Control and Obstacle Avoidance for Agile Fixed-Wing Aircraft

Eitan Bulka

The Department of Mechanical Engineering
McGill University, Montreal

April 2021

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

Abstract

Unmanned aerial vehicles (UAVs) have been increasingly proposed for aerial surveillance, mapping, and delivery tasks. Historically these vehicles fall into two categories: conventional fixed-wing aircraft, which are capable of efficient flight over long distances but lack maneuverability, and rotorcraft, which are capable of agile and maneuverable flight but lack efficiency and endurance. Recent advancements in aerial vehicle design aim to incorporate characteristics from both rotorcraft and conventional fixed-wing aircraft, ultimately creating aircraft that are capable of both maneuverable and efficient long distance flight. These type of platforms are ideal for tasks that require both the ability to maneuver through cluttered environments, and the ability to fly long distances efficiently. An aircraft of this type, the agile fixed-wing aircraft, is a fixed-wing aircraft characterized by a high thrust-to-weight ratio (> 1), and large control surfaces capable of large deflections.

The objective of this thesis is to further the autonomous capabilities of agile fixed-wing aircraft; specifically in the context of control systems and real-time collision avoidance. The thesis begins with a discussion of a previously developed flight dynamics model, and presents a method for validating a flight dynamics model in flight regimes that rely on feedback control. Subsequently, a single control architecture is developed that can track trajectories within both conventional and aerobatic flight regimes. This architecture is then extended to be applicable to many other types of vehicles, specifically vehicles which can generate a torque in an arbitrary direction, and can apply a single body-fixed force. We demonstrate autonomous aerobatic trajectories with an agile fixed-wing aircraft, specifically knife-edge, rolling harrier, aggressive turnaround and hovering maneuvers within conventional simulations, hardware-in-the-loop simulations, indoor flight tests and outdoor flight tests. We also validate the extension to other platforms by demonstrating flips with a quadrotor in both simulation and outdoor flight tests. All flights were performed with on-board sensing and computation.

We then present a reactive obstacle avoidance algorithm that utilizes the maneuvering capabilities of agile fixed-wing aircraft and can be run in real-time with on-board sensing and computation. At each time step, trajectories are selected in real-time from a pre-computed library that lead to various positions on the edge of the obstacle sensor's field-of-view. A cost is assigned to each collision-free trajectory based on its heading toward the goal and minimum distance to obstacles, and the lowest cost trajectory is tracked. If all of the potential trajectories leading to the various positions at the edge of the obstacle sensor's field-of-view result in a collision, the aircraft has enough space to hover and come to a stop, which theoretically guarantees collision-free flight in unknown static environments. Autonomous flight in unknown and unstructured environments using only on-board sensing (stereo camera, IMU, and GPS) and computation is demonstrated with an agile fixed-wing aircraft in both simulation and outdoor flight tests. During the flight testing campaign, the aircraft autonomously flew 4.4 km in a tree-filled environment with an average speed of 8.1 $\frac{m}{s}$ and a top speed of 14.4 $\frac{m}{s}$.

Résumé

Les véhicules aériens sans pilote (UAV) sont de plus en plus proposés pour les tâches de surveillance aérienne, de cartographie et de livraison. Historiquement, ces véhicules se divisent en deux catégories: les aéronefs à voilure fixe conventionnels, qui sont capables de voler efficacement sur de longues distances mais manquent de maniabilité, et les giravions, qui sont capables de voler agilement et manœuvrablement mais manquent de l'efficacité et de l'endurance. Les progrès récents dans la conception des véhicules aériens visent à intégrer les caractéristiques des giravions et des aéronefs à voilure fixe conventionnels, créant enfin des avions capables de voler à la fois manœuvrablement et efficacement sur de longues distances. Ces types de plates-formes sont idéales pour les tâches qui nécessitent la capacité de manœuvrer dans des environnements encombrés ainsi que la capacité de voler efficacement sur de longues distances. Un aéronef de ce type, l'aéronef agile à voilure fixe, est un aéronef à voilure fixe caractérisé par un rapport poussée/poids élevé (> 1), et de grandes gouvernes capables de grandes déflections.

L'objectif de cette thèse est d'approfondir les capacités autonomes des aéronefs agiles à voilure fixe; spécifiquement dans le contexte des systèmes de contrôle et de l'évitement des collisions en temps réel. La thèse commence par une discussion d'un modèle de dynamique de vol précédemment développé et présente une méthode pour valider un modèle de dynamique de vol dans des régimes de vol qui reposent sur un contrôle de rétroaction. Par la suite, une architecture de contrôle unique est développée qui peut suivre les trajectoires dans les régimes de vol conventionnels et acrobatiques. Cette architecture est ensuite étendue pour être applicable à de nombreux autres types de véhicules, en particulier des véhicules qui peuvent générer un torque dans une direction arbitraire, et qui peuvent appliquer une force dans une seule direction. Nous démontrons des trajectoires acrobatiques autonomes avec un aéronef à voilure fixe agile dans le cadre des simulations conventionnelles, des simulations hardware-in-the-loop, des tests de vol à l'intérieur et des tests de vol à l'extérieur. Nous validons également l'extension à d'autres plates-formes en démontrant des flips avec un quadrirotor en simulation et en vol en plein air. Tous les vols ont été effectués avec la détection et le calcul à bord.

Nous présentons ensuite un algorithme d'évitement d'obstacles réactif qui utilise les capacités de manœuvre des aéronefs à voilure fixe agiles et qui peut être exécuté en temps réel avec la détection et le calcul à bord. À chaque itération, des trajectoires sont sélectionnées en temps réel à partir d'une bibliothèque pré-calculée qui mènent à différentes positions sur le bord du champ de vision du capteur d'obstacles. Un coût est attribué à chaque trajectoire sans collision en fonction de son cap vers l'objectif et de la distance minimale aux obstacles, et la trajectoire la moins coûteuse est suivie. Si toutes les trajectoires potentielles menant aux différentes positions au bord du champ de vision du capteur d'obstacles entraînent une collision, l'avion dispose suffisamment d'espace pour planer et s'arrêter, ce qui garantit théoriquement un vol sans collision dans des environnements statiques inconnus. Le vol autonome dans des environnements inconnus et non structurés en utilisant uniquement le calcul et la détection embarquée (caméra stéréo, IMU et GPS) est démontré avec un aéronef agile à voilure fixe dans les tests de

simulation et de vol en extérieur. Au cours des essais de vol, l'avion a volé de manière autonome pendant $4,4\ km$ dans un environnement arboré avec une vitesse moyenne de $8,1\ \frac{m}{s}$ et une vitesse maximale de $14,4\ \frac{m}{s}$.

Acknowledgements

First and foremost, I would like to thank my supervisor, Meyer Nahon, for his guidance, support, and constructive criticism throughout these years. I could not have asked for a better role model to guide me through this research, and in life in general.

Next, thanks to Inna Sharf, who had no official role in relation to my research, but provided excellent advice and guidance along the way. I also thank my thesis advisory committee, David Meger and Arun Misra, who both provided valuable feedback throughout the progression of the research.

I would like to express my gratitude to my many friends and colleagues in the Aerospace Mechatronics Lab. Not only did the lab environment help fuel my research, but also made this time a pleasant experience. A special thanks goes to Joshua Levin, Juan Carlos Hernandez Ramirez, and Mikkel Jorgensen. Many of the conversations I had with Josh helped shape aspects of this thesis, and I enjoyed our collaboration and friendship along the way. Juan was always willing to help with anything from hot gluing an aircraft to a controller stability analysis, and more importantly, his encouragement during the flight testing campaign was invaluable. Whether it was as a lab mate, roommate, teammate, or sous-chef, Mikkel's presence always brightened the room and made my time during graduate studies a joyful experience.

Next, I thank the Macdonald Campus of McGill University, the Concordia University Stinger Dome, the West Island Model Aeronautics Club, the Montreal Area Thermal Soarers Club, and Raad Jassim for giving me a place to fly the aircraft.

I am grateful for McGill Athletics and Recreation's intramural program, which helped me stay active and gave me lots of joy throughout my time as a McGill student. Special shout out to the legendary Geezers basketball team and Cuddles Inc. soccer team.

Finally, I'd like to thank my parents, Nancy Cooper and Zeev Bulka, who have always enabled, supported, and encouraged furthering my education. My siblings, Ben and Tamar Bulka, for their support and encouragement along the way. My girlfriend, Ana Robert, for her encouragement, support and love, and for allowing me to hog the desk during quarantine to write this thesis.

The work in this thesis was made possible with financial support from the Natural Sciences and Engineering Research Council (NSERC), the Fonds de Recherche du Quebec—Nature et technologies (FRQNT), a McGill Engineering Undergraduate Student Masters Award (MEUSMA) and by a McGill Engineering Doctoral Award (MEDA).

Claims of Originality

The main contributions of this thesis are listed below:

- A model validation technique that does not require open-loop control of the vehicle.
- A single control architecture capable of aerobatic maneuvering of agile fixed-wing UAVs, validated through simulations and flight testing on a wide range of maneuvers.
- Extending the applicability of the control architecture to any UAV that can exert a body-fixed force and apply a moment along an arbitrary axis, which is validated with quadrotor simulations and flight testing.
- A collision avoidance methodology that enables autonomous high-speed flight of an agile fixed-wing UAV in cluttered, unknown, and unstructured environments using only on-board computation and sensing. The methodology is validated in various simulations and outdoor flights.

