5.10: Sample motion plans.

aircraft has to pass through narrow gaps. Note that the paths are not optimally short but that they are smooth in the horizontal plane, i.e. continuity of heading is ensured.

5.3.6.1 Comparison to Dubins Curves

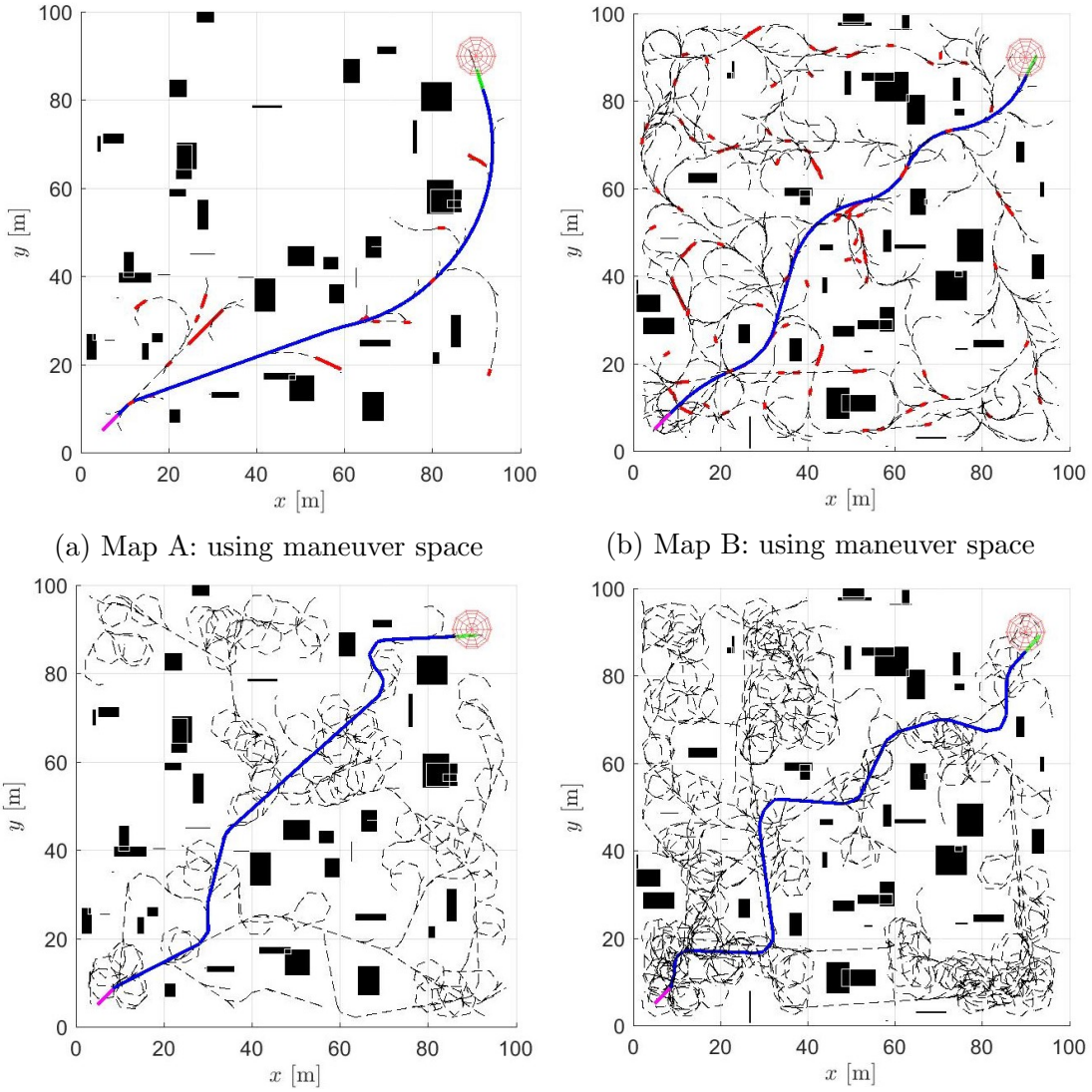
Simulations were also run to contrast the motion planning framework against a baseline approach. The approach employed for this purpose is RRT with Dubins curves, which is a commonly used technique for path-planning with ground vehicles [116, 117] and fixed-wing UAVs [72, 118–120]. Dubins curves are minimum-distance paths between two points with prescribed headings, for a vehicle that is subject to the constraints of the Dubins kinematic model [94]:

$$\begin{aligned} \dot{x} &= V\cos\psi \\ \dot{y} &= V\cos\psi \\ \dot{\psi} &= u, \end{aligned} \tag{5.8}$$

where (x, y) is the position of the vehicle, V is a constant speed, and ψ is the heading. By incorporating an additional configuration variable for altitude, the Dubins model has been extended to 3D problems [121], but for simplicity we will use the 2D model here for comparison. From these equations, we see the most apparent difference between the two approaches, which is that Dubins curves are the product of a very simple kinematic model evolving on the configuration space $\mathcal{C} = \mathbb{R}^2 \times \mathbb{S}^1$. Using this model, $\mathbf{p}_1$ and $\mathbf{p}_2$ of Fig. 5.7 (in 2D and with prescribed headings) are connected using the shortest feasible path. The solution to this problem, given the constraints of Eq. 5.8, is proven to always consist of minimum-radius circular arcs and/or straight line segments [94]. Therefore, using the Dubins curve approach, every time the trajectory changes heading, it does so using the most aggressive turn.

To employ the Dubins curve approach, we replaced our steer functions with one that solves for the optimal Dubins path. Both motion planning frameworks were run on a 100 m by 100 m map that includes 50 randomly generated obstacles. Representative samples of trajectory solutions are shown in Fig. 5.11. To compare the two approaches, Maps A and C use the same layout, as do Maps B and D. In the figures, the axes stretch the length of the environment, the black objects represent obstacles, and the orange spheres are the desired goal regions. The trajectories flown are colored blue, except for the agile maneuvers, where magenta is the hover-to-cruise maneuver, green is the cruise-to-hover maneuver, and the aggressive turn-arounds are colored red. The remaining parts of the tree, which were not flown, are colored black. Note that we had to incorporate a part of our maneuver space, the hover transitions, into the Dubins curve approach just to be able to solve the desired planning problem, which includes stationary initial and final states.

The few cases plotted in Fig. 5.11 highlight features of the proposed approach. Because the maneuver space includes many more primitives than there are Dubins curves, the turns of the trajectories tend to be smoother and less aggressive. Although the Dubins curves do solve for the shortest paths connecting individual nodes, this optimality tends to be lost in terms of the full trajectory solutions, as can be seen in Figs. 5.11c and 5.11d.



(a) Map A: using maneuver space

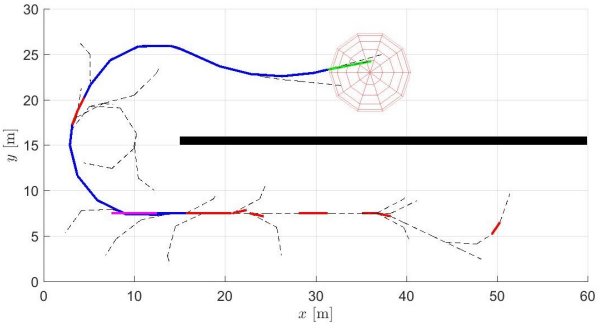
(b) Map B: using maneuver space

(c) Map C: using Dubins curves

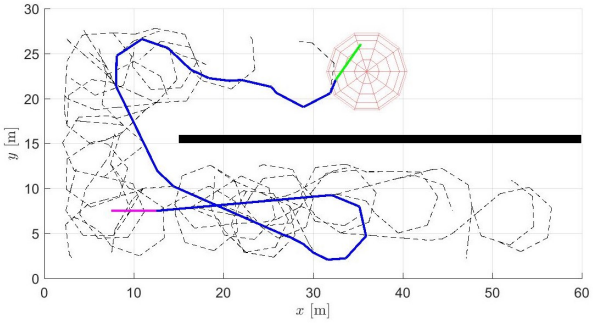
(d) Map D: using Dubins curves

FIGURE 5.11: Motion plans through a 100 m by 100 m map with 50 randomly generated obstacles. Maps A and B use the proposed maneuver space approach; Maps C and D use Dubins curves.

In a different map, we investigate how the two approaches would fare in a situation that necessitated a near 180-degree turn-around; the results are plotted in Fig. 5.12. While Dubins curves can indeed be used to generate a feasible path through this map, there are disadvantages to the approach relative to ours. The planner using Dubins curves generally takes more time to solve such a problem (the greater number of black paths in Fig. 5.12b signifies the longer time it took to find a feasible solution). This is because many positive collision checks have to occur before the Dubins curves can navigate a path out of the dead end. Using the proposed maneuver space, however, one of the first positive collision checks results in the generation of the functionally-designed aggressive turn-around maneuver, which immediately provides the beginning of a way out. Figure 5.12 also illustrates how



(a) Map E: using maneuver space



(b) Map F: using Dubins curves

FIGURE 5.12: Motion plans involving a retreat from a narrow corridor with a dead end. Map E uses the proposed maneuver space approach; Map F uses Dubins curves.

our planner tends to generate smoother trajectories through narrow corridors, where the Dubins approach bounces around between the minimum-radius curves.

We evaluated the performance of each algorithm, in terms of computational efficiency and cost (length of the path solution). To do so, we programmed them both on the test platform's computer (an ODROID XU4 – see Section 6.1.3). Each algorithm was run 1000 times over the map of Fig. 5.11, and again over the map of Fig. 5.12. Only the former map uses randomly generated obstacles, but in either case the planner itself is random in its sampling of the environment, making each solution unique. For the map of Fig. 5.11, the average time to find a feasible trajectory using the maneuver space was 150 milliseconds, compared to 96 milliseconds using Dubins curves. The average path length was 180 m using the maneuver space, and 182 m using Dubins curves. These results reinforce the point that the optimality of individual Dubins paths does not carry over to the full solution. For the map of Fig. 5.12, in which the aircraft had to retreat from a dead end, the average computation time and path length using the maneuver space was 20 milliseconds and 71 m, respectively. With Dubins curves, the algorithm took slightly longer, 26 milliseconds, and averaged a significantly longer path of 102 m.

A final matter of differentiation between the two approaches is how well they lend themselves to the trajectory tracking problem. In this respect, there are a few things to note about the Dubins curves approach. There are no transition maneuvers between curves, and the kinematic model assumes accelerations can be controlled directly. Being restricted by the aircraft's dynamics, no such arbitrary accelerations can in fact be generated. Also recall that the only turns available are the minimum-radius turns. A conservatively chosen turn-rate constraint will limit the abilities of the planner to navigate around obstacles, while a high turn-rate will require the aircraft to track a more aggressive trajectory. Minimum-radius turns without transitions make the tracking problem demanding, and on top of this, the Dubins model offers no feedforward control input solutions for the aircraft's actuators. We revisit this matter in Section 6.5.1.

5.4 Benefits and Limitations of RRT

The RRT algorithm was determined to be the most suitable planning algorithm for the objectives of this research, but it is worth noting both its benefits and limitations. The main appeals of RRT in any application are that it efficiently explores high-dimensional state spaces, and can easily handle obstacles and differential constraints. The algorithm can produce dynamically feasible trajectories in a short period of time. Another strong appeal of RRT for our work is that it allowed integration of the maneuver space. When a random point is sampled, the connection drawn to it from the tree is not required to reach the random point exactly to maintain the integrity of the algorithm. In fact, it is common of RRT implementations to limit the length of the connections, such that the random samples are controlling the direction of the tree growth but not the rate. This is something the maneuver space can be used for - growing the tree in a desired direction. It cannot, however, guide the aircraft exactly to a commanded point. In terms of the advantages of RRT, it is also worth noting that the algorithm is widely used and actively being developed; variations and improvements are consistently surfacing in the literature.

The main disadvantage of RRT is that the plan produced by the algorithm is likely suboptimal, which is generally the price of efficiency. In some applications, RRT is used as an initial planning step, then followed by a post-processing method that may smooth or even optimize the trajectory. There is also an optimal version of RRT available, called RRT*, but employing this version of the algorithm requires the ability to find exact connections between nodes. The maneuver space is not appropriate for this operation, which entails solving costly boundary value problems that dominate the complexity of the algorithm. Furthermore, RRT* takes many iterations before any significant benefit is seen over the

standard algorithm, i.e. even if RRT* could be employed using the maneuver space, it is unlikely we would see any noticeable difference in optimality if we aimed to maintain the short execution time of the planner.

Another drawback of RRT is that completion, i.e. reaching the goal, is only guaranteed given infinite time. In practice, however, we never found this to be a serious issue. Rarely was the algorithm unable to find a trajectory to the goal in more than two seconds. Furthermore, with our proposed implementation, the user could always decide to restart the motion planner if it did not immediately find a solution. Because the planner is based around random sampling, restarting it would begin drawing a completely new tree, even given the same initial condition.

The algorithm is also known to function poorly in certain types of environments. Like many other planning algorithms, RRT can struggle to pass through narrow halls and passages, and is much better at exploring open spaces with dispersed obstacles than maze-like maps. As described in Section 5.3.2, this deficiency can be mitigated with a simple alteration of the Extend Tree function. Finally, we note that while the planner produces feedforward control inputs, it is detached from the feedback controller. The planner outputs a trajectory that must be tracked well; failings of the feedback controller are not addressed within the planner, except via re-planning if errors grow sufficiently large.

Chapter 6

Simulation and Flight Test Validation

In this chapter we discuss the setup and results of simulations and flight tests. A first set of simulations and flights was aimed at validating the methodology for generation and control of the agile maneuver primitives. The agile maneuvers were performed autonomously to test how well they could be tracked. In the second set of testing, the real-time motion planner is evaluated. Both sets of flight tests demonstrate a significant level of implementation. For one, all tests are performed using only on-board sensing and computing – including the real-time motion planning. Additionally, unlike some of the relevant literature [49, 73], no launching device is used, meaning that maneuvers and motion plans are not begun from precisely controlled initial conditions; the tests are initiated midair, either from level flight or a hover.

This chapter begins with a description of the aircraft platform and its various components used for autonomous flight testing. Section 6.2 will describe the MATLAB/Simulink simulation environment. Section 6.3 will discuss the hardware-in-the-loop setup that was used as an intermediate step between Simulink simulations and flight tests. The first set of tests on agile maneuver performance is described in Section 6.4, and the tests involving the motion planner are discussed in Section 6.5.

6.1 Platform Description

This section details the aircraft platform. We discuss the basic layout of the aircraft on which the autonomous hardware is mounted, as well as the autopilot flight controller and

the computer used to run the motion planning algorithm.