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

- Eitan Bulka and Meyer Nahon. Autonomous control of agile fixed-wing UAVs performing aerobatic maneuvers. In *2017 international conference on unmanned aircraft systems (ICUAS)*, pages 104–113. IEEE, 2017
- 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
- Eitan Bulka and Meyer Nahon. A universal controller for unmanned aerial vehicles. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4171–4176. IEEE, 2018
- Eitan Bulka and Meyer Nahon. Automatic control for aerobatic maneuvering of agile fixed-wing UAVs. *Journal of Intelligent & Robotic Systems*, 93(1-2):85–100, 2019
- Walter Jothiraj, Corey Miles, Eitan Bulka, Inna Sharf, and Meyer Nahon. Enabling bidirectional thrust for aggressive and inverted quadrotor flight. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 534–541. IEEE, 2019
- Eitan Bulka and Meyer Nahon. High-speed obstacle-avoidance with agile fixed-wing aircraft. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 971–980. IEEE, 2019
- Eitan Bulka and Meyer Nahon. A unified control strategy for autonomous aerial vehicles. In *Autonomous Robots (Under Review)*, 2020

- Eitan Bulka and Meyer Nahon. Reactive obstacle-avoidance for agile fixed-wing unmanned aerial vehicles. In *Field Robotics (Under Review)*, 2021

Notation

Abbreviations

ADS-B	**A**utomatic **D**ependent **S**urveillance - **B**roadcast
BEC	**B**attery **E**liminator **C**ircuit
CAD	**C**omputer-**A**ided **D**esign
DC	**D**irect **C**urrent
EKF	**E**xtended **K**alman **F**ilter
ENU	**E**ast-**N**orth-**U**p
EPP	**E**xpanded **P**oly**p**ropylene
ESC	**E**lectronic **S**peed **C**ontroller
FLU	**F**ront-**L**eft-**U**p
FOV	**F**ield-**o**f-**V**iew
FRD	**F**ront-**R**ight-**D**own
GPS	**G**lobal **P**ositioning **S**ystem
HIL	**H**ardware-**i**n-the-**l**oop
IMU	**I**nertial **M**easurement **U**nit
LiPo	**Li**thium **Po**lymer
LQR	**L**inear **Q**uadratic **R**egulator
NED	**N**orth-**E**ast-**D**own
PD	**P**roportional-**D**erivative
PI	**P**roportional-**I**ntegral
PID	**P**roportional-**I**ntegral-**D**erivative
PRM	**P**robabilistic **R**oad**m**ap
PWM	**P**ulse **W**idth **M**odulation
QGC	**QG**round**C**ontrol
RC	**R**adio **C**ontrol
ROS	**R**obot **O**perating **S**ystem
RRT	**R**apidly Exploring **R**andom **T**rees

SLAM	**S**imultaneous **L**ocalization **a**nd **M**apping
UAV	**U**nmanned **A**erial **V**ehicle
UBEC	**U**niversal **B**attery **E**liminator **C**ircuit
UDP	**U**ser **D**atagram **P**rotocol
VTOL	**V**ertical **T**ake**o**ff and **L**anding Aircraft

Symbols

$\mathbf{A}$	closed-loop position error state transition matrix
$\mathbf{a}^{des}$	desired acceleration
$\mathbf{C}_{bi}$	direction cosine matrix from $\mathcal{F}_i$ to $\mathcal{F}_b$
$\mathbf{C}_{cb}$	direction cosine matrix from $\mathcal{F}_b$ to $\mathcal{F}_c$
$\mathbf{C}_{ri}$	direction cosine matrix from $\mathcal{F}_i$ to $\mathcal{F}_r$
c	total cost of potential trajectory
c^{diff}	potential trajectory cost for selecting different yaw rate
c^{h}	potential trajectory cost for steering away from goal in horizontal plane
c^{obs}	potential trajectory cost for being near obstacles
c^{v}	potential trajectory cost for steering away from goal in vertical plane
c_j	control surface to force constant for the j^{th} actuator
$\bar{c}_j$	control surface to moment constant for the j^{th} actuator
d^{obs}	minimum distance from the potential trajectory to the point cloud
$\hat{\mathbf{d}}_j$	direction of force for the j^{th} actuator
d_{xy}	distance between the aircraft and the final position in the horizontal plane
$\mathbf{e}_b$	angular error about the body frame axes
$\mathbf{f}^{aero}$	aerodynamic force
$\mathbf{f}^{c}$	control force
f^{c}	magnitude of control force ($\|\mathbf{f}^{c}\|$)
$\mathbf{f}^{nc}$	non-control force
$f^{additional}$	additional thrust to increase the control authority
$\hat{\mathbf{f}}$	direction of control force ($\frac{\mathbf{f}^{c}}{f^{c}}$)
$\hat{\mathbf{f}}^{ref}$	direction of control force of the reference aircraft
$\mathcal{F}_b$	body frame
$\mathcal{F}_c$	camera frame
$\mathcal{F}_i$	inertial frame
$\mathcal{F}_r$	reference body frame
$\mathbf{g}$	acceleration due to gravity
g	magnitude of acceleration due to gravity

$g(u_j^s)$	flapping thrust model
$\mathbf{I}$	moment of inertia with respect to center of mass
J_j	propeller advance ratio for the j^{th} actuator
K_{aero}	aerodynamic force scaling parameter
$\mathbf{K_{a_d}}$	derivative attitude control gain
$\mathbf{K_{a_p}}$	proportional attitude control gain
K_{h_i}	integral height control gain
K_{h_p}	proportional height control gain
K_{p_d}	derivative position control gain
K_{p_p}	proportional position control gain
K_v	proportional speed control gain
k	time-step
k_t	propeller thrust coefficient
k_q	propeller torque coefficient
L_{xy}	trajectory arclength in the horizontal plane
m	mass
$\mathbf{m}^c$	control moment
$\mathbf{m}^{nc}$	non-control moment
$\mathbf{n}$	axis in axis-angle parametrization
$\mathcal{N}$	normal distribution
$\mathbf{p}$	position
$\mathbf{p}^{ref}$	reference position
$\mathbf{p}^{ref/0}$	reference position with respect to aircraft
$p_{\|\|}$	planar reference position for path following
$\mathbf{q}$	orientation quaternion
$\mathbf{q}^{ref}$	reference orientation quaternion
$\bar{\mathbf{q}}^{ref}$	augmented reference orientation quaternion
$\mathbf{q}^x$	quaternion rotation of θ_x
$\mathbf{q}^y$	quaternion rotation of θ_y
$\mathbf{q}^z$	quaternion rotation of θ_z
$\mathbf{r}_j$	position vector from the center of mass to the j^{th} actuator
R	propeller radius
r_{xy}	turn radius in the horizontal plane
t	time
u_j	normalized actuator signal for the j^{th} actuator
$\mathbf{u}^s$	column matrix of actuator signals
$\mathbf{u}_j^f$	force generated by the j^{th} actuator

u_j^s	actuator signal for the j^{th} actuator
$\mathbf{u}_j^\tau$	torque generated by the j^{th} actuator
V	Lyapunov function
v	speed ($\|\|\mathbf{v}\|\|$)
$\mathbf{v}$	velocity
$\mathbf{v}^{ref}$	reference velocity
$v_{s,j}$	slipstream speed over the j^{th} actuator
w^{diff}	weight used to compute c^{diff}
w^h	weight used to compute c^h
w^{obs}	weight used to compute c^{obs}
w^v	weight used to compute c^v
x	x-position in $\mathcal{F}_i$
x^{ref}	reference x-position in $\mathcal{F}_i$
y	y-position in $\mathcal{F}_i$
y^{ref}	reference y-position in $\mathcal{F}_i$
z	z-position in $\mathcal{F}_i$
z^{ref}	reference z-position in $\mathcal{F}_i$
β	angle in axis-angle parametrization
γ	wing tilt angle
$\Delta\mathbf{h}$	height error
$\Delta\mathbf{p}$	position error
$\Delta\mathbf{q}$	error quaternion
Δt	coasting time
$\Delta\mathbf{v}$	velocity error
$\boldsymbol{\zeta}_s$	sensor s measurement
$\boldsymbol{\eta}_s$	sensor s noise
$\boldsymbol{\theta}$	triad of rotations for position control ($[\theta_x\ \theta_y\ \theta_z]$)
θ	pitch
θ^{ref}	pitch reference
θ_L	angle between the current yaw and vector from initial to final position in the horizontal plane
μ	torque cancellation constant
$\boldsymbol{\mu}$	magnetic field vector
ρ	air density
$\boldsymbol{\sigma}_s$	sensor s noise standard deviation
ϕ	roll
ϕ^{ref}	roll reference
$\boldsymbol{\chi}(x, y, z)$	transformation from North-East-Down to Geodetic coordinates

ψ	yaw
ψ^{ref}	yaw reference
Ω	desired roll rate for rolling Harrier
$\boldsymbol{\omega}$	angular velocity
$\boldsymbol{\omega}^{ref}$	reference angular velocity
$\square_0$	scalar part of quaternion
$\square_1$	first element of vector part of quaternion
$\square_{1:3}$	vector part of quaternion
$\square_2$	second element of vector part of quaternion
$\square_3$	third element of vector part of quaternion
$\square_b$	resolved in the body frame
$\square_i$	resolved in the inertial frame
$\square_{prev}$	value at the previous time step
$\square_r$	resolved in the reference body frame
$\square_x$	the x component of a vector
$\square_{xy}$	$\square$ in the horizontal plane
$\square_y$	the y component of a vector
$\square_z$	the z component of a vector
$\square^0$	initial of $\square$ (applied to $\mathbf{p}, x, y, z, \psi$)
$\square^f$	final of $\square$ (applied to $\mathbf{p}, x, y, z, \psi$)
$\square^g$	goal of $\square$ (applied to $\mathbf{p}, x, y, z$)
$\square^T$	transpose
$\square^*$	conjugate
$\dot{\square}$	time derivative
$\odot$	Hamilton quaternion product

Contents

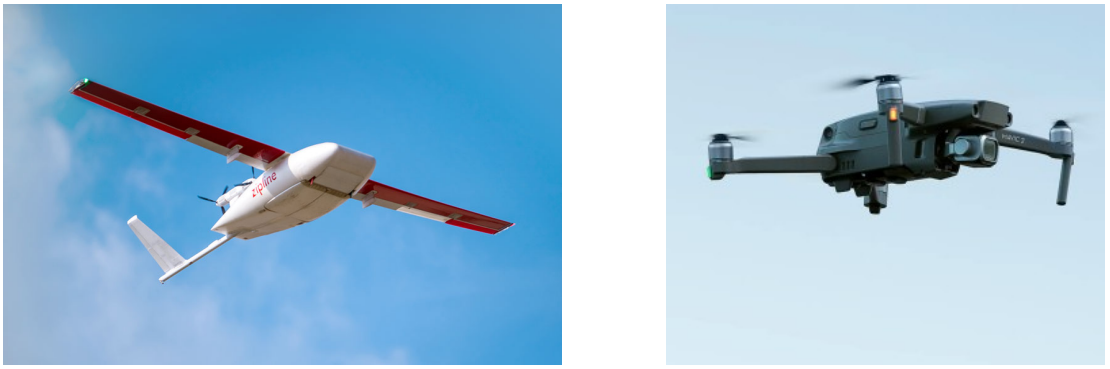
List of Figures

List of Tables

Chapter 1

Introduction

Unmanned aerial vehicles (UAVs) have been increasingly proposed for aerial surveillance, mapping, monitoring, and delivery tasks. Historically these vehicles fall into two categories: rotorcraft (Fig. 1.1b) and fixed-wing aircraft (Fig. 1.1a). Rotorcraft remain aloft by thrusting vertically, while fixed-wing aircraft use the lift generated by their wing. This fundamental difference causes rotorcraft to be less energy efficient than fixed-wing aircraft, but also allows them to take off vertically, hover, and fly at low speeds through dense cluttered environments. On the other hand, conventional fixed-wing aircraft require runways to takeoff and land and cannot operate in cluttered environments, but have the ability to fly long distances efficiently.