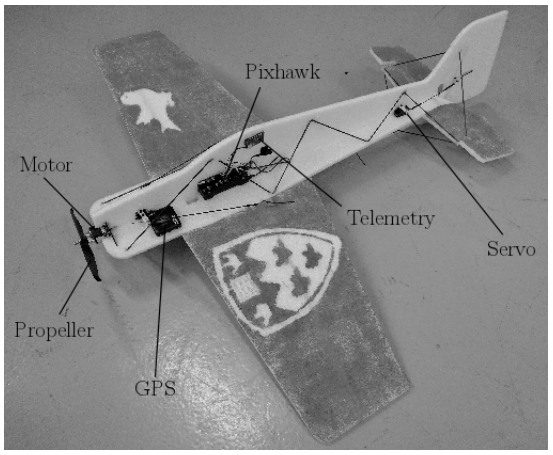
6.1.1 Aircraft

The aircraft used for flight test experiments is constructed around the McFoamy airframe by West Michigan Park Flyers. The airframe is meant for RC piloting, and was designed for smooth flying, aggressive maneuvers, and easy hovering. It is made of EPP foam, has a 0.86 m wingspan, and weighs under 0.2 kg out of the box. The airframe is intended to be loaded with servos, carbon fiber reinforcement rods (that support the fuselage), a receiver for the RC remote, a battery, an electronic speed controller (ESC), and a motor and propeller. For flying autonomously, we also had to add an autopilot flight controller, an ODROID computer, a GPS, and a radio telemetry kit. The full list of equipment attached to the airframe is as follows:

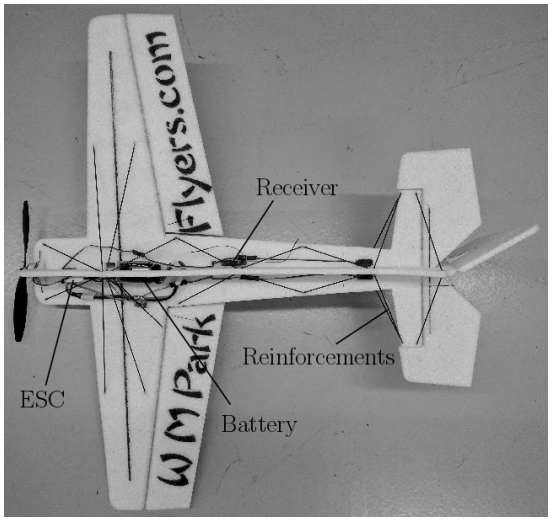
- Great Planes RimFire 400 Outrunner brushless DC motor
- Electrifly Powerflow 10×4.5 propeller
- Electifly Silver Series 25A brushless electronic speed controller
- Venom 50C 3S 850mA 11.1V LiPo battery
- 4 × HiTEC HS-65MG metal gear feather servos
- Pixhawk flight controller
- 3DR uBlox GPS
- 2mm-diameter carbon fiber reinforcement rods
- Futaba R6303SB micro receiver
- HKPilot micro radio telemetry kit
- ODROID XU4 single board computer

Equipped with only the parts needed for RC piloting, the weight and weight distribution of the aircraft are ideal for agile flight. Adding the extra parts needed to fly autonomously, we aimed to keep the weight low, and the weight distribution minimally affected. Parts of the foam fuselage, near the middle of the wing, are cut away to make room for the autopilot and ODROID near the aircraft's center of gravity. The LiPo battery is bigger and heavier than what is typically used for RC flight because, in addition to the motor, it has to power the two computers. Because it is relatively heavy, the battery is also placed near the aircraft's center of gravity. To avoid magnetic interference with other components, the GPS is mounted on the aircraft's nose.

The radio telemetry kit is used for communication between the aircraft and a ground station. MAVLink [122] is the protocol used for communication between the Pixhawk



(a) Top view of aircraft



(b) Bottom view of aircraft

FIGURE 6.1: Aircraft platform setup.

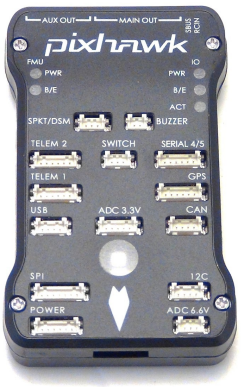
flight controller and the ground station software, QGroundControl (QGC). QGC is only used to calibrate sensors and tune gains on the fly; the aircraft does not need the ground station to fly autonomously.

Because of the extra weight on the aircraft, the motor we chose is more powerful than the ones typically used for RC flight. To keep the motor safely secured to the airframe, a custom-designed motor mount was 3D printed in the McGill Aerospace Mechatronics Lab to replace the mount that comes with McFoamy. Figure 6.1 shows the aircraft assembly for the first set of flight tests, for which the ODROID was not needed. For the flight tests involving motion planning, the battery was moved further towards the aircraft's nose, and the ODROID was placed directly behind it. For the first set of tests, the aircraft weighed 0.484 kg. Fully equipped, i.e. with the ODROID, the aircraft weighs 0.576 kg.

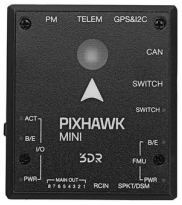
6.1.2 Pixhawk

The autopilot flight controller used on the aircraft is the *Pixhawk*. It runs the PX4 flight stack on a NuttX OS. Among other standard operations, the autopilot handles RC and desktop communication, data logging, sensor fusion, interpreting the motion plan, and control. The control system is programmed as a custom module within the PX4 open-source flight stack. This module implements the feedback control laws, and the control loop runs at 200 Hz. For the second round of flight tests, the control system module is additionally responsible for reading in the motion plan and converting it into a time-dependent trajectory with feedforward control inputs. The maneuver space developed in Chapter 4 is stored as a library of maneuvers on a micro SD card that is inserted in the Pixhawk. All the relevant information detailing the maneuvers is here and can be accessed by the control system module. For trim primitives, the states and control inputs that define each primitive are stored in the form of a look-up table. The agile maneuver primitives - because they are finite-time transitions - are stored in the form of time-dependent reference trajectories and feedforward control inputs; as matrices that are interpolated by time within the control system. When the planning algorithm is running, it tells the control system which maneuver to execute at a given time. The controller seeks out this maneuver from the library in order to compute the exact trajectory to track, and feedforward control inputs to use. As such, the planning algorithm does not need to deal with the contents of the library, it only needs to know basic characteristics of the maneuver space, in terms of the spatial geometry of the primitives. For instance, unlike the control system, the planner does not need to know the time-dependent trajectory, nor the control inputs corresponding to an agile maneuver. It simply needs to know where the maneuver begins and ends, and with what value of heading.

Two different Pixhawks were used throughout the flight tests. The first set of tests - on agile maneuver performance - used the *Pixhawk 1*, see Fig. 6.2a. The second set of tests - on motion planning - used the *Pixhawk Mini*, see Fig. 6.2b. The main difference between the two autopilots is their size and weight. The Pixhawk Mini is approximately a third the size and half the weight of the Pixhawk 1. Nevertheless, it has more powerful processors and sensors. The only reason the Pixhawk 1 was used for the first set of flight tests is that the Pixhawk Mini did not exist at the time. The sensors that the Pixhawks are housing include two inertial measurement units (IMUs), a gyroscope, a magnetometer, and a barometric pressure sensor. The only other sensor used on the aircraft is an external GPS. State estimation is performed using the flight stack's default extended Kalman filter (EKF). The EKF fuses measurements from the Pixhawk's internal sensors and the GPS to estimate the aircraft's full state vector.



(a) Pixhawk 1



(b) Pixhawk Mini

FIGURE 6.2: Pixhawk autopilot flight controllers.

6.1.3 ODROID

The motion planning algorithm requires more processing power and memory than the Pixhawk can provide. Instead, it is programmed (in C++) on an *ODROID XU4*, see Fig. 6.3. The ODROID is a single-board computer with a 2GHz quad-core processor and 2GB of RAM. The board runs Ubuntu on a Linux kernel. The ODROID is connected to the Pixhawk via an FTDI USB-to-UART cable. Communication goes both ways and uses the MAVLink protocol. When the motion planner is triggered to begin, the Pixhawk sends the aircraft's initial pose to the ODROID. As the motion planning algorithm runs, the ODROID sends individual nodes to the Pixhawk (in the Update Tree function), which are read within the control system module. The data stored in the nodes is interpreted by the Pixhawk's control system as time-dependent trajectories and feedforward control inputs.

Before conducting flight tests, the motion planning algorithm of Section 5.3 was repeatedly tested on the ODROID to gain insight into the efficiency of the algorithm and the processing capabilities of the computer. Twenty runs of each of the maps in Section 5.3.6 were performed on the ODROID. The average time to find a feasible path to the end goal was 0.5 s, and the maximum time never exceeded 5 s. There were no failure cases, although we cannot make any theoretical guarantees for finding a feasible path in

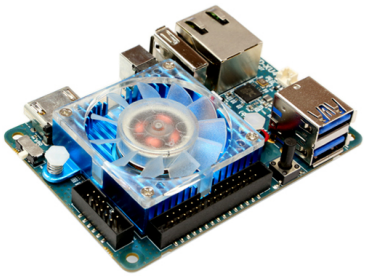


FIGURE 6.3: ODROID XU4.

a reasonable amount of time. Note that the execution time of the algorithm is subject to the size and complexity of the environment.

Tests were also run on the ODROID to evaluate the practicality of employing the re-planning step (as described in Section 5.3.5.1). We questioned whether the planner was efficient enough to recover from discarding ('pruning') almost all of the tree nodes in real-time, during flight. Using the map of Fig. 5.10b, we ran the motion planning algorithm again twenty times. This time, the planner was left to run until the Update Tree function had commanded a path that ended in the goal region; as though the aircraft were in flight and the algorithm were running in real-time. Instead of setting up the ODROID in a simulation loop with the aircraft dynamics model, we simply programmed fake position errors into the algorithm such that the re-planning step would be repeatedly triggered. For each run, the re-planning step was triggered twice, as though the aircraft had deviated too far from the commanded path. Each time the re-planning step occurred, the algorithm was able to rapidly rebuild a new tree. By the next time the Update Tree function was called after re-planning - it is called every half second - the tree would have already grown to hold approximately 1000 new nodes, on average. For reference, the tree rarely ever grew to have many more than 2000 nodes at a time, for the map in question. In the twenty runs, nineteen successfully resulted in a path to the goal region. In the one other case, the planner got stuck and had to send a command to perform the cruise-to-hover maneuver before reaching the goal region. It cannot necessarily be determined that the re-planning step was the cause of this failure. We noted that the number of tree nodes in subsequent calls of the Update Tree function - before and after re-planning - was barely affected, and thus we can at least rule out the notion of the failure being caused by a lack of trajectory options post-tree-pruning.

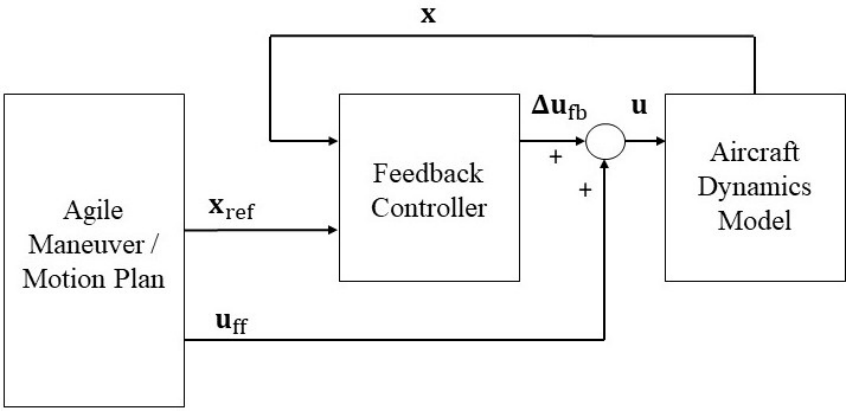


FIGURE 6.4: Block diagram of simulation architecture.

6.2 Simulation Environment

Simulations are performed in MATLAB/Simulink prior to conducting flight tests. Being much less expensive and more convenient, simulations were a valuable tool for rapid development and preliminary validation. Simulations were used to evaluate trajectory generation, motion plan generation, and the control system. In particular, the simulation environment was useful for developing and tuning the feedback controller, and also provided a sanity check of the full control system architecture, i.e. the combination of feedforward and feedback inputs.

Simulations were run for testing agile maneuver tracking and full motion plan tracking, and will be discussed in those respective sections. Where appropriate, simulation results will be contrasted with experimental results. A block diagram of the simulation is shown in Fig. 6.4. The agile maneuver trajectory or motion plan is output as a reference trajectory, $\mathbf{x}_{\text{ref}}$, and feedforward control vector, $\mathbf{u}_{\text{ff}}$. As the simulation is running, the output of the feedback controller, $\Delta\mathbf{u}_{\text{fb}}$, is summed with the feedforward component to produce the full control input, $\mathbf{u}$. The full aircraft dynamics model of Section 2.1 is modeled in Simulink.

6.3 Hardware-in-the-Loop Testing

To mitigate the transition from pure simulations to flight tests, an intermediate step of hardware-in-the-loop (HIL) testing is performed. The HIL testing captures the complexities and issues associated with a real-time implementation of the control system on the autopilot board. These include sensor noise, state estimation errors, controller discretization, and timing delays. By incorporating these factors, the HIL testing provides

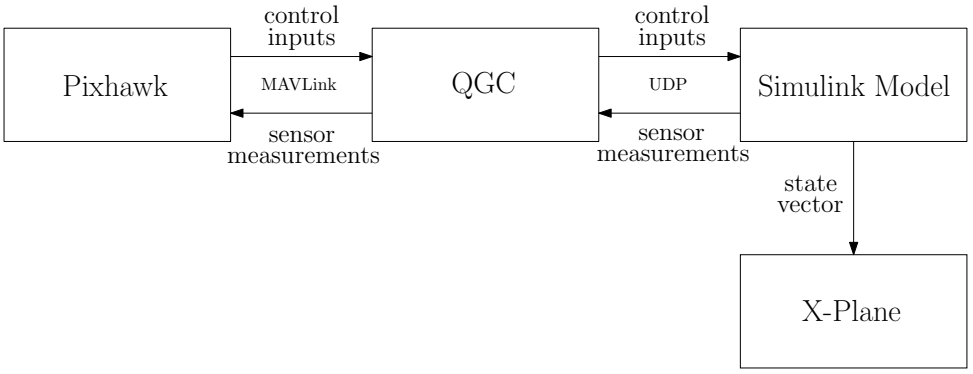


FIGURE 6.5: Hardware-in-the-loop configuration.

a further level of validation of the control system prior to flight testing. The HIL tests also ensure that implementation mistakes do not result in crashes or wasted time during flight testing; in that respect, the value of the HIL setup cannot be overstated.

The HIL configuration is illustrated in Fig. 6.5. The setup includes the Pixhawk autopilot, the ground control station, the Simulink aircraft dynamics model, and X-Plane. The ground control station sits between the flight controller and the dynamics simulator to act as a user interface. Our uses for the QGC interface include tuning gains on the fly, downloading data logs, and configuring the RC transmitter. The Pixhawk communicates with QGC using the MAVLink protocol via a USB connection. The ground control station in turn talks to the dynamics model via UDP messaging. By design, the PX4 flight stack can be integrated with X-Plane to harness their physics engine. However, given the high fidelity of our aircraft dynamics model, we use it in place of the X-Plane model. The X-Plane software stays in the loop to provide 3D visualization, which can be useful for interpreting the simulation outputs, i.e. the aircraft's motion.

The move from a pure simulation environment to the hardware-in-the-loop setup introduces outlier data and noisy signals (the characteristics of the sensors' noise is artificially generated in Simulink). Achieving effective autonomous flight in the HIL setup thus required detection and removal of outliers, as well as signal smoothing. The absence of these factors in the pure simulation environment meant that the feedback controller gains tuned there were not appropriate for the HIL, nor would they be for actual flight, and thus gains were re-tuned in the HIL. While the gains tuned in the pure simulation environment would be too large for flight test implementations, the gains tuned in the HIL were much more appropriate. This was a major benefit of the HIL implementation; having to tune the gains down in flight tests from the values attained in the pure simulation environment would be a tedious and time-consuming process that would have increased the likelihood of crashing the aircraft.

Details on outlier detection and removal, signal smoothing, and the general HIL setup are discussed in [123]. Results of various aerobatic maneuvers performed in the HIL, using the same aircraft model, can be found in [99]. In this thesis we focus only on the results of the pure (Simulink) simulations and flight tests. Although the HIL served as a useful tool for all the reasons stated, we found the pure simulation environment better suited for generating reproducible results that could be compared to flight tests. The pure simulation environment provides a more convenient way to generate, capture, and analyze data; and contrasting these results to those of experiments is useful for exposing practical implementation issues. As the chapter proceeds, the simulations we discuss will refer to pure simulations.

6.4 Agile Maneuver Tests

The first set of flight tests were used to validate the methodology of trajectory generation and control of the agile maneuvers developed in Section 4.2. Under evaluation was whether the motion primitives were in fact feasible, and whether the combined feedforward and feedback control scheme would effectively track the agile maneuver trajectories. Tests were performed for the aggressive turn-around, the cruise-to-hover, the hover-to-cruise, and the knife-edge maneuvers.

The concept of trim primitives had not been explored at the time that these tests took place. At the beginning of the second round of testing - the following year - some of the trim primitives were tested in a similar manner to the agile maneuvers. The results are not discussed in the thesis since they are trivial in comparison to the agile maneuver primitives.

6.4.1 Flight Test Implementation Details

Flight tests are performed in the *Concordia University Stinger Dome*. The dome is made of fabric that is GPS-transparent. The following procedure is adhered to each time an agile maneuver is tested. Using the Futaba T7C transmitter, an expert RC pilot manually has the aircraft take-off. The pilot then flips a binary switch on the transmitter (programmed via QGC and a custom PX4 module), putting the aircraft into fully automatic straight and level flight. Straight and level flight commands the trim conditions associated with the speed of $V = 7\mathrm{m\,s^{-1}}$, and the heading estimated when the switch was flipped. After the aircraft stabilizes at the desired speed, in approximately 1 to 3 s, the pilot flips

another switch to trigger the agile maneuver. In the case of the ATA, for example, the maneuver terminates with the aircraft recovering to automatic straight and level flight with the opposite heading. From there, the pilot switches back to manual control to land the aircraft.

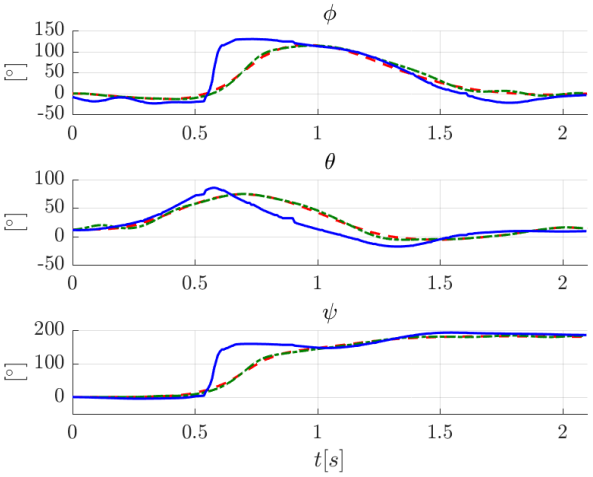
6.4.2 Aggressive Turn-Around

Results for the aggressive turn-around maneuver are displayed in Figs. 6.6 and 6.7. Simulations and experimental results are contrasted with the optimization solution, i.e. the reference trajectory and feedforward control found in Section 4.2.1 (note that simulations refer to Simulink simulations, not HIL simulations). Flight tests involved repetitions of each maneuver, however, for clarity the plots show only one experiment. In order to compare simulation and flight test results to the optimal maneuver, the optimization states and controls plotted are the direct solution associated with $V_0 = 7\mathrm{m\,s^{-1}}$; they have not been contaminated by the velocity parametrization process of Section 4.5.1.

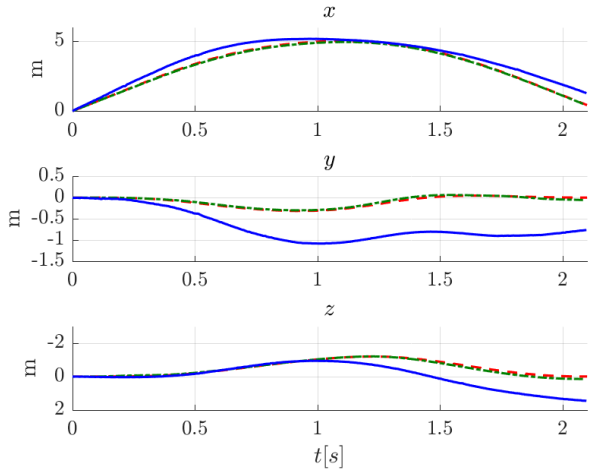
In simulations, the trajectory tracking performance is nearly ideal. Note in Fig. 6.7a, that the feedback controller is contributing to the full control action, since the simulated deflections are not perfectly aligned with the optimal feedforward inputs.

Observing the experimental results in Fig. 6.6a, we see that the attitude at the start and end of the maneuver is reasonably near the reference attitude. At approximately 0.5 s, the flight test attitude profiles begin to advance ahead of the optimization reference trajectory. This phenomenon is common among all flight tests and all maneuvers, and will be addressed shortly. Figure 6.6b shows room for improvement in position tracking. Such an improvement might be gained from tuning the position tracker gains for the maneuver specifically, and swapping into these gains when the maneuver begins. The forward speed, shown in Fig. 6.6c, is not tracked perfectly, but does start and end near the desired speed.

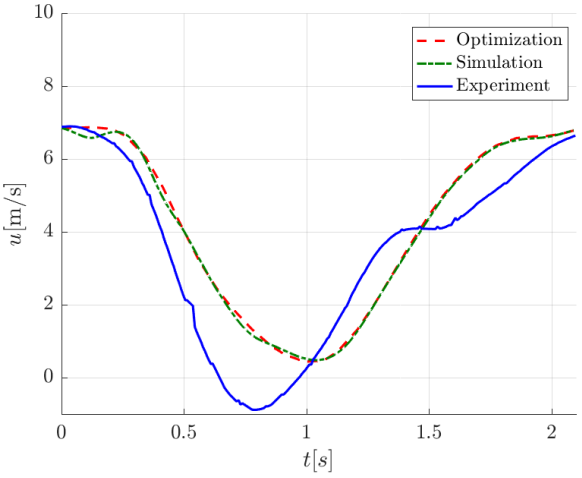
Examining the control surface deflections of the experiment shown in Fig. 6.7a, we note that they never deviate too far from the feedforward inputs, which would be the case if the tracking errors were large (and hence the feedback compensation was large as well). Figure 6.7b is where we see the most concerning results of the experiment. When initial testing began and the aircraft was first put in automatically controlled straight and level flight, it was immediately apparent that not enough thrust was being produced to fly at the desired velocity. The suspected reason for this is inaccuracies in drag modeling. At the time of modeling, the carbon fibre reinforcement rods (see Fig. 6.1b) were not



(a) Attitude

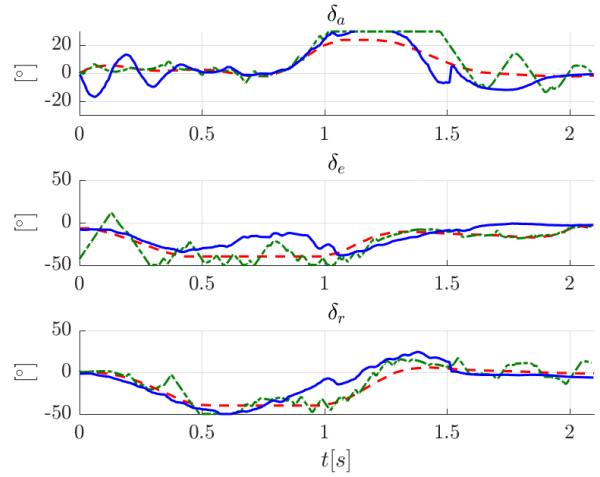


(b) Position

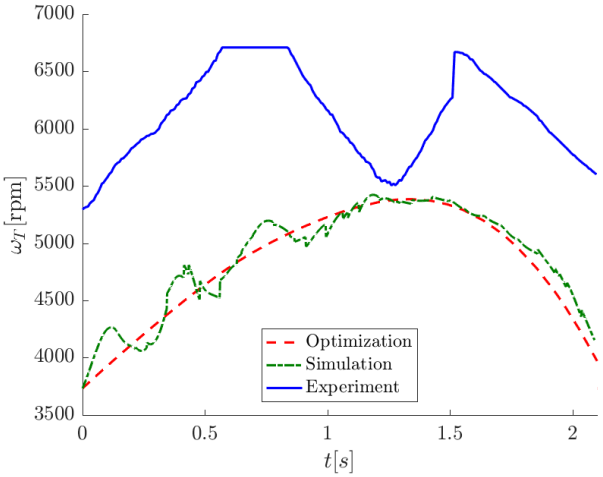


(c) Forward speed

FIGURE 6.6: ATA trajectories from optimization, simulation, and experiments, $V_0 = 7\,\mathrm{m\,s^{-1}}$.



(a) Control surface deflections



(b) Thrust

FIGURE 6.7: ATA control inputs from optimization, simulation, and experiments, $V_0 = 7\text{m}\,\text{s}^{-1}$.

considered, and thus there was no accounting for the drag they produce. To address this issue temporarily, a gain is introduced into the mapping of the commanded motor rotational speed, ω_T, to the actual thrust input to the aircraft, in PWM. The gain is tuned to a value of 1.1 by trial and error flights in the Stinger Dome. Where ω_T is plotted in Fig. 6.7b, the gain is removed to show the disparity between the feedforward and actual thrust input. This disparity presumably contributes to the phenomenon we see in the attitude curves in Fig. 6.6a. With a higher thrust input, the propeller is creating a greater slipstream effect, and thus the feedforward control surface deflections are more effective than as simulated. This hypothesis was confirmed using Simulink. Simulations of the ATA were conducted with a slight modification to reproduce this effect. With the rest of the model untouched, the slipstream velocities were multiplied by a gain, as though the

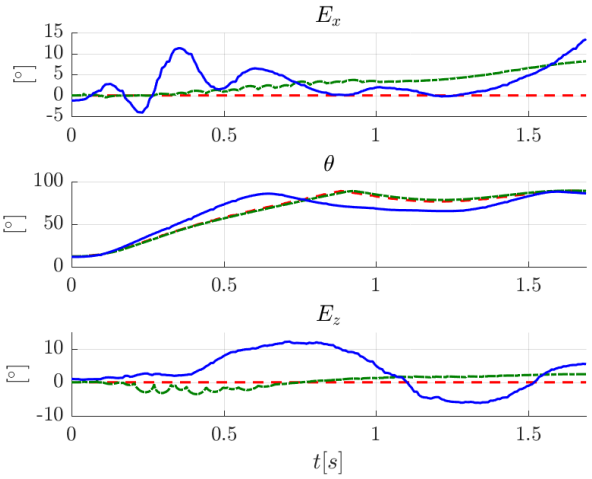
thrust input, ω_T, were as high as the flight test values in Fig. 6.7b. As predicted, the simulated attitude changes preceded the reference trajectory, just as in Fig. 6.6a.

We make some final observations about the experimental results, recognizing that the trajectory generation phase is pursued with specific objectives for the maneuver in mind - realized by the optimal control problem formulation. The rapid heading reversal is achieved as desired; the heading reaches 180 degrees by $t = t_f$. The optimal control problem is also defined such that the aircraft returns to straight and level flight at the initial position and with the initial velocity. We see from Figs. 6.6a–6.6c that the aircraft does in fact recover to straight and level flight, very near the initial velocity, but off from the desired final position. Given the importance of position tracking for following a motion plan, further gain tuning of the position tracker was required during that phase of implementation.

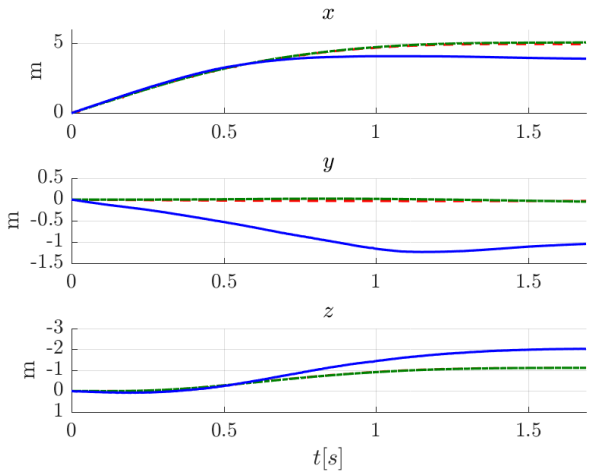
6.4.3 Cruise-to-Hover

Figure 6.9 displays the results from simulations and flight tests for the cruise-to-hover maneuver. The maneuver ends with a 90 degree pitch angle, which creates a singularity in the Euler angles. For this reason, the body-frame angular errors derived from the quaternion (see Section 2.2.2) are displayed for the body-frame x- and z-axes, instead of roll and yaw.

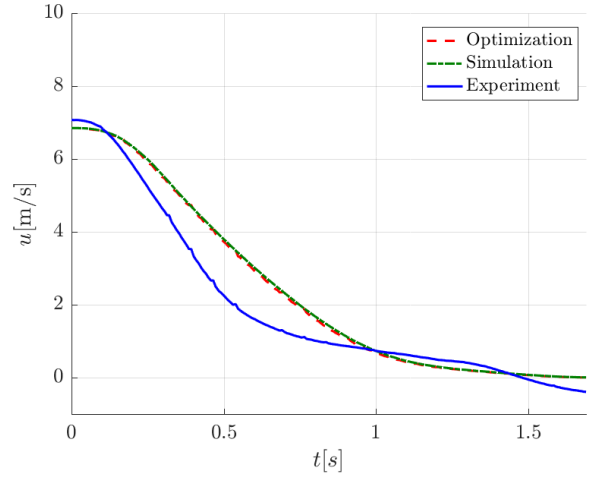
The pitch-up motion of the flight experiment precedes the trajectory it is meant to track (see Fig. 6.8a), as was observed during the ATA. At the end of the maneuver, the aircraft is pitched nose-up, as intended, with some errors in E_x and E_z. As shown in Fig. 6.8b, position tracking errors are present, however, just after 1 s, the aircraft maintains a near steady position. Note, in Fig. 6.8c, that the forward velocity is near zero by this point in time. We see in Fig. 6.9a that the control surfaces are normally far from being saturated, but the ailerons and rudder are consistently being actuated in attempt to keep the aircraft from rolling or yawing. This figure also reveals that less elevator deflection is needed to pitch the aircraft upwards than the modeling suggested, as observed between approximately 0 and 1.2 s. Shown again in Fig. 6.9b, more thrust is needed to overcome drag in experiments than in simulations.