(a) Fixed-Wing Aircraft [13] (b) Quadrotor [14]

FIGURE 1.1: Fixed-Wing Aircraft vs. Rotorcraft

Many applications of unmanned aerial vehicles require efficient long range flight and vertical takeoff and landing capabilities. Recent advancements in aerial vehicle design aim to incorporate characteristics from both rotorcraft and fixed-wing aircraft, ultimately creating aircraft that can takeoff vertically, hover, *and* generate lift from a wing when

flying long distances. Many new types of aircraft have been proposed to obtain these capabilities, including: tilt-wing, tilt-rotor, tailsitter, flapping-wing, and agile fixed-wing aircraft. These type of platforms are ideal for tasks that require both the ability to maneuver through cluttered environments, *and* the ability to fly long distances efficiently.

In this thesis, we specifically focus on *agile fixed-wing aircraft*, which are fixed-wing aircraft characterized by a high thrust-to-weight ratio (> 1), and large control surfaces capable of large deflections. The experimental testing in this thesis utilizes McFoamy, an agile fixed-wing UAV which is depicted in Fig. 1.2.

FIGURE 1.2: McFoamy agile fixed-wing UAV

1.1 Motivation

Looking more closely at the applications of unmanned aerial vehicles, such as surveillance, mapping, monitoring, and delivery, there are many situations within these applications that require both long distance and maneuverable flight. Consider a scenario where the UAV is used to monitor wildlife. On one hand, the area being monitored could be hundreds or even thousands of meters in scale which requires efficient flight; but on the other hand, the sensor used to monitor (a camera, chemical detection, heat etc...) may need to be close enough to the ground to capture the necessary information. In this scenario, the platform would need enough agility to avoid obstacles that arise at these low altitudes such as trees or large boulders. Consider another scenario where law enforcement personnel may want to use a UAV to survey an accident or crime scene remotely. This task requires the UAV to fly a long distance to arrive at the scene, but then also the ability to remain stationary once on site. Lastly, consider a scenario where a UAV is used for package delivery. The UAV needs to fly a long distance to get to the

delivery destination, but then must be able to safely land in a confined and potentially cluttered space, in order to deliver the package.

For all of these applications, the ability to perform these missions autonomously drastically increases the appeal of this technology because of the reduction in operational cost and the scalability when using multiple UAVs. Recently, researchers have begun developing strategies to automate flight for these types of scenarios. Some motion planning strategies have been developed to autonomously guide these vehicles through cluttered environments and some control systems have been developed to automatically track aggressive trajectories. While this research is still in its infancy, radio control (RC) pilots have demonstrated impressive control over agile aircraft for many years — flying extreme aerobatic maneuvers including: backflips, barrel rolls, knife-edge, and hovering, which are classified as aerobatic flight.

1.2 Objectives

From a high-level point of view, the goal of this thesis is to achieve autonomous flight with agile fixed-wing aircraft in scenarios similar to the missions discussed above. These missions require flying outdoors in unstructured and unknown environments. Achieving this autonomously spans several research topics including: obstacle-detection, motion planning, state estimation, and controller development. In this thesis, we focus on developing control and motion planning strategies that utilize the entire maneuvering capability of agile fixed-wing aircraft. We rely on off-the-shelf hardware and open-source software for obstacle-detection and state estimation.

1.3 Literature Review

This literature review covers control systems and motion planning strategies; both in the context of unmanned aerial vehicles. While there can be significant overlap between both topics, we separate them here for clarity. A motion planning algorithm generates a collision-free reference trajectory to the goal region, while the automatic control system generates actuator signals to track the reference trajectory. For both topics, we present the most relevant literature pertaining to any type of unmanned aerial vehicle, and then narrow in on the literature specifically pertaining to agile fixed-wing aircraft.

1.3.1 Control

Historically, UAV control has mostly focused on relatively mild maneuvering, with attitudes staying reasonably close to a reference flight state in equilibrium; such as hovering for a rotor-craft or level cruise flight for a fixed-wing aircraft. Since flight in these regimes is a less active research topic, we begin the control portion of this literature review with the significant works regarding aggressive maneuvering of unmanned aerial systems. We then focus on algorithms particularly pertaining to aerobatic flight with agile fixed-wing aircraft. Finally, we investigate control strategies that apply to multiple types of UAVs.

1.3.1.1 Control Strategies for Wing-Less UAVs

Control with large attitude excursions originated in spaceflight [15]. A quaternion-based PD attitude controller in [16] is shown to be robust and globally stable. The algorithm is validated using a simulated spacecraft.

In recent years, researchers have developed control systems capable of performing aerobatic maneuvers on small RC helicopters. In [17, 18], control strategies were developed based on mimicking a human pilot's control inputs. This idea was further explored in [19], where the authors apply reinforcement learning techniques to achieve automatic aerobatic helicopter flight. Some research has also focused on developing controllers for aerobatic maneuvering of quadrotors. In [20], aerobatic quadrotor flight is achieved by decomposing maneuvers into discrete phases, where each phase has a local controller. The controllers consist of an outer-loop PID position controller, and an inner-loop PD attitude tracker. Nonlinear robust tracking control of a quadrotor is shown in [21] using geometric based control. A Lyapunov stability analysis is shown for three flight modes: an attitude controlled flight mode, a position controlled flight mode, and a velocity controlled flight mode. While some aspects of these controllers apply to fixed-wing aircraft, the aerodynamics of a wing must be accounted for in the controller design, rendering these control strategies not directly applicable to fixed-wing aircraft.

1.3.1.2 Control Strategies for Agile Fixed-Wing UAVs Executing the Hover Maneuver

There has been relatively little research developing control systems for aerobatic maneuvering of agile fixed-wing UAVs. The majority of that effort has been towards developing

controllers for level flight and hovering conditions and transitioning between these conditions. One of the earlier works focusing on these maneuvers is presented in [22], where indoor hovering with on-board computation and sensing is demonstrated with a PD attitude controller, manual throttle control, and no position control. This work is extended in [23], where the transition to hover, and hovering are achieved without any pilot input. Without having a position estimate, some form of position control is demonstrated by wall following and doorway traversal using ultrasonic and infrared sensing.

In [24], four linear quadratic controllers are used for level flight, hovering, and transitioning between the two. The control logic is validated by demonstrating transitions from hover to level-flight, which is followed by many ovals with excellent tracking. A transition from level-flight to hover, followed by hovering and a perched landing is also demonstrated. These maneuvers are performed indoors using a motion capture system for sensing and off-board computation. This work is continued using a cascaded PID controller and a nonlinear Lyapunov backstepping controller in [25].

Transitioning from level flight to hover has also been studied with fixed-wing gliders with no thrust. In [26], a glider with dihedral on the wings for passive lateral stability and only one actuator–an elevator, is used to validate an optimal controller. Successful perching on a power line is demonstrated in an indoor motion capture environment, and the computation is done off-board. This work is continued in [27], where the LQR-Trees controller is presented and validated with an experimental perching demonstration.

Perching on vertical surfaces using a fixed-wing glider with limited sensing and control is investigated in [28]. An open-loop pitching motion is initiated as the plane approaches the wall and the ensuing open-loop dynamics result in the plane reaching the wall within a range of acceptable orientation and velocity, allowing the plane to cling to the wall using microspines [29]. The authors extend the capabilities of the aircraft in [30] by demonstrating a takeoff from perched on a wall. Unlike their previous work, this latter maneuver requires closed-loop control and an aircraft with a thruster, rather than a glider. A further continuation of this work in [31] demonstrates climbing along vertical surfaces.

Some researchers embed real-time learning of aerodynamic models within the control system. A single adaptive controller that uses dynamic inversion with a real-time neural network adaptation is presented in [32]. Outdoor experimental demonstration of level flight, hovering flight, and the transitions between them are shown using on-board computation and sensing. In [33], a cascaded control strategy is presented that is based on a first-principles model of the vehicle dynamics with an on-board parameter learning scheme to estimate unknown aerodynamic parameters. The strategy is validated in flight

tests with a flying-wing tailsitter, where the aircraft performs autonomous lateral figure eights, level flight, hovering, and the transitions between level flight and hover.

1.3.1.3 Control Strategies for Agile Fixed-Wing UAVs Executing a Broader Range of Aerobatic Maneuvers

Besides the hover maneuver, the majority of autonomous aerobatics for fixed-wing UAVs has only been demonstrated in simulation. In [34], a multimodal flight control scheme is presented which is capable of performing many aerobatic maneuvers in simulation. Each maneuver is comprised of a number of flight modes, where each mode is locally controlled by a dynamic sliding mode control law. In [35], a control system is developed where a PD control law is used to track a time-varying pitch and roll trajectory. This technique is used to perform a few aerobatic maneuvers in a simulation environment and one maneuver (360° roll) in experiment.

A deep reinforcement learning controller is used in [36] to handle the nonlinear attitude control problem of a fixed-wing UAV. The method enables autonomous flight in an extended flight envelope of a traditional fixed-wing aircraft. The controller is validated in simulation by tracking an attitude with oscillating pitch and roll. In [37], deep reinforcement learning is also used to control an agile fixed-wing aircraft. The methodology is validated in simulation by performing two aerobatic maneuvers: a slow roll and knife-edge.

In [38], a control system is developed which can perform many aerobatic maneuvers along a specified flight path outdoors. The control system uses a non-linear path following guidance law in an outer-loop to create an acceleration command. The elevator and rudder are used to track the acceleration command using a PI control law. The roll can be selected independently of the flight path as this component is decoupled from tracking the specified path, which allows knife-edge and rolling harrier flight. In [39, 40], an agile fixed-wing aircraft autonomously flies between objects narrower than its wing span using the knife-edge maneuver in an indoor motion capture environment. An open-loop trajectory is formed using a direct collocation method, which is tracked using a time-varying linear quadratic regulator.