(a) Attitude

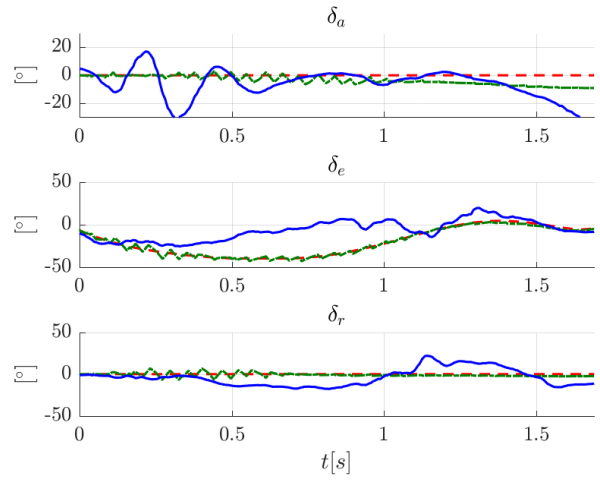


(b) Position

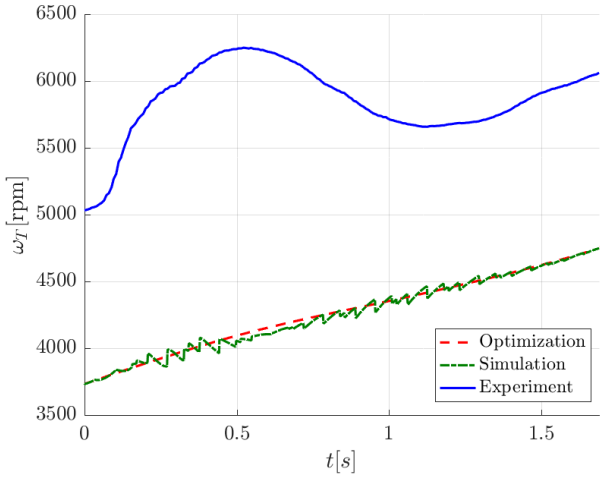


(c) Forward speed

FIGURE 6.8: CTH trajectories from optimization, simulation, and experiments, $V_0 = 7\,\mathrm{m\,s^{-1}}$.



(a) Control surface deflections



(b) Thrust

FIGURE 6.9: CTH control inputs from optimization, simulation, and experiments, $V_0 = 7\text{m}\,\text{s}^{-1}$.

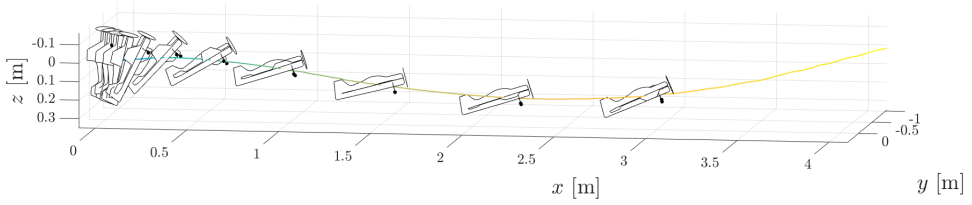


FIGURE 6.10: 3D visualization of HTC experiment trajectory, $V_f = 7\text{m}\,\text{s}^{-1}$. Aircraft drawings are scaled down and spaced out over a consistent time interval.

TABLE 6.1: Root-mean-square error for ATA, CTH, and HTC maneuvers in simulations ('Sim') and experiments ('Exp').

States	**Unit**	**ATA**		**CTH**		**HTC**	
		Sim	Exp	Sim	Exp	Sim	Exp
$\boldsymbol{u}$	$\mathrm{m\,s^{-1}}$	0.10	1.15	0.05	0.70	0.11	0.62
$\boldsymbol{\phi}/E_x$	°	4.57	24.2	3.80	4.67	11.1	13.9
$\boldsymbol{\theta}$	°	2.62	15.1	1.38	10.3	2.80	4.50
$\boldsymbol{\psi}/E_z$	°	2.44	25.3	1.72	6.43	4.50	6.34
$\boldsymbol{x}$	m	0.05	0.41	0.05	0.61	0.20	0.24
$\boldsymbol{y}$	m	0.02	0.67	0.03	0.87	0.29	0.36
$\boldsymbol{z}$	m	0.07	0.69	0.01	0.53	0.03	0.23

6.4.4 Hover-to-Cruise

In terms of the procedure, the experimental tests for the HTC maneuver always follow the CTH maneuver. In flight testing, the aircraft automatically executes the CTH maneuver, holds a stationary hover for a few seconds, and then executes the HTC maneuver. A 3D visualization of an HTC flight test is shown in Fig. 6.10. In the case plotted, $V_f = 7\mathrm{m\,s^{-1}}$. A quantitative assessment of the tracking performance exhibited in this maneuver and all others plotted (Figs. 6.6–6.10) is provided in Table 6.1. Table 6.1 lists the root-mean-square error (RMSE) values of the states being tracked with feedback. Note that for the hover transition maneuvers, E_x and E_z are listed in place of ϕ and ψ. As expected, the tracking performance is superior in simulations than in experiments. The gap in the performance metrics between simulations and experiments is smallest for the HTC maneuver, which has comparable RMSE values for most tracked states. In experiments, the attitude tracking was best during the CTH maneuver, while the position tracking was best during the HTC maneuver. Multiple flight test demonstrations of each maneuver are included in a supplementary video: https://www.youtube.com/watch?v=_hEF10_yiE4.

6.4.5 Knife-Edge

The flight tests for the knife-edge maneuver were not as successful as the others. Only after some heuristic tuning on the fly was the knife-edge able to be performed adequately, as seen in the video. To get the knife-edge working, the feedforward rudder input used to hold the steady maneuver - the one found in Section 4.4 - had to be lowered significantly. Another unexpected observation from the flight tests was that, after the tuning, the aircraft could transition directly from straight and level flight into a 90 degree roll, and

vice-versa, very quickly (in a fraction of a second). In practice, the transition maneuvers were redundant; they made no noticeable difference to the maneuver performance.

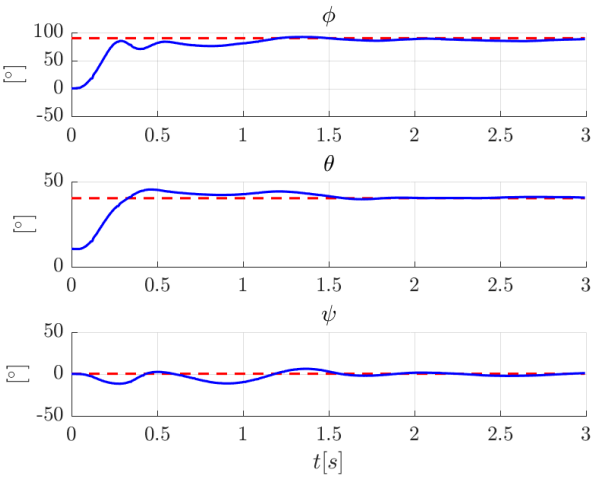
The knife-edge maneuver that was flown in tests is one optimized using $V = 7\ \mathrm{m\,s^{-1}}$. In Section 4.4, we used $V = 5\ \mathrm{m\,s^{-1}}$ because a slower speed seemed more appropriate for the small size of the flight test environment. However, once flight tests began we found the aircraft to generally be more stable at a slightly higher speed. At $V = 5\ \mathrm{m\,s^{-1}}$, the aircraft had to pitch up to a large angle to fly in trimmed straight and level flight.

Figures 6.11 and 6.12, show the optimization and experimental results. The transition to a 90 degree roll happens in approximately 0.25 s, and by approximately 1.5 s, the aircraft has stabilized about the desired attitude. In Fig. 6.11b, we see that the plane climbs approximately one meter during the knife edge, and begins to stray from the desired horizontal-plane path as the knife-edge is held. This highlights another reason why the knife-edge maneuver was not further pursued for motion planning. The purpose of flying a knife-edge would be to allow the aircraft to pass through passages more narrow than its wingspan. This utility is negated by the path-tracking errors that result from executing the maneuver. If the knife-edge maneuver cannot very closely track a desired path to thread the needle between two neighboring obstacles, it loses its utility for motion planning. Note that in Fig. 6.11c, the reference value for u is approximately $6\ \mathrm{m\,s^{-1}}$, rather than $7\ \mathrm{m\,s^{-1}}$, since u is the component of velocity aligned with the body frame x-axis, and doesn't represent the full magnitude of the velocity.

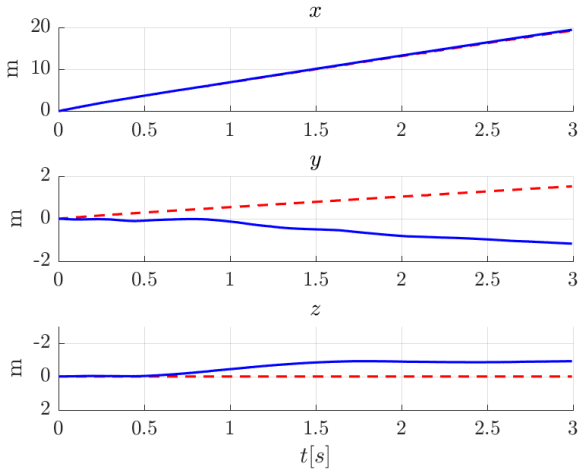
In Fig. 6.12a, we see that the aileron and elevator usage once knife-edge flight is steady is nearly as commanded by the optimization, but, as mentioned, the rudder input is far from the feedforward input of the optimization. The actual thrust used to hold the knife-edge is also between 500 and 1000 rpms more than the optimized value; these results all suggest model inaccuracies. As previously noted, one source of error is that the aircraft is experiencing more drag than was accounted for in the model during optimization, because at the time there was no modeling of the carbon fibre reinforcement rods. We acknowledge the possibility that there are other modeling inaccuracies present, which are more relevant during this maneuver than the other three.

6.5 Motion Planning Tests

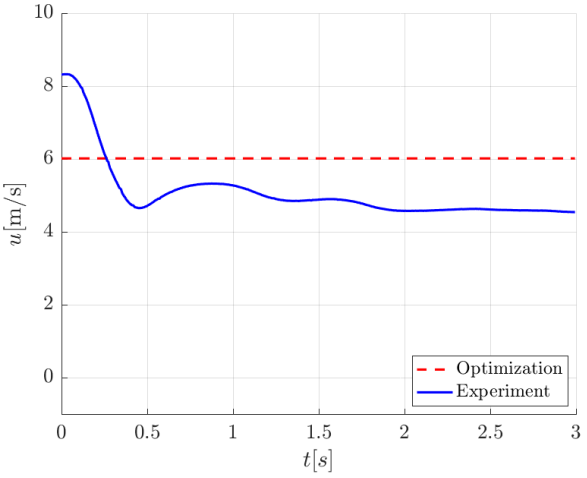
The second set of tests sought to validate the motion planner in concert with the control system. We look at simulation and flight test results separately, since flight test motion plans are generated on the fly and thus will be different from simulated plans.



(a) Attitude

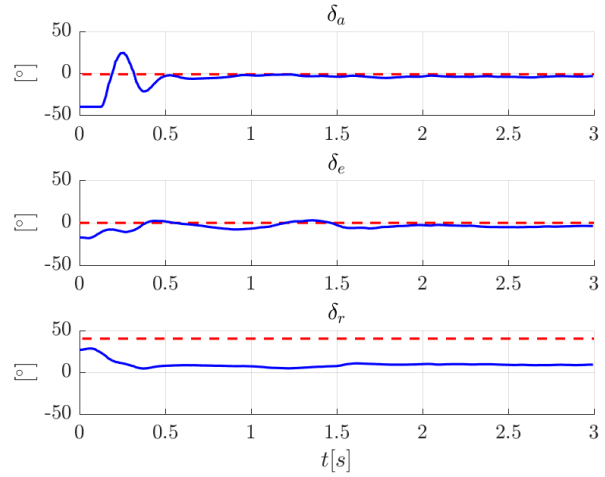


(b) Position

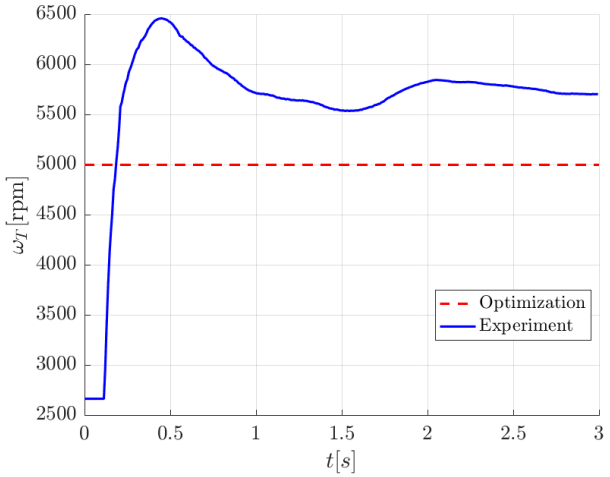


(c) Forward speed

FIGURE 6.11: Knife-edge trajectories from optimization and experiments, $V_0 = 7\,\mathrm{m\,s^{-1}}$.



(a) Control surface deflections



(b) Thrust

FIGURE 6.12: Knife-edge control inputs from optimization and experiments, $V_0 = 7\,\mathrm{m\,s^{-1}}$.

6.5.1 Simulations

Simulations were run to evaluate how well a generated motion plan would be tracked by the control system. A benefit of simulation testing is that, relative to the actual testing area, the simulation environment allows for flight on larger maps.

To establish the role that the planner plays in terms of the trajectory tracking problem, we also used simulations to track plans generated using the Dubins approach introduced in Section 5.3.6.1. Unlike our planner, the Dubins approach solves only for the reference path and heading, since it is based on the simple kinematic model of Eq. 5.8. Of particular note is that the Dubins model provides no means of calculating the feedforward control inputs. Accordingly, these terms were set to zero when tracking the Dubins curves (we

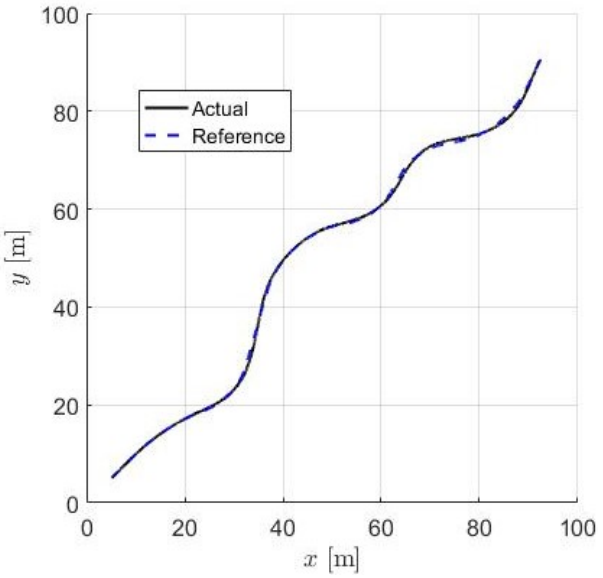


FIGURE 6.13: Simulated path tracking, obstacles have been removed for clarity.

TABLE 6.2: Root-mean-square errors and maximum errors for position tracking in simulations. Feedforward control inputs denoted by 'FF'.

Map	RMSE [m]	max(e_p)[m]
A (Maneuver Space)	0.22	0.86
B (Maneuver Space)	0.27	0.62
E (Maneuver Space)	0.38	0.71
C (Dubins)	5.04	12.37
D (Dubins)	–	–
F (Dubins)	3.81	13.68
C (Dubins + FF)	1.21	2.76
D (Dubins + FF)	1.58	2.91
F (Dubins + FF)	2.13	3.79

made the exception to include feedforward inputs during hover transitions). For good measure, we ran the set of simulations associated with the Dubins approach an extra time, including the feedforward inputs generated using the high-fidelity model. In effect, we thereby treat the Dubins paths as a small subset of our trim primitives: straight and level flight, a sharp banked turn to the right, and a sharp banked turn to the left.

We simulated all of Maps A-F in Figs. 5.11 and 5.12. As an example, the path tracking performance for the trajectory shown in Fig. 5.11b is plotted in Fig. 6.13. The RMSE and maximum error on position for each simulation are listed in Table 6.2. Maps A, B, and E use the maneuver space, while Maps C, D, and F use Dubins curves – note the large discrepancies in tracking performance between the two approaches. In the case of Map D, the aircraft essentially failed to track the trajectory; the position errors grew

so large we opted not to list them. Also shown in this table are the results of tracking the Dubins curves whilst incorporating the feedforward inputs generated using the high-fidelity model. Even after integrating this aspect of our motion primitives into the Dubins approach, the tracking performance remained inferior. With respect to the RMSE values, there was still a difference in position error by a factor of approximately five between the two methods. These results can be attributed to the lack of transition modeling between the straight segments and highly aggressive turns.