1.3.1.4 Unified Control Strategies

Many of the above state-of-the-art control strategies are tailored to a specific maneuver, and almost all of them are tailored to a specific platform. Use of these strategies would

require a different algorithm for each platform. In [41], a generalized control strategy is developed for the class of vehicles with the ability to generate a body-fixed thruster force, and three linearly independent moments. However, this strategy specifically assumes the vehicle generates small lift and drag forces. This assumption makes the strategy applicable to most wingless VTOL vehicles such as multi-rotors, but *not* applicable to winged vehicles such as fixed-wing aircraft or flying-wing tailsitters, due to the substantial wing lift they generate.

A continuation of this approach is presented in [42], in which the controller design does not assume small lift and drag forces. The authors present a Lyapunov-based controller that aims to exert a desired force by using angular velocity as a control input to align the body-fixed thrust with the direction of the desired force. This desired force is calculated based on a reference acceleration, position and velocity feedback, gravity, and aerodynamic forces. A limitation of this approach is that it is designed for axisymmetric aerial vehicles, which exclude vehicles with asymmetric body shapes such as a conventional fixed-wing aircraft.

This approach is developed further in [43], allowing the control logic to be applied to fixed-wing aircraft, and it is successfully implemented in a simulation environment. While this work presents a methodology towards a unified control approach in theory, it is unclear whether this method would work as well in practice. The controller feedback is based on the angle of attack; which is difficult to sense and estimate. In addition, the simulation control inputs are an applied torque and thrust. Although aircraft control surfaces effectively apply torque, it is difficult to know the magnitude of this torque, and thus in a hardware implementation, its likely that the torque to control surface mapping is not accurate. Beyond this, there are many complexities that are inherent to an experimental implementation, including control time delays, sensitivity to unmodeled disturbances and state estimation errors. These complexities can make a seemingly-promising approach much less promising in practice.

1.3.2 Obstacle Avoidance

Autonomously avoiding obstacles with any type of UAV is a challenging task—as they have complicated dynamics in three dimensions, often fly at high speed, and have a limited payload, leading to limited computational power and sensing options. In order to solve such a complex problem, researchers often make various simplifying assumptions. While surveying the literature it is important to acknowledge if the solution presented is implemented:

- in real-time or off-line
- in an unknown map or known map of the environment
- using on-board or off-board computation
- using on-board or off-board sensing
- with assumptions about the size or motion of the obstacles
- in flight experiments or simulation

In addition to the various assumptions that could be made, the applicability of a proposed algorithm will significantly depend on the type of vehicle and speed at which the vehicle flies.

While some researchers continue to publish methodologies with some of these underlying assumptions, others aim to relax these assumptions in order to plan the motion of UAVs in real-time in unknown environments using on-board sensing and computation with state-of-the-art hardware. Researchers designing algorithms for realistic operating scenarios are able solve the motion planning problem by breaking it down into two parts using a global and local planner [44, 45]. A global motion planner, operating at a slower update rate and reliant on a transformation of sensor data into a map of the world, is responsible for finding a collision-free trajectory to the goal if one exists. A local or reactive motion planner, operating at a faster update rate and using a more basic form of obstacle detection sensor data, is responsible for guaranteeing collision-free flight with newly-perceived obstacles.

While the long-term goal of our research is to enable an agile fixed-wing aircraft to autonomously fly through an unknown and unstructured cluttered environment using only on-board sensing and computation, we limit the scope of the motion planning aspect of this thesis to reactive obstacle avoidance. Future work can combine the contributions of this thesis with on-board mapping and a global motion planner to achieve the long-term goal. Given the very limited research on algorithms that have been implemented on agile fixed-wing UAVs in real-time in unknown environments with on-board sensing and computation, we broaden our literature review to works that demonstrate motion planning with other types of UAVs in real-time in unknown environments with on-board sensing and computation, as well as works that present motion planning with fixed-wing UAVs, but are not implemented real-time in unknown environments with on-board sensing and computation.

1.3.2.1 Motion Planning Strategies Not Validated in Unknown Environments with On-board Sensing and Computation

One of the important early works on motion planning with UAVs is presented in [46], in which a miniature helicopter navigates through an environment with sliding doors in a simulation environment. In this work, the helicopter motion is broken down into a finite number of trim states (steady-state), and maneuvers (finite-time) which connect one trim state to another. These trim states and maneuvers formulate a library of motion primitives which are precomputed off-line. During flight, a tree in the rapidly exploring random trees (RRT) algorithm [47, 48] is expanded using these motion primitives, as opposed to the entire state-space of the aircraft, which enables the algorithm to run in real-time.

In [49], motion planning for an agile fixed-wing aircraft is achieved in three steps. First, a goal biased RRT algorithm is used to find a connectivity path from initial to goal positions. Next, unnecessary detours are removed through a line-of-sight filter, and the refined path is used to generate a sequence of way points. Finally, the waypoints are converted into kinodynamically feasible paths using a model-based probabilistic roadmap (PRM)[50]. In [51], a similar approach is presented which generates a sequence of waypoints in the same manner, but this time employs a B-Spline method to generate dynamically feasible trajectories to connect waypoints. The authors believe once a library of flight modes is built (presumably off-line), their motion planning approach has real-time implementation capability.

In [52], a two-phase motion planning approach is proposed, which consists of a coarse global planner and a fine local planner. The coarse global planner computes kinematically feasible obstacle-free path in a discretized three-dimensional space, while the fine local motion planner is used to compute a dynamically feasible trajectory using motion primitives. Ultimately, this algorithm uses a pre-computed (off-line) library of motion primitives to achieve real-time motion planning; demonstrated by performing an air slalom task in simulation. This work is extended in [53], in which the task is performed in flight, while introducing a Markov decision process to compute the most probable path to a target in the presence of wind gusts and imperfect control.

In [54], two families of motion primitives, three-dimensional circular paths and aggressive-turnarounds, are used in a receding horizon control-based motion planner to enable fast flight through a dense obstacle field with an agile fixed-wing aircraft. The motion planning results are presented in simulation, but control of the maneuvers is experimentally demonstrated in a motion capture environment.

In [55], an agile fixed-wing UAV motion planning framework is presented which utilizes RRTs in conjunction with a trajectory library, similar to the method in [46]. The authors achieve real-time motion planning in cluttered environments, all while utilizing the full capabilities of the aircraft, such as an aggressive turnaround and hovering maneuvers. The algorithm is validated in a simulation in [55], and in flight tests in [56]. During the flight tests all of the computation is done on-board, but there is no on-board sensing, and instead a map of the environment is provided to the motion planner at run time.

In [57], the rapidly-exploring random belief trees motion planning algorithm is proposed. Unlike the previously mentioned sampling-based methods, this algorithm accounts for uncertainties in the sensed environment, and converges to an optimal path. The methodology is tested in a simulation of a two-dimensional system with a Dubins-type vehicle dynamics, and is intended to be later implemented on a fixed-wing aircraft in real-time. Later in [58], the authors couple the search process with optimization in the output space of a differentially flat vehicles to find aggressive trajectories that utilize the full maneuvering capabilities of a quadrotor. This is extended to fixed-wing vehicles with a novel trajectory representation called a "Dubins-Polynomial trajectory". This planning strategy is used in conjunction with the work in [59], where an IMU and planar laser range finder are used in an extension of the Gaussian particle filter to localize a fixed-wing aircraft within a predetermined map of the environment. These algorithms enable an autonomous fixed-wing aircraft traveling at over $11\frac{m}{s}$ to avoid obstacles in a GPS-denied parking garage.

A motion planning framework is presented in [60], where an agile fixed-wing aircraft avoids obstacles utilizing the full aerobatic capabilities of the aircraft. The algorithm runs in real-time and is guaranteed to succeed despite uncertainty. The approach utilizes a precomputed library of funnels along different maneuvers, where the system state is guaranteed to remain inside the funnel throughout the maneuver. While these results are very impressive, they are obtained in a motion capture environment where the obstacle locations are given to the planner at run-time, and the algorithm is run off-board.

Receding horizon path planning with implicit safety guarantees is presented in [61], and demonstrated in a simulation of a fixed-wing aircraft. In [62], direct nonlinear model predictive control is used with a minimalistic dynamics model to maneuver an agile fixed-wing UAV in constrained spaces in real-time at 5 Hz. Randomized motion planning is used to avoid local minima and local-linear feedback is used to compensate for model inaccuracies between updates. The methodology is validated with flight tests through a virtual narrow corridor, where a motion capture system is used for sensing and the computation is done off-board.

Genetic algorithms have also been proposed in UAV path planning in [63]. Two methods are proposed, an off-line planner in a known environment, and an on-line planner in an unknown environment. The off-line planner generates a single B-Spline curve that connects the starting and target points, while the on-line planner produces a trajectory comprised of multiple smaller B-Spline curves that are smoothly connected together. While no experimental results are presented, the on-line algorithm is meant to be used in conjunction with a radar.

A simple and computationally efficient class of reactive obstacle avoidance algorithms are potential field methods, in which obstacles produce virtual repelling forces while the goal produces a virtual attractive force, ultimately avoiding obstacles and steering towards the goal. A popular potential field method, the vector field histogram [64], has been extended to three dimensions and applied to a quadrotor in simulation in [65]. Although not demonstrated in any simulation or flight test environment, a potential field based reactive controller for an agile fixed-wing UAV is presented in [66], which generates a desired yaw rate to steer around obstacles.

While the majority of active research focuses on autonomous flight through cluttered environments with static (or slowly-moving) obstacles, there has also been research on UAVs flying in a relatively open sky, where the other obstacles are other aircraft. Most commercial aircraft transmit an Automatic Dependent Surveillance-Broadcast (ADS-B) message, which allows a UAV with an ADS-B receiver to receive the position and velocity information of other nearby aircraft. These ADS-B messages are utilized in a dynamic obstacle avoidance framework for fixed-wing UAVs presented in [67, 68]. Closed-loop RRT is used to generate intermediate avoidance waypoints to ensure collision-free flight. The methodology is validated using recorded ADS-B messages nearby an airport within a hardware-in-the-loop simulator.