With respect to the maneuver space approach, the position errors, along with the 0.86 m wingspan of the aircraft, can be used to inform the buffer distance parameter found in the collision checker. Given these values, it would be reasonable to set the buffer distance to at least 1.5 m to ensure safe, collision-free flight. The tracking performance and buffer size must be interpreted with respect to the environment the aircraft is tasked with passing through. As long as there is sufficient room left after the obstacles are buffered for the tree to efficiently expand through the map, as has been the case here, the 1.5 m distance is acceptable.

6.5.2 Flight Test Implementation Details

Once again, flight tests are performed in the Concordia Stinger Dome. The flights were mainly limited to one quarter of the dome, a 30 m by 60 m field. For a limited number of tests, we were able to utilize half of the dome, a 60 m by 60 m field. Overall, the dome was a convenient setting for performing flight tests, however, one disadvantage was that there was no practical way of installing extra obstacles in the environment. The dome itself is too empty to show off the capabilities of the planner and control system. Obstacles that we would have liked to install in the dome were programmed into the motion planner code on the ODROID as inertial coordinates. As such, the only thing lacking due to this restriction is the visual presence of the obstacles in videos. The supplementary videos, however, have been post-processed to include obstacle shapes edited into the environment as they would exist according to the coordinates programmed in the motion planner.

The experimental procedure for testing the motion planner and control system is as follows. Using the Futaba T7C RC transmitter, the trained pilot manually takes off and flies the aircraft into a hover. He then flicks a switch on the transmitter to put the aircraft into an autonomously controlled hover (the hover trim primitive). Next, he flicks another switch to trigger the motion planner to begin. It is at this time that the Pixhawk Mini sends the aircraft's current measured pose (position and heading) to the ODROID as the initial condition to the motion planning algorithm.

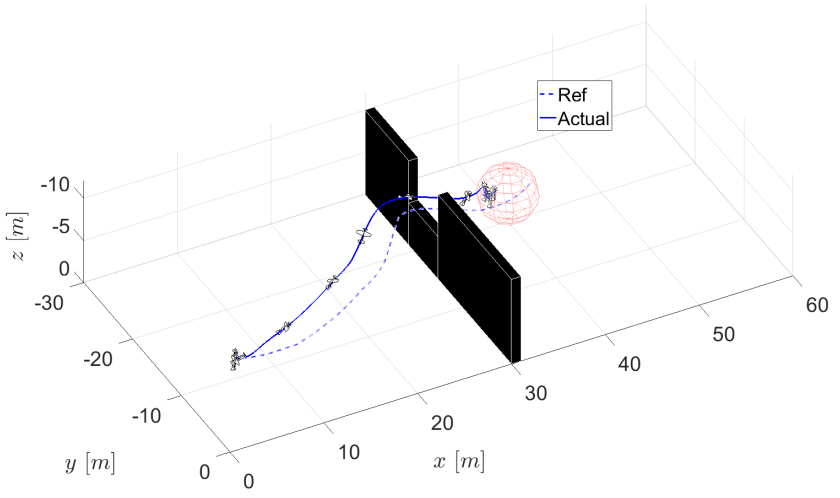
As a practical measure, the planner was programmed to delay the first iteration of the Update Tree function until the tree reached the goal region. While not strictly necessary, doing so added an extra measure of safety by ensuring that at least *some* feasible path to the goal region existed before taking off from the hover. This step also allowed us to demonstrate in video how efficient the planner was at finding a feasible path. The delay never lasted more than a few seconds, and often less than one. The planner continues to run while in flight, and as per the *Update Tree* function, may end up on a more direct path towards the goal. The step of delaying the first iteration of the Update Tree function can certainly be discarded for applications in which an immediate take-off is preferable.

The dome is only open for a few months of the year, and our access to it during those months is limited. For this reason, we unfortunately did not have the time to validate every part of the motion planner. Specifically, we did not get to implementing and testing the re-planning algorithm. In Section 6.1.3, we included it in simulations of the full motion planning algorithm to show that, in terms of efficiency, the re-planning step is feasible. The flight tests that follow in this section, however, omit the re-planning step from the algorithm, and thus we cannot yet quantify the effect it has on tracking performance.

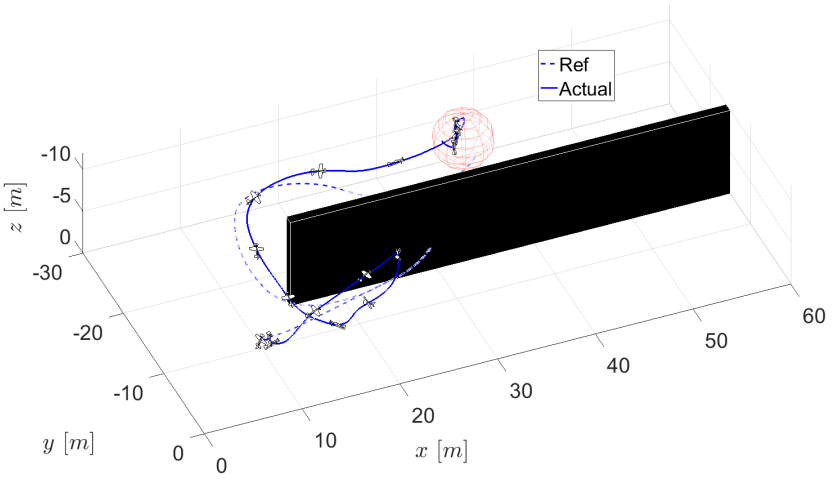
6.5.3 Flight Tests

With flight tests, we aimed to discover how efficiently and effectively the motion planning algorithm could generate a plan to guide the aircraft to the goal region, and how well that plan would be tracked. Three maps of virtual obstacles were programmed onto the ODROID. The available sections of the dome are relatively small and thus the flights are short; nonetheless, they showcase many features of the motion planner.

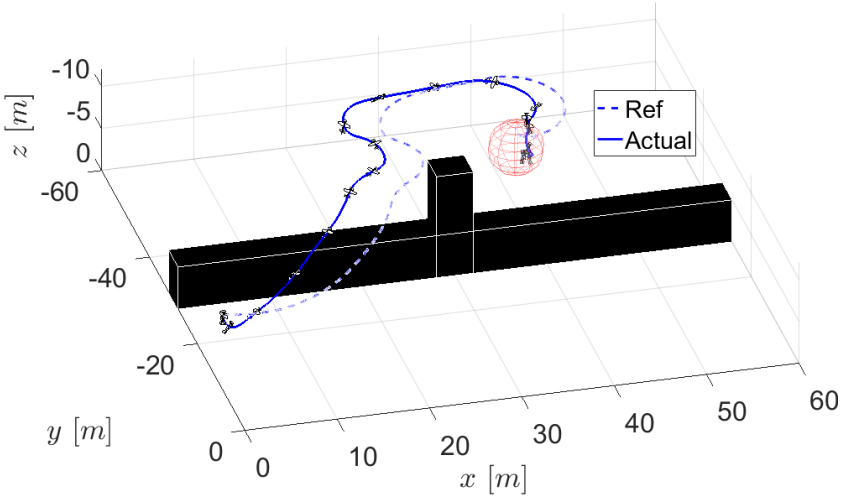
Figure 6.14 shows trajectories and path tracking results for the three new maps, where the plot edges match the maps' boundaries. The maps are different than those of Section 5.3.6 because the space in the dome is limited to a smaller area. The plots show the reference paths and the actual paths flown, and the red spheres represent the goal regions. In the first map, the aircraft must climb to and navigate through a narrow gap. In the second map, the aircraft begins with a heading that is pointed towards a dead-end, and thus must turn around to proceed in the proper direction. It executes an aggressive turn-around maneuver here to do so. The third map uses a larger portion of the dome, a 60 m by 60 m field. The obstacles in this plot overlay actual obstacles in the environment - wires and meshing that separate quarters of the field and thus exist in this larger portion of the dome. The values of RMSE and max error in position for the three maps are



(a) Map 1



(b) Map 2



(c) Map 3

FIGURE 6.14: Motion planning flight test results.

TABLE 6.3: Root-mean-square errors and maximum errors for position tracking in flight tests.

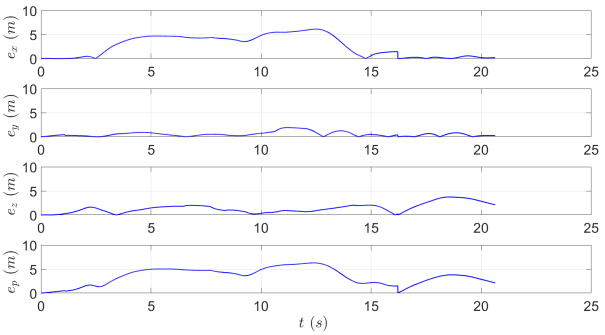
	Map 1	**Map 2**	**Map 3**
RMSE [m]	2.82	3.03	3.88
max(e_p)[m]	4.49	4.56	6.30

given in Table 6.3. A supplementary video includes the flights through Maps 1 and 2: https://www.youtube.com/watch?v=63g5NSn-odU.

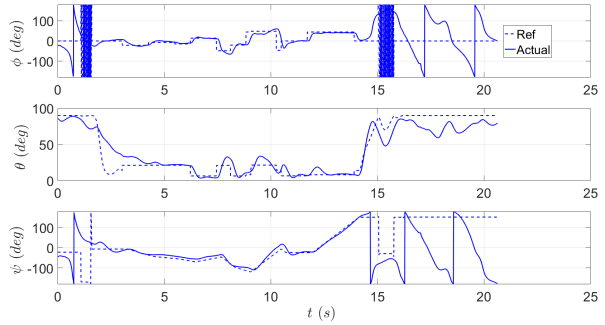
Figure 6.15 shows time histories of the state variables and control inputs for the flight test associated with Map 3 in Fig. 6.14c. We take a closer look at the results of this map because it has the longest trajectory of the three, and the discussion of its results largely extends to the findings of the other maps. The position errors are shown in Fig. 6.15a, and note that the x, y, and z axes are aligned with the map, as in Fig. 6.14c. We see that around 2 s the position errors start to grow as the hover-to-cruise maneuver is occurring. Thereafter, the errors more or less plateau and only diminish around when the cruise-to-hover maneuver takes place. The reason the errors plateau instead of diminish is attributed to the fact that the aircraft is continually being destabilized by switching motion primitives. As demonstrated in Section 4.3, the transition maneuver heuristic helps deal with this tracking problem, however, it does not eliminate it. The change from one primitive to the next can most clearly be seen in Fig. 6.15b, where every step input change in the reference roll angle implies that a new primitive is being commanded.

The values of RMSE and max error in position for all three maps are given in Table 6.3. The position errors are larger than those found in Section 6.5.1, for simulations, because of the presence of measurement noise, imperfect state estimation, and modeling inaccuracies in the experimental setup. The errors could presumably be resolved in a number of different ways in future work. One option would be to implement gain scheduling, so that the optimal gains for the hover-to-cruise maneuver could be used and thus the initial errors would be reduced. Position errors could also be addressed on the side of the motion planner, via the re-planning algorithm, which would be triggered when position tracking errors become sufficiently large.

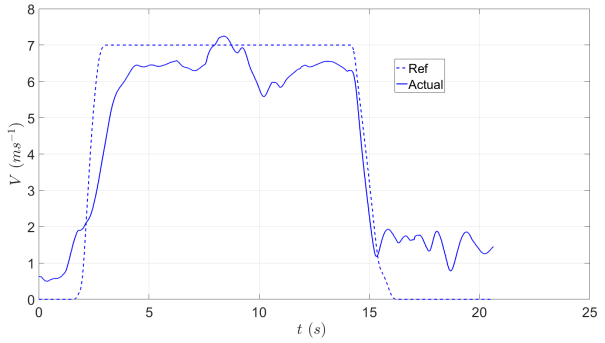
Figure 6.15b plots the attitude as Euler angles. Note that the aircraft is in a hover at the beginning and end of the plan. This causes a singularity in the Euler angle attitude representation, which is why the roll and yaw values are spiking back and forth. The reference yaw angle time history is continuous, but the roll and pitch values change in steps because they are associated with the primitives. This highlights the importance of the transition maneuvers, which allow the aircraft time to reach - and ideally settle - at



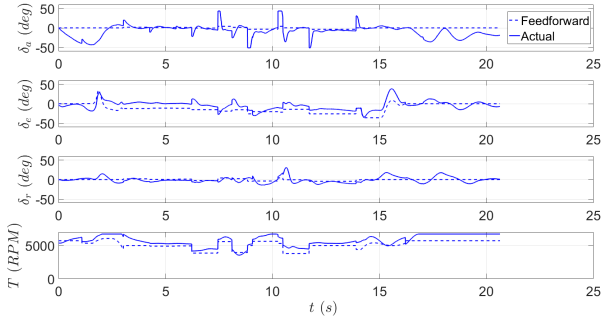
(a) Position errors



(b) Attitude tracking



(c) Speed tracking



(d) Control inputs

FIGURE 6.15: Flight test states and control inputs for Map 3 of Fig. 6.14c.

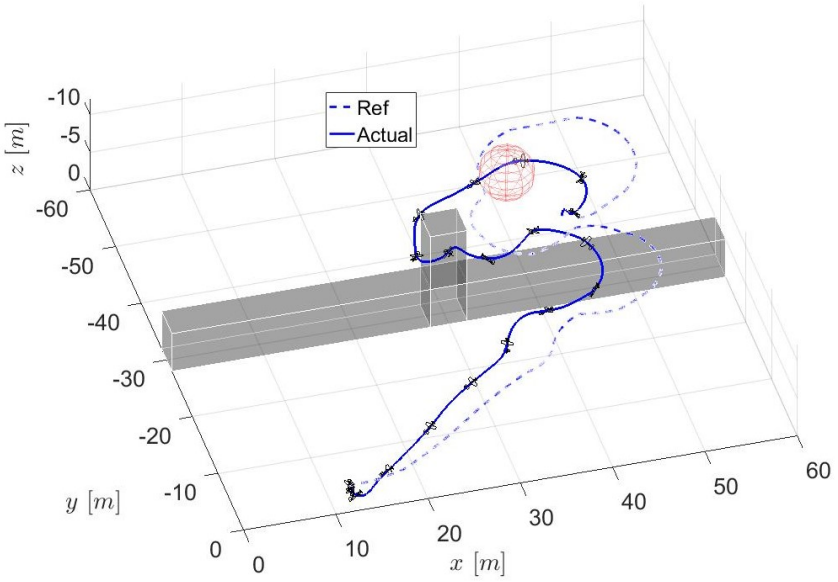


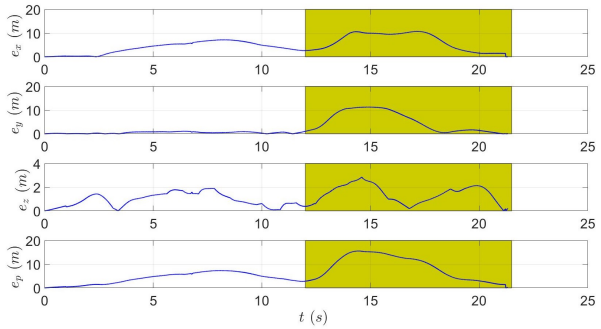
FIGURE 6.16: Flight test involving crash with wire.

the state of the new primitive. The attitude tracking is at its worst at the beginning of the plan. At approximately two seconds in, the hover-to-cruise maneuver is initiated. We note that the pitch profile is not tracked as accurately during this maneuver as it is for the remainder of the plan.