1.3.2.2 Motion Planning Strategies Validated in Unknown Environments with On-board Sensing and Computation

We now turn our attention to motion planning strategies that have been validated experimentally in more realistic operating environments, but in most cases using aerial vehicles other than fixed-wing aircraft.

The Vector Field Histogram is experimentally validated on a quadrotor equipped with a two-dimensional LiDAR in [69]. Two-dimensional (constant height) obstacle avoidance around two large panels is demonstrated.

High-speed ($3\frac{m}{s}$) collision avoidance in unknown environments using on-board sensing and computation is achieved with a quadrotor in [70]. A short range planner uses a local map to generate a dynamically feasible collision-free trajectory with a safe stopping policy. Collision avoidance is guaranteed since the quadrotor is always able to come to an emergency stop if needed. After an emergency stop, a long range planner can be used for completeness.

A computationally efficient collision avoidance strategy using instantaneous perception data for high-speed quadrotor flight is presented in [71]. By exploiting the differential flatness of quadrotors, minimum-time motion primitives are generated in real-time and a cost for each primitive is assigned based on angle differences between heading and heading to the goal, and the angle difference between heading and previously selected heading. The lowest cost collision-free primitive is selected to execute. Safety is guaranteed by restricting the motion primitives to remain within the sensors field-of-view and ensuring a stop maneuver is possible if no collision-free path exists. Experimental results are demonstrated in a cluttered environment with a quadrotor flying at $3\frac{m}{s}$ with on-board perception, planning, and control. A motion capture system is used, but only for vehicle state information. This work is extended in [72] by using a relaxed constraint Model Predictive Control framework. A main component of the methodology is the ability to safely use a previously generated motion primitive which enables motions outside the perception field-of-view to have guaranteed safety.

The standard method of guaranteeing safety in unknown environments is by ensuring the ability to stop within the known free space. This imposes limitations on the speed of the vehicle. The work in [73, 74] aims to ensure safety without sacrificing speed. The proposed methodology, FASTER (Fast and Safe Trajectory Planner), enables a quadrotor to autonomously fly in unknown cluttered environments at speeds up to $7.8\frac{m}{s}$. The approach optimizes trajectories in both the free-known and unknown spaces while ensuring safety by always having a feasible, safe back-up trajectory in the free-known space at the start of each replanning step.

A methodology presented in [45] enables a quadrotor to autonomously fly in flights totaling 22 km in unknown urban environments at speeds up to $9.4\frac{m}{s}$. A sparse global path to the goal is found using an occupancy grid. A position along the global path is chosen as the local goal for the local planner. A set of minimum-jerk motion primitives are computed and evaluated based on the distance to local goal, dynamic feasibility, depth image, and occupancy grid. The best motion primitive is followed until the next depth image arrives.

In [44], a two-stage three-dimensional motion planner for UAVs is presented, which uses a potential field based method for local planning and a Laplacian-based planner for global planning. The algorithm is applied to large unmanned RC helicopter equipped with a three-dimensional LiDAR. Although this motion planner does not respect the vehicle dynamics, the methodology has demonstrated over 700 successful runs of obstacle avoidance in unchartered cluttered environments traveling at speeds up to $10\frac{m}{s}$, all using on-board real-time motion planning on-board.

Obstacle avoidance with a 20 g flapping-wing aircraft using on-board sensing and computation is demonstrated in [75]. The "Droplet" strategy is proposed which uses a droplet shaped region such a that a future circular trajectory remains within the current field-of-view of the stereo cameras. If an obstacle enters this droplet region, the aircraft turns along that circular trajectory until a new collision-free droplet region is found. In the event a collision-free droplet does not exist while turning, the aircraft can remain in this circular collision-free trajectory indefinitely, guaranteeing collision-free flight (in the absence of sensor and motor noise).

Insect-inspired optic-flow based collision avoidance is demonstrated in an unknown environment using on-board sensing and computation with a fixed-wing aircraft in [76]. The relatively simple strategy enables high-speed ($14\ \frac{m}{s}$) avoidance of large groups of trees in flight testing, as well as in an urban-like environment in simulation. However, the gaps of free-space in both scenarios are wide ($> 50\ m$). It would be difficult to use this methodology in dense obstacle fields, as it fails when the obstacle is symmetrically in front of the aircraft. Optic-flow based collision avoidance is also used in [77] to demonstrate autonomous flight in an unknown environment using on-board sensing and computation with an agile fixed-wing UAV. The approach is used to avoid lateral collisions but fails when obstacles are directly in front of the aircraft. In the failure scenario, the authors propose using a distance sensor to initiate a transition to hover to avoid the collision.

The most significant research in fixed-wing obstacle-avoidance in unstructured and unknown environments is presented in [78, 79]. A fixed-wing aircraft detects and avoids trees flying up to $14\frac{m}{s}$ using all on-board sensing and computation, without the prior knowledge of a map of the environment. The obstacle avoidance algorithm relies on a library of precomputed trajectories, and the trajectory that is furthest from the point cloud of obstacles is selected. While this work is extremely impressive, it is open to improvement. The author mentions two failure scenarios with this planning algorithm: one associated with an insufficiently rich maneuver library, and the other with the trajectory initial state. The first failure mode occurs when there is no trajectory in the library that can avoid the obstacle. The second failure mode is best illustrated through an example. If

the aircraft needs to bank right, and is already slightly banked right, but the time-varying 'bank right' maneuver starts from level flight, the aircraft will initially bank left to first return to level flight. The aircraft will ultimately not turn as quickly, potentially causing a collision. In addition to the failure modes, the planning algorithm does not have a goal: the aircraft is commanded to fly straight, unless it needs to avoid obstacles.

1.4 Open Research Questions

While significant progress has been made towards control and obstacle avoidance strategies for unmanned aerial vehicles, there are still many open research questions yet to be solved. Even though there are many open research questions in this field, we highlight the questions tackled in this thesis:

- Is it possible to develop a control architecture that can execute a wide range of aerobatic trajectories using a single set of control gains?
- Is it possible to develop a control architecture that is directly applicable to wide range of vehicles?
- Is it possible to utilize the agile maneuvering capabilities of an agile fixed-wing aircraft in order to autonomously fly through an unstructured and unknown obstacle-filled environment?

In this thesis, we develop methods to solve these research questions, and validate these methods in realistic testing conditions—an aircraft flying outdoors using only on-board sensing and computation.

1.5 Thesis Organization

The thesis is organized as follows. Chapter 2 discusses the dynamics modelling of an agile fixed-wing aircraft, which is used in various simulations, and for trajectory generation. In addition to discussing the model, a model validation method is proposed. A feedback controller for agile fixed-wing UAVs is developed and presented in Chapter 3. In that same chapter, the controller is also applied to a broader range of flight vehicles. The controller is then validated in both simulation and flight testing in Chapter 4. An obstacle avoidance algorithm is developed and presented in Chapter 5. In Chapter 6, the obstacle avoidance algorithm is validated in both simulation and flight testing using on-board computation and sensing in unknown environments. To provide a more intuitive perspective of the

thesis organization, its structure is shown in Fig. 1.3, overlaid on a block diagram of the system's feedback control structure. Each block is labeled with the associated chapter. In Chapter 7, we conclude the thesis and suggest future work.

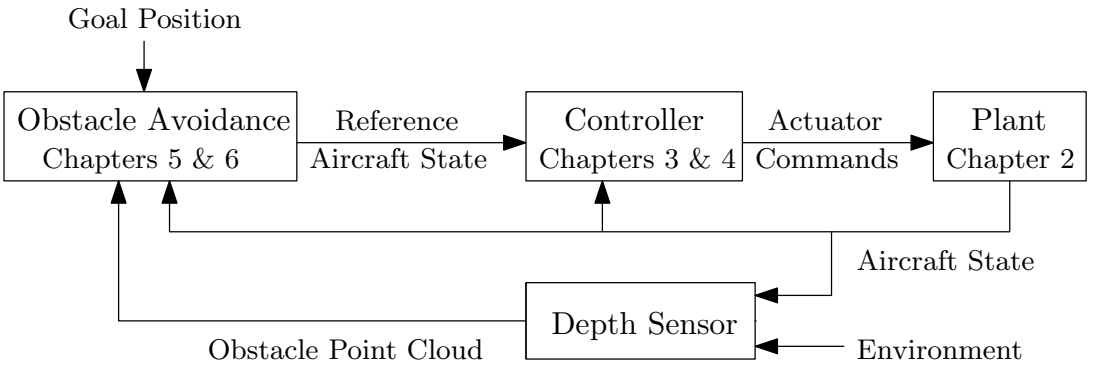


FIGURE 1.3: Thesis Structure Block Diagram

Chapter 2

Modelling Methodology and Validation

This chapter discusses the mathematical modeling of agile fixed-wing UAV motion. The model is based on first principles and can be used for various agile fixed-wing aircraft. The model is used as a tool towards achieving the objectives of this thesis, but is not a contribution of this thesis, and it is therefore not presented here in detail. A general overview is given here, while a complete detailed description of the model can be found in [9]. The components of the model are presented in various articles which are cited throughout this summary.

While different methods of model validation are presented in [9], the authors did not validate the model using real flight data. We develop an approach for model validation of an agile fixed-wing UAV using real flight data, and validate the model using this technique.

2.1 Modelling Methodology

2.1.1 UAV Kinematics and Dynamics

We use two reference frames to describe the dynamics of a UAV. $\mathcal{F}_i$ is the ground-fixed inertial reference frame with north-east-down basis vectors and $\mathcal{F}_b$ is the body-fixed reference frame. Both of these frames are depicted in Fig. 2.1.