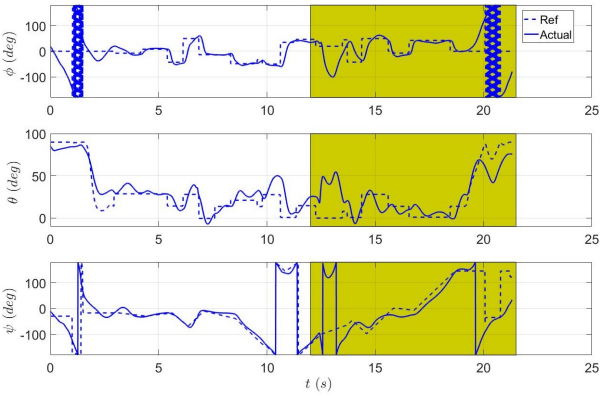
In Fig. 6.15c, the speed throughout the plan is plotted. Throughout the middle portion of the plan, the aircraft is able to stay near the desired constant speed of $V = 7\mathrm{m\,s^{-1}}$. The most challenging sections for the speed tracking portion of the control system are the hovers. The aircraft is inherently unstable in this configuration, and often has to use non-zero velocities to maintain a commanded position.

The feedforward and full control inputs are plotted in Fig. 6.15d. The difference between the feedforward and actual inputs is the feedback control. By comparing the solid and dashed lines, we can see from these plots that both feedforward and feedback inputs are valuable. The elevator, rudder, and thrust control are largely guided by the feedforward inputs, i.e. the actual control is close to the feedforward control. The ailerons, however, are using a large amount of feedback control. The aileron response is proportional to the large step input changes in roll that are being commanded.

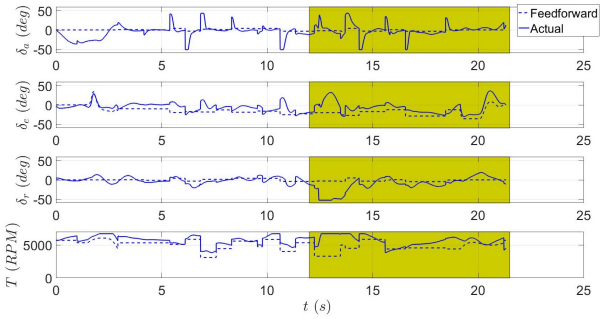
A final set of flight test results is presented that demonstrates the outcome of a flight that involved a mid-air collision. As previously mentioned, when using half of the dome (as opposed to a quarter) there are actual obstacles that run down the middle of the environment. There is a mesh divider that spans the length of the field and rises a few meters from the ground, and there is a wire that attaches to this divider and holds it up



(a) Position errors



(b) Attitude tracking



(c) Control inputs

FIGURE 6.17: Position errors and control inputs for flight test involving crash with wire.

from the ceiling. In Fig. 6.16, these obstacles are plotted as inflated rectangular prisms. This figure shows that the motion plan's reference trajectory avoids the obstacles, but due to position errors, the actual path of the aircraft ends up colliding with the wire. The crash was not insignificant; the wing was struck and this spun the aircraft around, reducing its speed and pointing its propeller in a very different direction. Interestingly, rather than plummeting to the ground, the aircraft recovers and then continues to track the motion plan until it ends in a hover. Figure 6.17 plots the position errors and control inputs of the flight. The yellow background in these plots begins when the crash took place. In Fig. 6.17a, we see that the position errors grow quite large, over 15 m, and

yet the control system is able to keep the aircraft in the air, and continues to greatly reduce the position errors over the next few seconds. The attitude tracking, Fig. 6.17b, also takes a large hit at the time of the collision, but is nearly right back on track after approximately 2 seconds. In Fig. 6.17c we can observe that the difference between the feedforward and actual control inputs is large at the time of the crash. The feedforward inputs cannot react to disturbances from the nominal trajectory and thus the feedback control inputs become dominant. After a couple seconds, as the tracking errors are diminishing, the differences between the feedforward and actual controls reduce as well. Although the control system should have kept the aircraft closer to the desired path and avoided the collision altogether, it is valuable to know that the control system was able to recover from such major deviations from the reference trajectory.

Chapter 7

Conclusion

To conclude the thesis, we summarize the contributions made in the order they were originally presented, and make suggestions for future work.

7.1 Summary of Contributions

An investigation into the role of sideslip and propeller slipstream in the extreme maneuvering capabilities of agile fixed-wing UAVs was performed. The cost, in terms of performance loss, was identified if either of the two phenomena is not accounted for in maneuver design. It was concluded that ignoring these effects in the model used for trajectory generation will lead to sub-optimal results. This is especially true in the case of the slipstream effect, where differences in the space and time required to perform extreme maneuvers was found to be multiple orders of magnitude different depending on whether the effect was modeled or not. This highlights the importance of model fidelity in trajectory generation for this class of agile fixed-wing UAV.

Next, a general and systematic approach for generating and automating a variety of steady and transient agile maneuvers was developed. Trajectories were generated by solving optimal control problems. The framework for the problems involved an accurate physics-based model, and therefore the trajectory solutions are dynamically feasible and fully exploit the aircraft's physical capabilities. To track the trajectory, feedforward control inputs - found as part of the optimization solution - are paired with a general and largely physics-based feedback controller.

The trajectories generated – trim and agile maneuver primitives - compose a maneuver space that can be used for motion planning. The maneuver space acts as a hybrid

representation of the aircrafts dynamics, such that the planning algorithm can generate dynamically feasible trajectories in real-time without have to solve complex dynamic constraints. The maneuver space was made large enough to represent a significant portion of the aircraft's flight envelope, while being suitable for the available computational resources. The maneuver space concept borrows from [45], however, the development of the maneuver space itself and its particular integration within an RRT-based planner are original contributions.

To complement the maneuver space, a novel method for trajectory parametrization, making use of dynamic time warping, was proposed. The parametrization makes the agile maneuver space more robust while incurring a negligible cost to the computational load needed in flight. The parametrization method is strategically split into off-line and on-line portions, such that the storage requirements and computational burden on the flight controller are minimal.

The maneuver space was integrated into an RRT-based real-time motion planner in such a way that the planner is able to exploit the extent of the aircraft's flight envelope and account for the full dynamic constraints of the aircraft while retaining computational efficiency on-line. The agile maneuvers are used in the planner following a logic of achieving functional goals that allow the aircraft to stop and start in a hover, and turn around in minimal space as needed. Altogether, the motion primitives of the maneuver space enable the aircraft to navigate through obstacle-dense, three-dimensional environments.

In simulations, the planner was evaluated against a baseline approach that uses Dubins curves. The most significant finding of the comparison was that our proposed planning approach generated smoother trajectories that brought about superior tracking performance. Position errors resulting from tracking plans generated using Dubins curves were more than an order of magnitude greater than when tracking plans generated using the maneuver space.

Finally, flight test experiments with an agile fixed-wing UAV were performed to shed insight into practical implementation issues. A first set of flight tests was used to validate the execution of four agile maneuvers: a cruise-to-hover transition, a hover-to-cruise transition, an aggressive turn-around, and a knife-edge maneuver. The second and final set of flight tests evaluated the real-time motion planner. To the best of our knowledge, these flight tests were the first to demonstrate their level of implementation in terms of the extent of the flight envelope utilized. Furthermore, all testing fully relied on on-board sensing and computing, which had yet to be demonstrated for real-time motion planning with a fixed-wing UAV.

7.2 Recommendations for Future Work

In pursuit of the objectives of this thesis, many possible avenues for future research were opened:

- It was revealed during flight tests that the weight and weight distribution of the aircraft have a significant effect on tracking performance. Mounting both the Pixhawk and ODROID on the aircraft increased its weight significantly, and the size and layout of the airplane made it difficult to keep all the peripheral equipment near the aircraft's center of gravity. In light of this, we see value in implementing all computations on one capable single-board computer.
- While the feedback controller generally performed well given only one set of gains, tracking performance could presumably be improved by using gain scheduling (and/or gain optimization) for different maneuvers. It would also be valuable to perform a comparison of tracking performance with other feedback controllers, such as time-varying linear quadratic regulators [124].
- The collision check function in the motion planning algorithm could be replaced with one that can handle more complex geometries. More capable algorithms exist, such as the Bullet Physics SDK [125], but one would have to ensure that the motion planner's efficiency is not compromised; the collision check function represents a costly portion of the full algorithm.
- The motion planner currently has two methods of dealing with disturbances and uncertainties: buffers are placed around obstacles, and re-planning can be performed when position errors grow too large. Nevertheless, it would be worthwhile to investigate more sophisticated ways of dealing with these issues; approaches that can provide guarantees such that the motion planner does not generate trajectories that may result in collisions. One such approach is to compose the maneuver space of closed-loop reachable sets, i.e. trajectories with guaranteed robustness properties [73].
- The motion planning framework is based on two assumptions, the first of which is that the environment is known in advance. In this regard, the planner could be enhanced towards a broader set of real-world applications, i.e. for partially known environments, by augmenting it with an obstacle detection and avoidance system. In this case, the RRT-based planner would create a 'global' plan based on the known environment. Then, when a new obstacle is sensed in real-time, the detection and avoidance system would temporarily take over to evade the obstacle and return to the global plan. One possible approach to obstacle avoidance would be to employ Potential Field methods [126].

- The second assumption of the motion planner is that obstacles in the environment are stationary. This could also be addressed within the planner's framework. If obstacles are in motion and their dynamics are known, the problem becomes only slightly more complex. The algorithm would have to forward simulate the obstacles' dynamics to know their position and orientation with respect to time before checking for collisions. The more difficult problem of avoiding obstacles with uncertain motion would involve detecting them in real-time, predicting their dynamics, and accounting for the uncertainty of the prediction in the avoidance system [127].
- The re-planning algorithm of Section 5.3.5.1 should be tested in experiments in order to determine how effectively it can reduce tracking errors.
- A strategy should be devised to deal with the switching problem observed in Section 6.5.3. The transition maneuver heuristic could potentially be improved by making the time-delay value a function of the trim primitives it is switching between, rather than a constant.
- Lastly, a system for taking off and landing could be investigated as a means of enhancing the autonomous capabilities of the UAV. If the aircraft could take-off from a stationary state more stable than a hover, the overall tracking performance during motion planning might be improved. Ground take-off and landing maneuvers could be investigated, but the aircraft is not currently equipped with landing gear because it would increase weight and drag. An alternative option would be to create a launching mechanism, like in [49, 128]; the aircraft could then land by hovering over a safety net and reducing or shutting off its thrust.

Appendix A

Feedback Gain Optimization

This appendix describes a methodology that was explored but not further pursued because it did not directly contribute to the objectives of the thesis. The idea was to include feedback control laws in the optimal control problem used for trajectory generation, and the reasoning for doing so was two-fold. First, the gains of the feedback controller could be optimized as part of the solution to the problem. Additionally, we could then evaluate the feedback controller's efficacy for executing the maneuver in question, by comparing the trajectory solutions that result from framing the problem with and without the inclusion of the control laws.

A.1 Optimal Control Framework with Feedback Control Laws

The maneuver that was considered during this study was one very similar to the aggressive turn-around of Section 4.2.1. One difference was the cost function,

$$\min \; J \triangleq \int_0^{t_f} (\eta_x x^2 + \eta_y y^2 + \eta_z z^2) dt, \tag{A.1}$$

which penalized displacements in space. The other difference was that there were no boundary conditions on the final position of the maneuver. The weights in the cost function, η_x, η_y, and η_z, were tuned to shape the path of the turn-around; the values used for the study were set to $[\eta_x, \eta_y, \eta_z] = [1, 1, 5]$ $\mathrm{m}^{-2}\,\mathrm{s}^{-1}$.

The purpose of the endeavor was not to delve into the details of feedback controller design, but rather to present a framework for optimizing gains and evaluating the performance of control laws. For this purpose we chose to consider a combination of four single-input single-output (SISO) proportional-integral (PI) controllers: one channel controlling the elevator as a function of angle-of-attack; one channel controlling the aileron as a function of roll angle; one channel controlling the rudder as a function of sideslip angle; and a last channel controlling thrust as a function of speed. The feedback terms are one-to-one mappings between (a) α and δ_e; (b) ϕ and δ_a; (c) β and δ_r; and (d) V and T. The control inputs, $\delta_{e,a,r}$ and T, thus become state variables. The elevator deflection, for instance, is found using:

$$\delta_e(t) = k_{p\alpha}(t)(\alpha_c(t) - \alpha(t)) + k_{I\alpha}(t) \int_0^t (\alpha_c(t) - \alpha(t))dt + \delta_{e_{trim}} \tag{A.2}$$

where $\alpha_c(t)$ is the commanded angle-of-attack, and $\delta_{e_{trim}}$ is a constant level flight trim value. The gains k_p and k_I are also set as state variables in the optimal control problem, governed by the following dynamics:

$$\begin{aligned} \dot{k}_p &= \tilde{u}_p \\ \dot{k}_I &= \tilde{u}_I \end{aligned} \tag{A.3}$$

The optimization solves for $\alpha_c(t)$, $\tilde{u}_p(t)$, and $\tilde{u}_I(t)$ as the control inputs. The same procedure is followed for δ_a, δ_r, and T.

Two methods of optimally tuning the PI gains were investigated. In the first method, the gains k_p and k_I were optimized but constrained to be constant values by equating $\tilde{u}_p$ and $\tilde{u}_I$ to zero. In the other method, they were allowed to vary with time. The reason for conducting two methods of gain tuning was to compare how well a very simple, constant gain PI controller would perform in contrast to one with varying gains. We were also, at the time, interested in investigating whether gain scheduling could be a viable strategy for performing turn-around maneuvers.

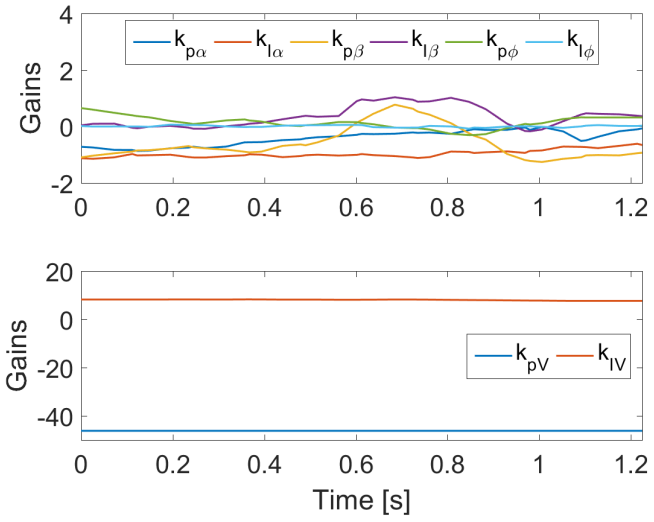


FIGURE A.1: Optimized proportional k_p and integral k_I gains for varying gain controller.

TABLE A.1: Optimized proportional k_p and integral k_I gains for constant gain controller.

k_{p_α}	$k_{I_\alpha}[\mathrm{s}^{-1}]$	k_{p_β}	$k_{I_\beta}[\mathrm{s}^{-1}]$	k_{p_ϕ}	$k_{I_\phi}[\mathrm{s}^{-1}]$	k_{p_V}	$k_{I_V}[\mathrm{s}^{-1}]$
-0.64	-0.84	-0.83	0.94	0.36	0.01	-58	7.6

TABLE A.2: Cost function values.