The translational and rotational dynamics of a UAV can be derived from the Newton-Euler equations for a single rigid body, which can be succinctly stated in the following first-order form:

$$\dot{\mathbf{p}}_i = \mathbf{C}_{bi}^T \mathbf{v}_b \tag{2.1}$$

$$\dot{\mathbf{v}}_b = \frac{1}{m}(\mathbf{f}_b^{nc} + \mathbf{f}_b^c) - \boldsymbol{\omega_b} \times \mathbf{v}_b \tag{2.2}$$

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \odot \boldsymbol{\omega}_b \tag{2.3}$$

$$\dot{\boldsymbol{\omega}}_b = \mathbf{I}_b^{-1}\left((\mathbf{m}_b^{nc} + \mathbf{m}_b^c) - \boldsymbol{\omega}_b \times \mathbf{I}_b \boldsymbol{\omega}_b\right) \tag{2.4}$$

where $\odot$ is the Hamilton quaternion product, $\mathbf{p}_i$ is the absolute position of the UAV centre of mass expressed as components in $\mathcal{F}_i$ (designated with subscript i) and $\mathbf{q}$ is the aircraft orientation, expressed as a unit quaternion, $\mathbf{q} = \left[q_0, \mathbf{q}_{1:3}^T\right]^T$. Analogously at the velocity level, $\mathbf{v}_b$ is the translational velocity of the centre of mass and $\boldsymbol{\omega}_b$ is the rotational velocity of the UAV, both expressed in the body-fixed frame (designated with subscript b). The direction cosine matrix $\mathbf{C}_{bi}$ describes the orientation of the body frame relative to the inertial frame and can be formed from $\mathbf{q}$:

$$C_{bi} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 + q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_1 q_2 - q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 + q_0 q_1) \\ 2(q_1 q_3 + q_0 q_2) & 2(q_2 q_3 - q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}. \tag{2.5}$$

The mass of the UAV is denoted by m, and $\mathbf{I}_b$ is the moment of inertia relative to the center of mass, resolved in the body frame.

For controller development purposes, we separate the cumulative forces and moments acting on the UAV into a control and non-control force, $\mathbf{f}_b^c$ & $\mathbf{f}_b^{nc}$, and a control and non-control moment, $\mathbf{m}_b^c$ & $\mathbf{m}_b^{nc}$. The dominant control force of an agile fixed-wing UAV is the propeller thrust, which is written as a product of the magnitude of the control force, f^c, and the unit vector in the direction of the body-fixed thrust force, $\hat{\mathbf{f}}_b$, which is resolved in the body frame:

$$\mathbf{f}_b^c = f^c \hat{\mathbf{f}}_b. \tag{2.6}$$

The control moment is generated by the control surfaces. The non-control force is the sum of all the forces exerted on the aircraft excluding the propeller thrust, and the non-control moment is the sum of all the moments exerted on the aircraft excluding the moment generated by the control surfaces.

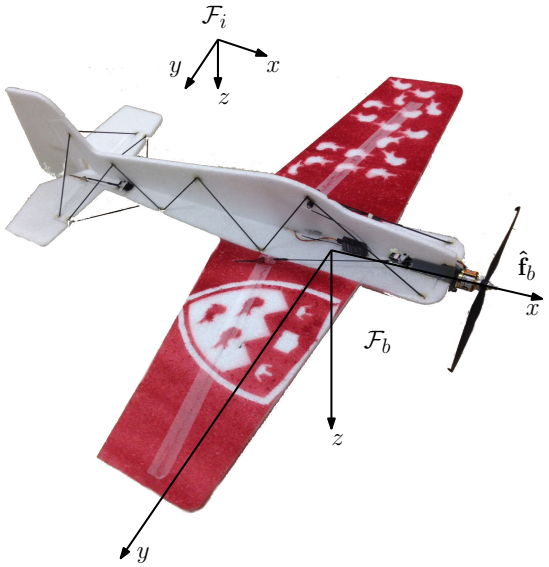


FIGURE 2.1: Coordinate Frames

The cumulative control and non-control forces and moments stem from gravity, aerodynamics, and the thruster. While the gravity force is trivial to obtain, the rest of these generalized forces are difficult to model. We use three lower-level models to model these forces and moments: a thruster model, a propeller slipstream model, and an aerodynamics model.

2.1.2 Thruster Model

The thruster model [80] uses the propeller geometry to predict the total force (3-axis) and moment (3-axis) exerted by the propeller as a function of the incoming airflow and propeller rotational speed. The thruster model is based on blade element momentum theory (see Fig. 2.2), coupled with an inflow model to predict the thruster generalized forces. The model also includes the modelling of the thruster gyroscopic effects, and can account for various incoming airflow conditions that can occur in an aerobatic flight of an agile fixed-wing aircraft. These incoming airflow conditions can be: stationary (when the aircraft is hovering), pure axial flow (flow aligned with the propeller rotational axis, i.e. level flight), oblique flow conditions (flow at an angle to its rotation axis, due to aerobatic flight or wind gusts), and in reverse flow conditions.

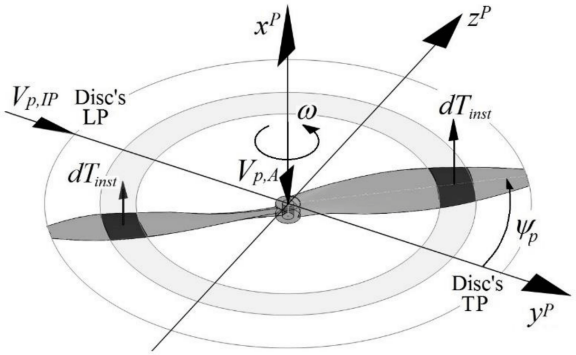


FIGURE 2.2: Thruster Model [9]

2.1.3 Propeller Slipstream Model

The propeller slipstream (see Fig. 2.3) has an important impact on the ability of the control surfaces to generate generalized forces. The slipstream provides additional flow over the control surfaces, enabling the aircraft to maintain control authority at low speeds. The novel slipstream model [81] includes two components: the axial velocity and the swirl velocity. The axial component models both the acceleration and diffusion phenomena that occur in the slipstream. The model has been shown to accurately predict the axial slipstream velocity up to 6 propeller diameters downstream of the propeller. On the other hand, the swirl velocity is known to cause a rolling moment on the aircraft that reduces the thruster reaction moment. To model this, the rolling reaction moment caused by the thruster [82] is reduced accordingly.

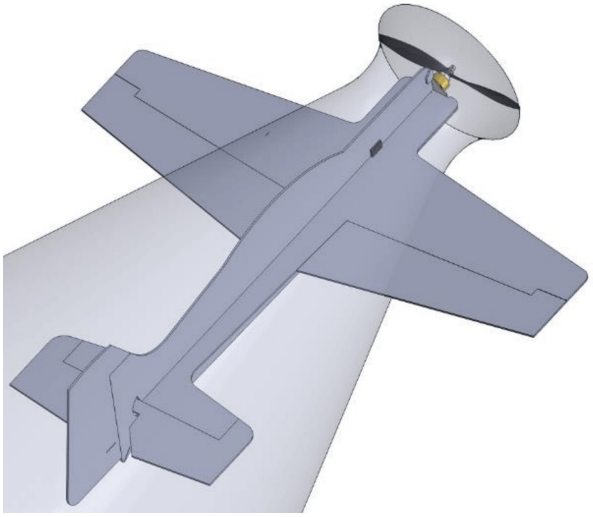


FIGURE 2.3: Propeller Slipstream Model [9]

2.1.4 Aerodynamics Model

The aerodynamics model [83] uses the aircraft geometry to predict the aerodynamic forces exerted on the aircraft as a function of the incoming airflow (which includes the propeller slipstream). These aerodynamic forces are calculated based on a component breakdown approach that partitions each component of the aircraft (wing, tail, rudder etc.) into small segments that produce lift, drag, and moment about the aircraft's center of mass (see Fig. 2.4). The aerodynamics model includes modelling of: the full flight envelope (i.e $\pm$ 180° angle of attack and sideslip range), partial flow conditions over the aerodynamic surfaces, low aspect ratio surfaces, and large control surfaces with large deflections.

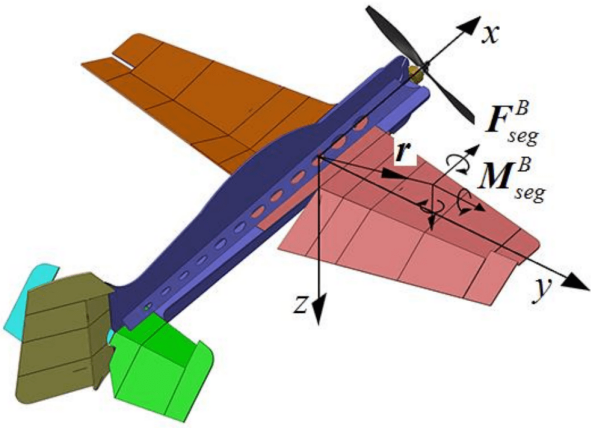


FIGURE 2.4: Aerodynamics Model [9]

2.1.5 Complete Flight Simulator

The components are assembled to create a real-time flight simulator, which is used throughout the thesis. First, the model is used in a real-time simulation, which is implemented in Matlab/Simulink and uses X-Plane [84] for visualization. This implementation is extended to enable hardware-in-the-loop simulation. Both simulations are used as an initial validation of the control algorithm before proceeding to flight tests. Next, the complete model is used as a dynamic constraint in a trajectory optimization solver which is used to generate reference trajectories for the obstacle avoidance algorithm. Lastly, the complete model is implemented in Gazebo [85], which also contains obstacles and a simulated depth camera. This implementation is used to initially validate the obstacle avoidance algorithm before flight testing in obstacle-dense environments.

2.2 Model Validation

The individual components of the physics-based model, including thruster, slipstream, and aerodynamics have been individually validated in [9] through wind-tunnel tests and static bench tests. In addition to the validation of the component models, the complete flight simulator was qualitatively validated through pilot-in-loop simulations, in which an experienced professional RC pilot found the simulator to be accurate [9]. In addition to qualitative validation, the complete flight simulator should also be validated quantitatively. However, doing so is not straightforward, as will now be discussed.

Using experimental flight data to validate the complete simulator is ideal [9]. The same control inputs could be given to both the real and simulated aircraft, and the response (aircraft states) of both the simulated and real aircraft can be compared. Variants of this idea have been proposed in the literature [86–90].

A nonlinear fixed-wing UAV model with a linear force and moment model based on stability and control derivatives is presented in [86]. In flight tests, open-loop commands are used to experimentally determine the stability and control derivatives, which are then compared to those computed from Digital DATCOM. Three different fixed-wing UAV models are evaluated using open-loop flight tests in [87]. The response of each dynamic mode (Dutch Roll, roll, short period and Phugoid) in flight are compared to those in each model. In [88], a fixed-wing dynamics model is validated by giving the same open-loop control input in flight and in simulation, and the roll, roll rate, pitch and pitch rate are compared between the flight and simulation. A similar approach is used in [89], where the aircraft is flown with the same constant throttle and zero control surface deflections in both experiment and simulation, and the complete position and attitude are compared between the two. In [90], a UAV is flown outdoors while the control inputs and wind are recorded. They are then input into the simulator, and the simulated response is compared to the actual response.

These approaches are all reliant on the UAV remaining stable while flying open-loop. While it may be possible to achieve stable open-loop level flight with an agile fixed-wing UAV, aggressive maneuvering is heavily reliant on feedback control, whether from a pilot or automatic controller, and it is for these flight regimes where the novelty of the model lies and thus validation has particular importance. The necessity for closed-loop control to perform these maneuvers hinders the ability to replicate the same motion between simulation and flight testing with the same set of control inputs. If the aircraft was flown in closed-loop during flight tests, and then the recorded control inputs were input to the simulator, the aircraft would be flown open-loop in the simulator. Since some

uncertainties and unmodeled effects are always present, and hence some degree of error will exist between the model and real system, an open-loop input will cause this error to propagate over time and eventually dominate the resultant motion [9]. In light of this, validation based on flight data was not pursued in [9] nor did that work develop a closed-loop controller. However, controller development, closed-loop flight tests, and subsequent flight test validation was recommended for future work in [9].

A closed-loop controller is developed in this thesis, and by using the same controller and reference trajectory in both simulation and experiment, the resultant motion from both environments can be compared. As shown in Chapter 4, the motion from both the simulation and flight tests are similar. However, since the control inputs are not the same in both environments, similar motion does not necessarily imply that the plant model is accurate due to the presence of the controller. Any validation that includes a closed-loop controller will have the same issue.

We propose a method of validation that does not require a control system, and does not propagate errors due to a lack of feedback into the simulated system. The core of the dynamics model predicts the forces and moments due to the aerodynamics and the thruster, and then these forces and moments cause translational and angular accelerations, which are integrated to predict the motion of the aircraft. It is the integration step that creates problems when performing validation with the open-loop plant model, due to the propagation of errors. If we were to instead validate the model by comparing the simulated translational and angular accelerations with the experimental ones, the integration is bypassed and errors are no longer propagated.

To accomplish this, the aircraft is flown (closed-loop) in a wind-free environment, and the state estimates and control inputs are recorded. The recorded velocity and angular velocity are then used to numerically approximate the translational and angular acceleration during the flight test at each time-step. The translational acceleration can be approximated during flight using the forward-euler derivative approximation for the k^{th} time-step as

$$\dot{\mathbf{v}}_b(k) = \frac{\mathbf{v}_b(k+1) - \mathbf{v}_b(k)}{t(k+1) - t(k)} \tag{2.7}$$

where t is the time in seconds. Similarly, the in-flight angular acceleration can be approximated as

$$\dot{\boldsymbol{\omega}}_b = \frac{\boldsymbol{\omega}_b(k+1) - \boldsymbol{\omega}_b(k)}{t(k+1) - t(k)}. \tag{2.8}$$

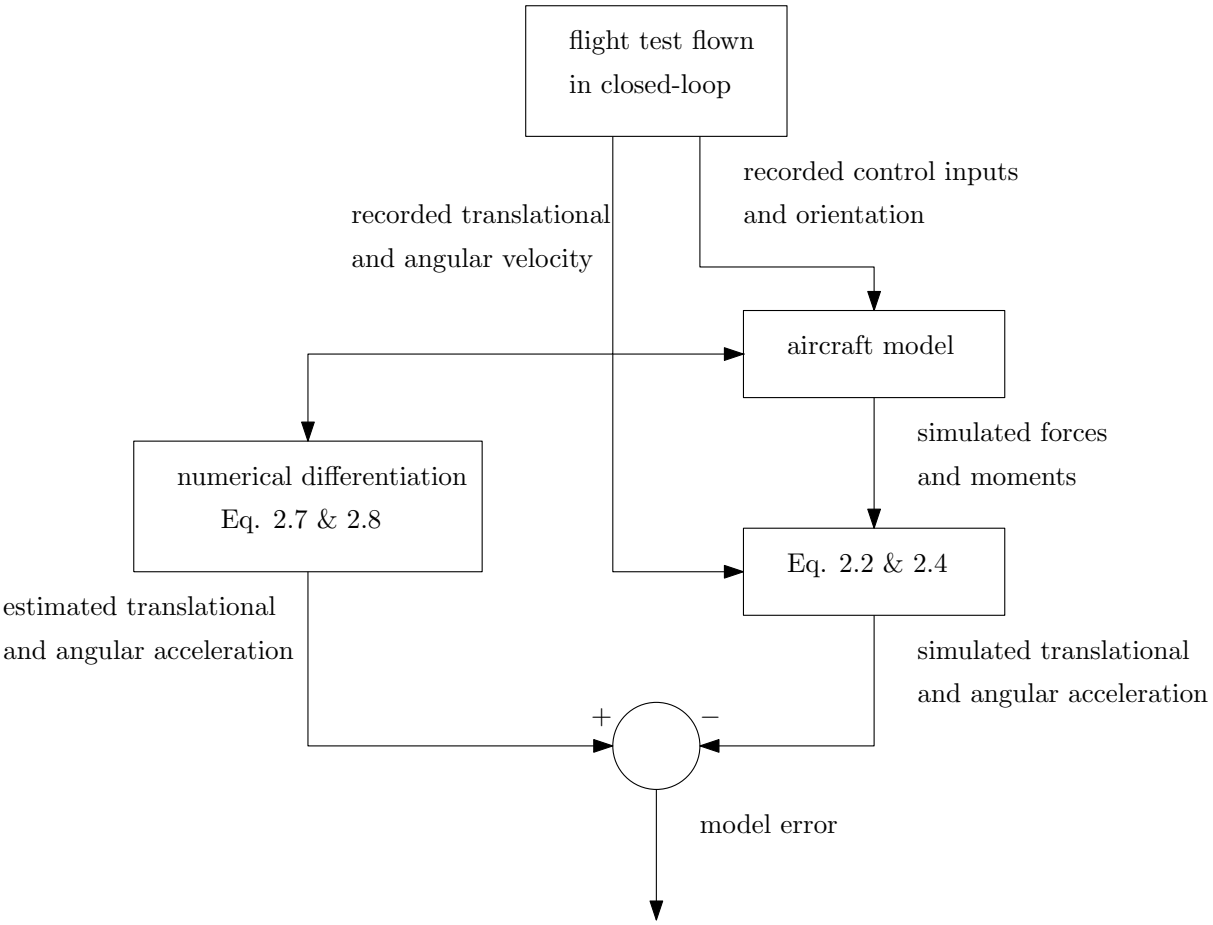


FIGURE 2.5: Block Diagram of Model Validation Process

The average time-step between each data point, i.e. the average of $t(k+1) - t(k)$ for all k, was $0.008s$. Due to the small time-step available in comparison the aircraft motion, we are able to obtain adequate derivative estimates using the simple forward-euler derivative approximation.

Separately, the recorded control inputs, orientation, velocity, and angular velocity estimates can be fed into the aircraft model at each time-step, to obtain the force and moment predicted by the model. Using the predicted force and moment, the recorded velocities, estimates of the aircraft's mass and moment of inertia, and Eq. (2.2) & Eq. (2.4), we can estimate the simulated translational and angular acceleration at each time-step. The complete model validation process is summarized by the block diagram in Fig. 2.5.

2.2.1 Results

We analyze a flight containing manual flight followed by a sequence of automated maneuvers including level flight, inverted flight, rolling Harrier, aggressive turnaround, knife-edge, and hover. We compare the model's predicted translational and angular acceleration with those extracted from the experiment in Fig. 2.6. In Fig. 2.7, we show the corresponding aircraft orientation (roll (ϕ), pitch (θ), yaw (ψ)), translational and angular velocity, and control inputs, which were recorded during flight and used in the model. The control inputs are presented with normalized units, where the minimum and maximum control surfaces deflections correspond to -1 and 1 respectively, and the minimum and maximum thrust correspond to 0 and 1. While we only display this particular 55 seconds of flight, the flight contained four of these maneuver sequences, all showing similar results.

There is a good match between the predicted and experimental translational acceleration, while there are larger discrepancies in angular acceleration. While a further investigation, outside the scope of this thesis, is required to confidently explain the cause of these discrepancies, we propose some possible explanations. First, it is highly likely that the moment of inertia approximation has more error than the mass estimate, since the moment of inertia is obtained from a CAD model, while the mass is found by direct measurement with a scale. This could explain why the translational acceleration is more accurate than the angular acceleration. Second, we suspect error with the methodology that combines propeller slipstream airflow with the incoming airflow. This hypothesis is backed up in the data, when the angular acceleration prediction is more accurate when the aircraft is in a stationary hover ($t = 135 - 140$) but inaccurate when the aircraft is traveling at high-speed and has a large throttle command ($t = 105 - 110$ and $t = 115 - 120$). In hover, the control moment is solely generated through the slipstream, and thus to accurately predict the angular acceleration, the slipstream model is likely accurate. When flying at significant speed, the control authority will arise from both the airflow and the propeller slipstream, and thus error in angular acceleration could stem from the modelling of the airflow speed over the control surface. Moreover, the slipstream model was only validated in stationary conditions in [9].