No Controller	Varying Gains	Constant Gains
7.912	8.203	8.203

A.1.1 Results and Discussion

The optimal control problem was first solved without the control laws, and then for each of the two SISO control policies: constant gains and varying gains. Figure A.1 shows the optimized evolution of the gains when allowed to vary, and Table A.1 lists the optimized constant gains.

In Fig. A.2, the paths are plotted for the three trajectory solutions. Adding control laws to the aircraft dynamics is equivalent to further constraining the optimization problem. The cost function values, Table A.2, act as a performance metric and confirm that the control laws slightly diminish performance. With that said, however, performance is practically identical between the two cases of control policies. This information aligns with the fact that the optimally varying gains, Fig. A.1, remain practically constant, and nearly equivalent to the gains found in Table A.1. While we were initially interested

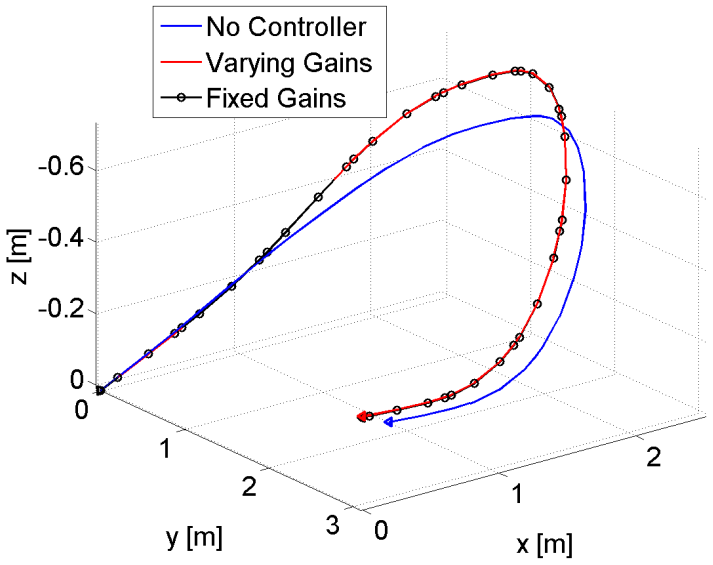


FIGURE A.2: Paths resulting from different controller setups.

in investigating the merits of gain scheduling throughout a maneuver, these results show that this strategy may be superfluous, at least for the maneuver in question.

We conclude this appendix with a few general remarks about the proposed methodology. In this approach, gains are optimized based on the constraints imposed by the control law, which is not how gains are typically tuned or optimized. Feedback gains are normally tuned to make up for uncertainties and appropriately respond to disturbances. With that said, the methodology proposed here could potentially be found useful as a first step in the gain tuning/optimization process. We also note that this method of gain optimization is only applicable if the control laws do in fact put constraints on the control inputs. For instance, if in Eq. A.2, the constant elevator trim value, $\delta_{e_{trim}}$, were instead a free feedforward input variable, the control laws would be imposing no constraints. With all gains set to zero, the resulting trajectory would be equivalent to the one found without the inclusion of the feedback controller. This was precisely the case of the approach taken in this thesis; feedforward control inputs were only constrained by their saturation values, and thus nothing could be gained by including feedback control laws in the optimal control problem.

Bibliography

[1] Joshua M Levin, Meyer Nahon, and Aditya A Paranjape. Aggressive turn-around manoeuvres with an agile fixed-wing UAV. In *20th IFAC Symposium on Automatic Control in Aerospace*, pages 242–247. Elsevier, 2016. doi: 10.1016/j.ifacol.2016.09.042.

[2] Joshua M Levin, Aditya Paranjape, and Meyer Nahon. Agile fixed-wing UAV motion planning with knife-edge maneuvers. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 114–123. IEEE, 2017. doi: 10.1109/icuas.2017.7991475.

[3] Joshua M Levin, Aditya A Paranjape, and Meyer Nahon. Sideslip and slipstream in extreme maneuvering with fixed-wing unmanned aerial vehicles. *Journal of Guidance, Control, and Dynamics*, 41(7):1610–1616, May 2018. doi: 10.2514/1.G003086.

[4] Joshua M Levin, Aditya A Paranjape, and Meyer Nahon. Agile maneuvering with a small fixed-wing unmanned aerial vehicle. *Robotics and Autonomous Systems*, 116:148–161, March 2019. doi: 10.1016/j.robot.2019.03.004.

[5] Joshua M Levin, Aditya A Paranjape, and Meyer Nahon. Motion planning for a small aerobatic fixed-wing unmanned aerial vehicle. In *The International Conference on Intelligent Robots and Systems (IROS)*, pages 8464–8470. IEEE, 2018. doi: 10.1109/IROS.2018.8593670.

[6] Joshua M Levin, Meyer Nahon, and Aditya A Paranjape. Real-time motion planning with a fixed-wing UAV using an agile maneuver space. In review with Autonomous Robots.

[7] Waqas Khan and Meyer Nahon. Development and validation of a propeller slipstream model for unmanned aerial vehicles. *Journal of Aircraft*, 52(6):1985–1994, April 2015. doi: 10.2514/1.C033118.

[8] Meinard Muller. *Information Retrieval for Music and Motion*. Springer-Verlag, New York, 2007.

[9] Darryl Jenkins and Bijan Vasigh. The economic impact of unmanned aircraft systems integration in the United States. *Association for Unmanned Vehicle Systems International*, 2013.

[10] Aditya Paranjape, Soon-Jo Chung, and Michael S Selig. Flight mechanics of a tailless articulated wing aircraft. *Bioinspiration & Biomimetics*, 6(2):1–15, April 2011. doi: 10.1088/1748-3182/6/2/026005.

[11] Bret Stanford, Mujahid Abdulrahim, Rick Lind, and Peter Ifju. Investigation of membrane actuation for roll control of a micro air vehicle. *Journal of Aircraft*, 44 (3):741–749, June 2007. doi: 10.2514/1.25356.

[12] Analiza Abdilla, Arthur Richards, and Stephen Burrow. Endurance optimisation of battery-powered rotorcraft. In *Conference Towards Autonomous Robotic Systems*, pages 1–12. Springer, Cham, 2015. doi: 10.1007/978-3-319-22416-9_1.

[13] J Verbeke, D Hulens, H Ramon, T Goedeme, and J De Schutter. The design and construction of a high endurance hexacopter suited for narrow corridors. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 543–551. IEEE, 2014. doi: 10.1109/ICUAS.2014.6842296.

[14] William E Green and Paul Y Oh. Autonomous hovering of a fixed-wing micro air vehicle. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2164–2169. IEEE, 2006. doi: 10.1109/robot.2006.1642024.

[15] Rick Cory and Russ Tedrake. Experiments in fixed-wing UAV perching. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 1–12. AIAA, 2008. doi: 10.2514/6.2008-7256.

[16] Adam M Wickenheiser and Ephrahim Garcia. Longitudinal dynamics of a perching aircraft. *Journal of Aircraft*, 43(5):1386–1392, September-October 2006. doi: 10.2514/1.20197.

[17] Wolfgang B Herbst. Dynamics of air combat. *Journal of Aircraft*, 20(7):594–598, 1983. doi: 10.2514/3.44916.

[18] William E Green and Paul Y Oh. A MAV that flies like an airplane and hovers like a helicopter. In *International Conference on Advanced Intelligent Mechatronics*, pages 693–698. IEEE, 2005. doi: 10.1109/AIM.2005.1511063.

[19] Aditya A Paranjape, Animesh Chakravarthy, Soon-Jo Chung, and Harry H Hilton. Performance and stability of an agile tail-less MAV with flexible articulated wings. In *AIAA Atmospheric Flight Mechanic Conference*, pages 1–13. AIAA, 2010. doi: 10.2514/6.2010-7937.

[20] Baron Johnson and Richard Lind. High angle-of-attack flight dynamics of small UAVs. In *AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, page 61, 2009. doi: 10.2514/6.2009-61.

[21] Huaiyu Wu, Dong Sun, and Zhaoying Zhou. Model identification of a micro air vehicle in loitering flight based on attitude performance evaluation. *IEEE Transactions on Robotics*, 20(4):702–712, 2004. doi: 10.1109/tro.2004.829442.

[22] David J Grymin and Mazen Farhood. Two-step system identification and trajectory tracking control of a small fixed-wing UAV. *Journal of Intelligent & Robotic Systems*, 83(1):105–131, July 2016. doi: 10.1007/s10846-015-0298-8.

[23] William E Green and Paul Y Oh. A hybrid MAV for ingress and egress of urban environments. *IEEE Transactions on Robotics*, 2(2):253–263, 2009. doi: 10.1109/tro.2009.2014501.

[24] Adrian Frank, James S McGrew, Mario Valenti, Daniel Levine, and Jonathan P How. Hover, transition, and level flight control design for a single-propeller indoor airplane. In *Guidance, Navigation, and Control Conference and Exhibit*. AIAA, 2007. doi: 10.2514/6.2007-6318.

[25] Eric N Johnson, Allen D Wu, James C Neidhoefer, Suresh K Kannan, and Michael A Turbe. Flight-test results of autonomous airplane transitions between steady-level and hovering flight. *Journal of Guidance, Control, and Dynamics*, 31(2):358–370, 2008. doi: 10.2514/1.29261.

[26] Michael S Selig. Real-time flight simulation of highly maneuverable unmanned aerial vehicles. *Journal of Aircraft*, 51(6):1705–1725, 2014. doi: 10.2514/1.c032370.

[27] Waqas Khan and Meyer Nahon. Real-time modeling of agile fixed-wing UAV aerodynamics. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1188–1195. IEEE, 2015. doi: 10.1109/icuas.2015.7152411.

[28] Daniel V Uhlig and Michael S Selig. Determining aerodynamic characteristics of a micro air vehicle using motion tracking. *Journal of Aircraft*, 50(5):1481–1490, 2013. doi: 10.2514/1.c031996.

[29] Waqas Khan and Meyer Nahon. Toward an accurate physics-based UAV thruster model. *IEEE/ASME Transactions on Mechatronics*, 18(4):1269–1279, 2013. doi: 10.1109/tmech.2013.2264105.

[30] John B Brandt and Michael S Selig. Propeller performance data at low Reynolds numbers. In *AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, pages 1–18. AIAA, 2011. doi: 10.2514/6.2011-1255.

[31] Michael S Selig. Modeling propeller aerodynamics and slipstream effects on small UAVs in realtime. In *AIAA Atmospheric Flight Mechanics Conference*, pages 1–23. AIAA, 2010. doi: 10.2514/6.2010-7938.

[32] M B Milam, K Mushambi, and R M Murray. A new computational approach to real-time trajectory generation for constrained mechanical systems. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 845–851. IEEE, 2000. doi: 10.1109/CDC.2000.912875.

[33] S K Kim and D M Tilbury. Trajectory generation for a class of nonlinear systems with input and state constraints. In *Proceedings of the 2001 American Control Conference*, pages 4908–4913. IEEE, 2001. doi: 10.1109/ACC.2001.945762.

[34] Nadeem Faiz, Sunil K Agrawal, and Richard M Murray. Trajectory planning of differentially flat systems with dynamics and inequalities. *Journal of Guidance, Control, and Dynamics*, 24(2):219–227, March-April 2001. doi: 10.2514/2.4732.

[35] Jonathan Chauvin, Laure Sinegre, and Richard M Murray. Nonlinear trajectory generation for the Caltech multi-vehicle wireless testbed. In *2003 European Control Conference (ECC)*, pages 2026–2034. IEEE, 2003. doi: 10.23919/ECC.2003.7085264.

[36] Angela Schollig, Markus Hehn, Sergei Lupashin, and Raffaello D'Andrea. Feasibility of motion primitives for choreographed quadrocopter flight. In *The 2011 American Control Conference*, pages 3843–3849. IEEE, 2011. doi: 10.1109/ACC.2011.5991482.

[37] Ony Arifianto and Mazen Farhood. Optimal control of a small fixed-wing UAV about concatenated trajectories. *Control Engineering Practice*, 40:113–132, April 2015. doi: 10.1016/j.conengprac.2015.03.007.

[38] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012. doi: 10.1177/0278364911434236.

[39] Jie Tang, Arjun Singh, Nimbus Goehausen, and Pieter Abbeel. Parameterized maneuver learning for autonomous helicopter flight. In *The 2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1142–1148. IEEE, 2010. doi: 10.1109/ROBOT.2010.5509832.

[40] Marc W McConley, Michael D Piedmonte, Brent D Appleby, Emilio Frazzoli, Eric Feron, and Munther Dahleh. Hybrid control for aggressive maneuvering of autonomous aerial vehicles. In *The 19th Digital Avionics Systems Conference*, pages 14–21. IEEE, 2000. doi: 10.1109/DASC.2000.886897.

[41] Vladislav Gavrilets, Emilio Frazzoli, Bernard Mettler, M Piedmonte, and Eric Feron. Aggressive maneuvering of small autonomous helicopters: A human-centered approach. *The International Journal of Robotics Research*, 20(10):795–807, October 2001. doi: 10.1177/02783640122068100.

[42] Florian Fisch, Jakob Lenz, Florian Holzapfel, and Gottfried Sachs. Trajectory optimization applied to air races. In *AIAA Atmospheric Flight Mechanic Conference*, pages AIAA 2009–5930. AIAA, 2009. doi: 10.2514/6.2009-5930.

[43] A Rahmani, X C Ding, and M Egerstedt. Optimal motion primitives for multi-UAV convoy protection. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4469–4474. IEEE, 2010. doi: 10.1109/ROBOT.2010.5509221.

[44] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, March-April 1998. doi: 10.2514/2.4231.

[45] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002. doi: 10.2514/2.4856.

[46] Chris Dever, Bernard Mettler, Eric Feron, Jovan Popovi, and Marc Mcconley. Nonlinear trajectory generation for autonomous vehicles via parametrized maneuver classes. *Journal of Guidance, Control, and Dynamics*, 29(2):289–302, March-April 2006. doi: 10.2514/3.44916.

[47] Aditya A Paranjape, Kevin C Meier, Xichen Shi, Soon-Jo Chung, and Seth Hutchinson. Motion primitives and 3-D path planning for fast flight through a forest. *The International Journal of Robotics Research*, 34(3):357–377, 2015. doi: 10.1177/0278364914558017.

[48] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-26(1):43–49, February 1978. doi: 10.1109/TASSP.1978.1163055.

[49] Joseph Moore, Rick Cory, and Russ Tedrake. Robust post-stall perching with a simple fixed-wing glider using LQR-trees. *Bioinspiration and Biomimetics*, 9(2): 025013, 2014. doi: 10.1088/1748-3182/9/2/025013.

[50] William J Crowther. Perched landing and takeoff for fixed-wing UAVs. In *RTO AVT Symposium on Unmanned Vehicles for Aerial, Ground, and Naval Military Operations*, pages 9–13. NATO, 2000.

[51] Sanghyuk Park. Autonomous aerobatics on commanded path. *Aerospace Science and Technology*, 22(1):64–74, 2012. doi: 10.1016/j.ast.2011.06.007.

[52] Andrew J Barry. Flying between obstacles with an autonomous knife-edge maneuver. Master's thesis, Massachusetts Institute of Technology, September 2012.

[53] Takaaki Matsumoto, Atsushi Konno, Ren Suzuki, Atsushi Oosedo, Kenta Go, and Masaru Uchiyama. Agile turnaround using post-stall maneuvers for tail-sitter VTOL UAVs. In *2010 International Conference on Intelligent Robots and Systems (IROS)*, pages 1612–1617. IEEE, 2010. doi: 10.1109/IROS.2010.5650204.

[54] Rick Cory and Russ Tedrake. On the controllability of agile fixed-wing flight. In *the Symposium on Flying Insects and Robots*, pages 21–22. Springer, 2007.

[55] Alexis Lussier Desbiens and Mark R Cutkosky. Landing and perching on vertical surfaces with microspines for small unmanned air vehicles. *Journal of Intelligent & Robotic Systems*, pages 313–327, October 2009. doi: 10.1007/s10846-009-9377-z.

[56] Alan T Asbeck, Sangbae Kim, M R Cutkosky, William R Provancher, and Michele Lanzetta. Scaling hard vertical surfaces with compliant microspines arrays. *The International Journal of Robotics Research*, 25(12):1165–1179, December 2006. doi: 10.1177/0278364906072511.

[57] Elena Glassman, Alexis Lussier Desbiens, Mark Tobenkin, Mark Cutkosky, and Russ Tedrake. Region of attraction estimation for a perching aircraft: A Lyapunov method exploiting barrier certificates. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2235–2242. IEEE, 2012. doi: 10.1109/ICRA.2012.6225361.

[58] C Goerzen, Z Kong, and B Mettler. A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent & Robotic Systems*, pages 65–100, November 2009. doi: 10.1007/s10846-009-9383-1.

[59] Lydia E. Kavraki, Petr Svestka, Jean-Claude Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration space. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, August 1996. doi: 10.1109/70.508439.

[60] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning, 1998. TR 98-11, Computer Science Dept, Iowa State University.

[61] Steven M LaValle and Jr James J Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, May 2001. doi: 10.1177/02783640122067453.

[62] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846894, 2011. doi: 10.1177/0278364911406761.

[63] Marin Kobilarov. Cross-entropy motion planning. *The International Journal of Robotics Research*, 31(7):855–871, May 2012. doi: 10.1177/0278364912444543.

[64] Alejandro Perez, Sertac Karaman, Alexander Shkolnik, Emilio Frazzoli, Seth Teller, and Matthew R Walter. Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4307–4313. IEEE, 2011. doi: 10.1109/IROS.2011.6094994.

[65] Dasol Lee and David Hyunchul Shim. RRT-based path planning for fixed-wing UAVs with arrival time and approach direction constraints. In *2014 International Conference on Unmanned Aircraft Systems*, pages 317–328. IEEE, 2014. doi: 10.1109/ICUAS.2014.6842270.

[66] M V Ramana, S Aditya Varma, and Mangal Kothari. Motion planning for a fixed-wing UAV in urban environments. In *The 4th IFAC Conference on Advances in Control and Optimization of Dynamical Systems*, pages 419–424. Elsevier, 2016. doi: 10.1016/j.ifacol.2016.03.090.

[67] Emre Koyuncu, N Kemal Ure, and Gokhan Inalhan. A probabilistic algorithm for mode based motion planning of agile unmanned air vehicles in complex environments. In *Proceedings of the 17th World Congress, the International Federation of Automatic Control (IFAC)*, pages 2661–2668. Elsevier, 2008. doi: 10.3182/20080706-5-KR-1001.00448.

[68] Emre Koyuncu, N Kemal Ure, and Gokhan Inalhan. Integration of path/maneuver planning in complex environments for agile maneuvering UCAVs. In *Selected Papers from the 2nd International Symposium on UAVs*, pages 143–170. Springer, 2009. doi: 10.1007/s10846-009-9367-1.

[69] E Frazzoli, M A Dahleh, and E Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, December 2005. doi: 10.1109/TRO.2005.852260.

[70] Tom Schouwenaars, Bernard Mettler, Eric Feron, and Jonathan How. Hybrid model for trajectory planning of agile autonomous vehicles. *Journal of Aerospace Computing, Information, and Communication*, 1:629–651, December 2004. doi: 10.2514/1.12931.

[71] Brian MacAllister, Jonathan Butzke, Alex Kushleyev, Harsh Pandey, and Maxim Likhachev. Path planning for non-circular micro aerial vehicles in constrained environments. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3933–3940. IEEE, 2013. doi: 10.1109/ICRA.2013.6631131.

[72] Ross Allen and Marco Pavone. Toward a real-time framework for solving the kinodynamic motion planning problem. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 928–934. IEEE, 2015. doi: 10.1109/ICRA.2015.7139288.

[73] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, June 2017. doi: 10.1177/0278364917712421.

[74] H Jin Kim, David H Shim, and Shankar Sastry. Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles. In *Proceedings of the American Control Conference*, pages 3576–3581. IEEE, 2002. doi: 10.1109/ACC.2002.1024483.

[75] Tom Schouwenaars, Jonathan How, and Eric Feron. Receding horizon path planning with implicit safety guarantees. In *Proceedings of the 2004 American Control Conference*, pages 5576–5581. IEEE, 2004. doi: 10.23919/ACC.2004.1384742.

[76] David Hyunchul Shim and Shankar Sastry. A dynamic path generation method for a UAV swarm in the urban environment. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 1–7. AIAA, 2008. doi: 10.2514/6.2008-6836.

[77] Myung Hwangbo, James Kuffner, and Takeo Kanade. Efficient two-phase 3D motion planning for small fixed-wing UAVs. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1035–1041. IEEE, 2007. doi: 10.1109/ROBOT.2007.363121.

[78] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968. doi: 10.1109/TSSC.1968.300136.

[79] Hiparco Lins Vieira and Valdir Grassi. Improving RRT's efficiency through motion primitives generation optimization. In *2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol*, pages 37–42. IEEE, 2014. doi: 10.1109/SBR.LARS.Robocontrol.2014.20.

[80] Navid Dadkhah and Bérénice Mettler. Survey of motion planning literature in the presence of uncertainty: Considerations for UAV guidance. *Journal of Intelligent & Robotic Systems*, 65(1-4):233–246, November 2011. doi: 10.1007/s10846-011-9642-9.

[81] S M LaValle and R Sharma. A framework for motion planning in stochastic environments: Modeling and analysis. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3057–3062. IEEE, 1995. doi: 10.1109/ROBOT.1995.525719.

[82] T Schouwenaars, B Mettler, E Feron, and J P How. Robust motion planning using a maneuver automation with built-in uncertainties. In *Proceedings of the 2003 American Control Conference*, pages 2211–2216. IEEE, 2003. doi: 10.1109/ACC.2003.1243402.

[83] Alberto Bemporad and Manfred Morari. Robust model predictive control: A survey. *Robustness in Identification and Control, Lecture Notes in Control and Information Systems*, 245:207–226, September 1999. doi: 10.1007/BFb0109870.

[84] P E Missiuro and N Roy. Adapting probabilistic roadmaps to handle uncertain maps. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1261–1267. IEEE, 2006. doi: 10.1109/ROBOT.2006.1641882.

[85] A Oualid Djekoune, Karim Achour, and Redouane Toumi. A sensor based navigation algorithm for a mobile robot using the DVFF approach. *International Journal of Advanced Robotic Systems*, 6(2):97–108, June 2009. doi: 10.5772/6797.

[86] Yoko Watanabe, Anthony J Calise, and Eric N Johnson. Vision-based obstacle avoidance for UAVs. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 1–11. AIAA, 2007. doi: 10.2514/6.2007-6829.

[87] Ioannis K Nikolos, Kimon P Valavanis, Nikos C Tsourveloudis, and Anargyros N Kostaras. Evolutionary algorithm based offline/online path planner for UAV navigation. *IEEE Transactions on Systems, Man, and Cybernetics*, 33(6):898–912, December 2003. doi: 10.1109/TSMCB.2002.804370.

[88] Timothy G McGee, Stephen Spry, and J Karl Hedrick. Optimal path planning in a constant wind with a bounded turning rate. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 1–11. AIAA, 2005. doi: 10.2514/6.2005-6186.

[89] Derek R Nelson, D Blake Barber, Timothy W McLain, and Randal W Beard. Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23 (3):519–529, June 2007. doi: 10.1109/TRO.2007.898976.

[90] Alan L Jennings, Raul Ordonez, and Nicola Ceccarelli. Dynamic programming applied to UAV way point path planning in wind. In *2008 IEEE International Conference on Computer-Aided Control Systems*, pages 215–220. IEEE, 2008. doi: 10.1109/CACSD.2008.4627357.

[91] Samuel Prentice and Nicholas Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research*, 28(11-12):1448–1465, July 2009. doi: 10.1177/0278364909341659.

[92] Adam Bry, Charles Richter, Abraham Bachrach, and Nicholas Roy. Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments. *The International Journal of Robotics Research*, 34(7):969–1002, 2015. doi: 10.1177/0278364914558129.

[93] Andrew J Barry, Peter R Florence, and Russ Tedrake. High-Speed autonomous obstacle avoidance with pushbroom stereo. *Journal of Field Robotics*, 35(1):52–68, January 2018. doi: 10.1002/rob.21741.

[94] L E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, July 1957. doi: 10.2307/2372560.

[95] Waqas Khan. *Dynamics Modeling of Agile Fixed-Wing Unmanned Aerial Vehicles*. PhD thesis, McGill University, 2016.

[96] J Gordon Leishman. *Principles of Helicopter Aerodynamics, 2nd ed.* Cambridge Aerospace Series, New York, 2006.

[97] Barnes W McCormick. *Aerodynamics, Aeronautics, and Flight Mechanics*. Wiley, New York, 1995.

[98] Michael S Selig. Modeling full-envelope aerodynamics of small UAVs in realtime. In *AIAA Atmospheric Flight Mechanics Conference*, pages 1–35. AIAA, 2010. doi: 10.2514/6.2010-7635.

[99] Eitan Bulka and Meyer Nahon. Automatic control for aerobatic maneuvering of agile fixed-wing UAVs. *Journal of Intelligent & Robotic Systems*, February 2018. doi: 10.1007/s10846-018-0790-z.

[100] Randal W Beard and Timothy W McLain. *Small Unmanned Aircraft - Theory and Practice*. Princeton University Press, Princeton, New Jersey, 2012. doi: 10.1515/9781400840601-005.

[101] Carla Durazzi and Emanuele Galligani. Nonlinear programming methods for solving optimal control problems. *Equilibrium Problems: Nonsmooth Optimization and Variational Inequality Models. Nonconvex Optimization and its Applications*, 58: 71–99, 2001. doi: 10.1007/0-306-48026-3_6.

[102] Michael A Patterson and Anil V Rao. GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using HP-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software*, 41(1):1–37, 2014. doi: 10.1145/2558904.

[103] Philip E Gill, Walter Murray, and Michael A Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005. doi: 10.1137/s0036144504446096.

[104] Honeywell t-hawk micro air vehicle (mav). http://www.army-technology.com/projects/honeywell-thawk-mav-us-army/. Accessed: 2018-01-03.

[105] Steven J Mills, Jason J Ford, and Luis Mejias. Vision based control for fixed wing UAVs inspecting locally linear infrastructure using skid-to-turn maneuvers. *Journal of Intelligent and Robotic Systems*, 61(1):29–42, 2011. doi: 10.1007/s10846-010-9480-1.

[106] Mir Feroskhan and Tiauw H Go. Control strategy of sideslip perching maneuver under dynamic stall influence. *Aerospace Science and Technology*, 72:150–163, 2018. doi: 10.1016/j.ast.2017.11.002.

[107] N Ananthkrishnan and Nandan K Sinha. Level flight trim and stability analysis using extended bifurcation and continuation procedure. *Journal of Guidance, Control, and Dynamics*, 24(6):1225–1228, 2001. doi: 10.2514/2.4839.

[108] P.B. Sujit, Srikanth Saripalli, and Joao Borges Sousa. Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicles. *IEEE Control Systems*, 34(1):42–59, February 2014. doi: 10.1109/MCS.2013.2287568.

[109] S Antony Snell, Dale F Enns, and William L Garrard Jr. Nonlinear inversion flight control for a supermaneuverable aircraft. *Journal of Guidance, Control, and Dynamics*, 15(4):976–984, July-August 1992. doi: 10.2514/3.20932.

[110] Benjamin C Kuo and Farid Golnaraghi. *Automatic Control Systems*. John Wiley & Sons, New York, 2003.

[111] Lindasalwa Mud, Mumtaj Begam, and I. Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques. *Journal of Computing*, 2(3):138–143, March 2010.

[112] Toni M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2003. doi: 10.1109/CVPR.2003.1211511.

[113] Nikolaos V. Boulgouris, Konstantinos N. Plataniotis, and Dimitrios Hatzinakos. Gait recognition using dynamic time warping. In *IEEE Workshop on Multimedia Signal Processing*, pages 263–266. IEEE, 2004. doi: 10.1109/MMSP.2004.1436543.

[114] Pieter Abbeel, Adam Coates, and Andrew Y Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, June 2010. doi: 10.1177/0278364910371999.

[115] Ken Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH 1985 Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, pages 245–254. ACM, 1985. doi: 10.1145/325334.325242.

[116] Ryo Takei, Richard Tsai, Haochong Shen, and Yanina Landa. A practical path-planning algorithm for a simple car: a Hamilton-Jacobi approach. In *Proceedings*

of the 2010 American Control Conference, pages 6175–6180. IEEE, 2010. doi: 10.1109/ACC.2010.5531607.

[117] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the RRT*. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011.

[118] Israel Lugo-Cárdenas, Gerardo Flores, Sergio Salazar, and Rogelio Lozano. Dubins path generation for a fixed wing UAV. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 339–346. IEEE, 2014. doi: 10.1109/ICUAS.2014.6842272.

[119] Mark Owen, Randal W Beard, and Timothy W McLain. *Handbook of Unmanned Aerial Vehicles*. Springer Netherlands, 2014. doi: 10.1007/978-90-481-9707-1120.

[120] Sertac Karaman and Emilio Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. In *49th IEEE Conference on Decision and Control (CDC)*, pages 7681–7687. IEEE, 2011. doi: 10.1109/CDC.2010.5717430.

[121] Hamidreza Chitsaz and Steven M LaValle. Time-optimal paths for a Dubins airplane. In *2007 46th IEEE Conference on Decision and Control*, pages 2379–2384. IEEE, 2007. doi: 10.1109/CDC.2007.4434966.

[122] MAVLink micro air vehicle communication protocol. http://www.qgroundcontrol.org/mavlink/start/. Accessed: 2018-07-30.

[123] Eitan Bulka and Meyer Nahon. Autonomous fixed-wing aerobatics: From theory to flight. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6573–6580. IEEE, 2018.

[124] Rick E. Cory. *Supermaneuverable Perching*. PhD thesis, Massachusetts Institute of Technology, 2010.

[125] Bullet real-time physics simulation. https://pybullet.org/wordpress/. Accessed: 2018-08-13.

[126] Sebastian Scherer, Sanjiv Singh, Lyle Chamberlain, and Mike Elgersma. Flying fast and low among obstacles: Methodology and experiments. *The International Journal of Robotics Research*, 27(5):549–574, May 2008. doi: 10.1177/0278364908090949.

[127] Georges S Aoude, Brandon D Luders, Joshua M Joseph, Nicholas Roy, and Jonathan P How. Probabilistically safe motion planning to avoid dynamic obstacles

with uncertain motion patterns. *Autonomous Robots*, 35(1):51–76, May 2013. doi: 10.1007/s10514-013-9334-3.

[128] Alexis Lussier Desbiens, Alan T Asbeck, and Mark R Cutkosky. Landing, perching, and taking off from vertical surfaces. *The International Journal of Robotics Research*, 30(3):355–370, January 2011. doi: 10.1177/0278364910393286.