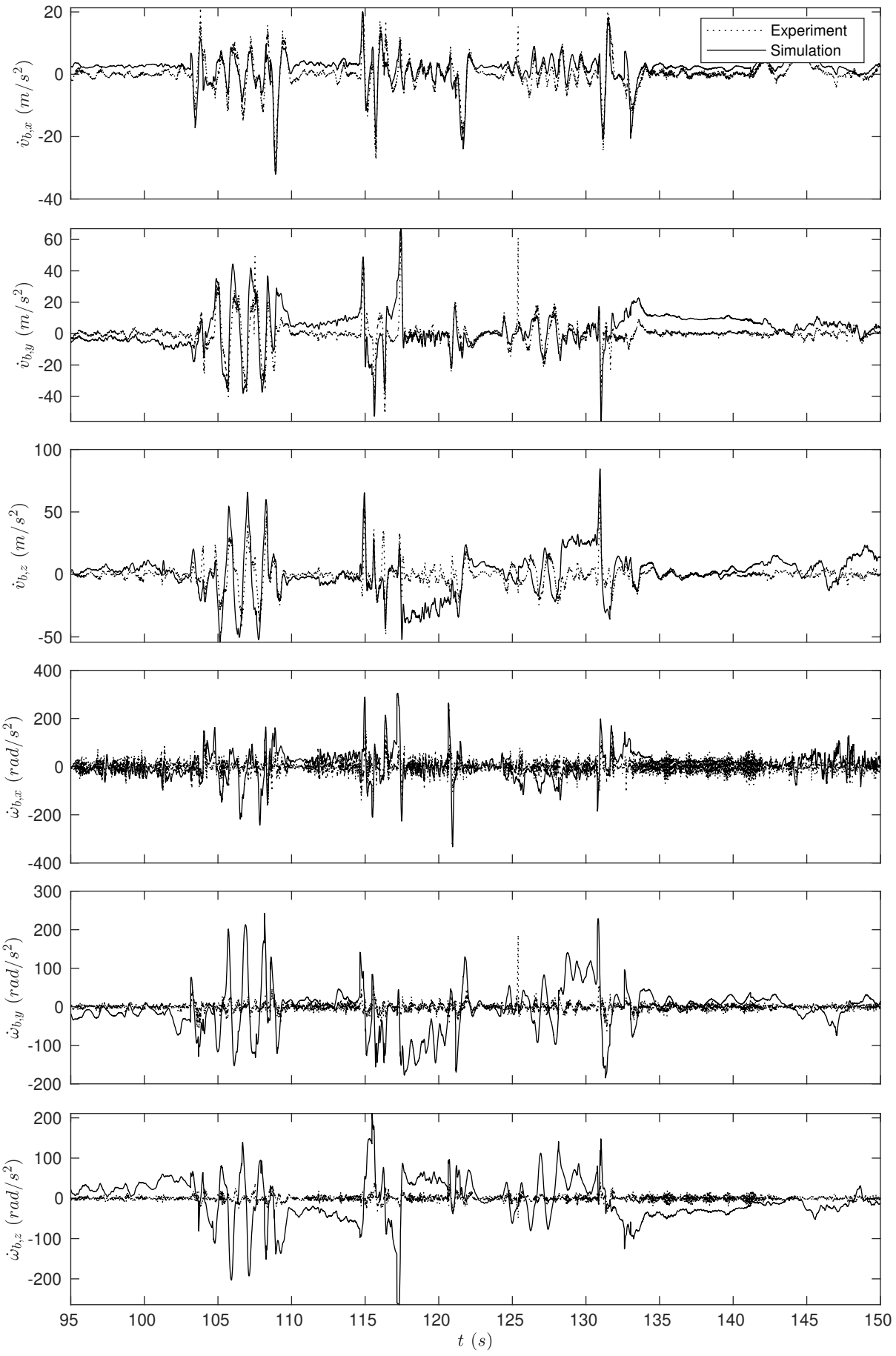


FIGURE 2.6: Predicted Vs. Actual Acceleration

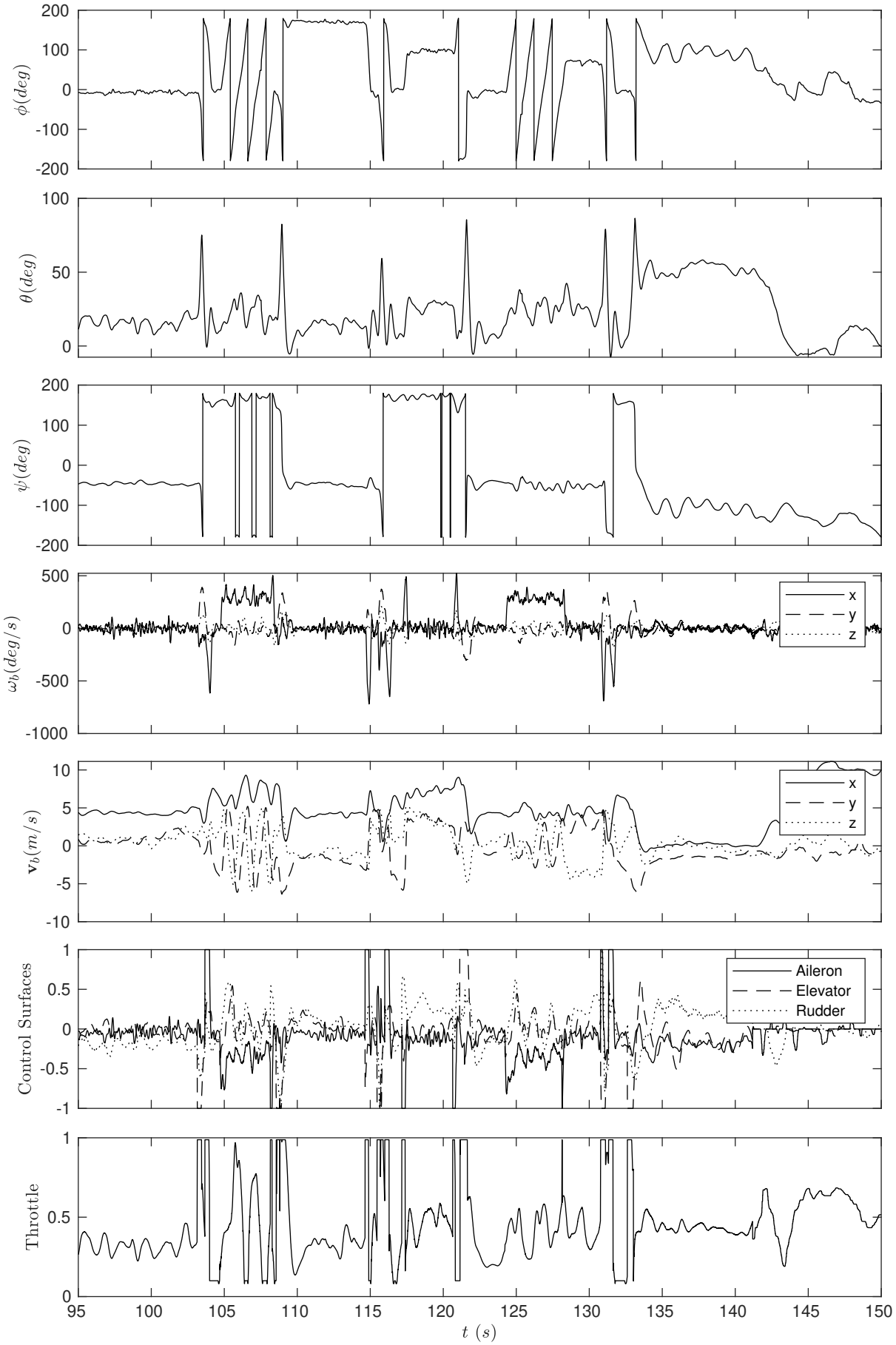


FIGURE 2.7: Model Validation Flight Data

Chapter 3

Controller

An aircraft control system must adjust the aircraft's pose using its available control inputs: aileron, elevator, rudder, and thrust. This is a challenging control problem because the system is under-actuated, and the actuators' effectiveness varies with the aircraft speed. In this chapter we present a *single* controller capable of following any feasible trajectory while maintaining the ability to recover from large deviations from the reference trajectory. These objectives are achieved by developing a nonlinear control system that is based on the physics of the aircraft, allowing simple control laws to achieve precise tracking of highly non-linear dynamics. We avoid any plant linearization, to ensure that the controller will remain effective throughout the entire flight envelope of the agile aircraft.

The control surfaces of a fixed-wing aircraft can control orientation but cannot directly control position. However, the orientation of an aircraft *can* indirectly control the aircraft's position. This leads to a modular control architecture where the control surfaces are used to track orientation and orientation is used to track position. While the control architecture is primarily developed for an agile fixed-wing aircraft, the majority of the control logic can be applied to many other types of UAVs. We can separate the control logic into two parts: one that is applicable to many types of UAVs, and one that is only relevant to an agile fixed-wing aircraft and would need to be modified if applying this strategy to another platform.

The first part of the controller determines the control moment (in any direction) and force (along the thrust axis) needed to track the reference trajectory of the UAV. This part is modular and contains an outer-loop position controller and force controller, and an inner-loop attitude controller. The second part, 'control allocation', which is more

aircraft-specific, allocates the control moment and force to the individual actuators. This modular structure is shown by the block diagram in Fig. 3.1.

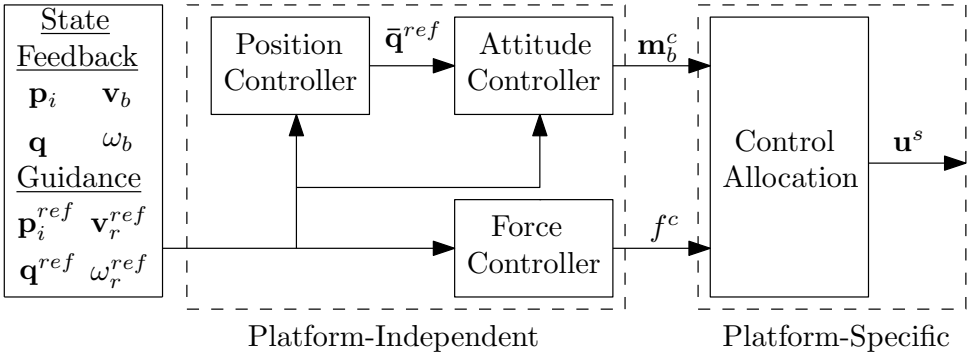


FIGURE 3.1: Control Architecture

We assume a state estimator provides pose and twist estimates of the aircraft, and a motion planning algorithm specifies a reference pose and twist ($\mathbf{p}_i^{ref}$, $\mathbf{q}^{ref}$, $\mathbf{v}_r^{ref}$, & $\boldsymbol{\omega}_r^{ref}$) to be tracked. First, the position controller uses the translational position and velocity errors to modify the reference orientation. This new augmented orientation, $\bar{\mathbf{q}}^{ref}$, is similar to the reference orientation $\mathbf{q}^{ref}$, but modified to correct translational motion errors. The attitude controller then generates control moments that track this augmented orientation. The force control is decoupled from the position and attitude controllers, and its goal is to counteract gravity and the aerodynamic forces, as well as track the height and velocity of the UAV. The control allocation is achieved by determining the forces and moments that are exerted on the UAV by a given set of actuator commands ($\mathbf{u}^s$), and then inverting this relationship to obtain a set of actuator commands that apply the commanded control moment and force.

3.1 Position Controller

The position controller augments the reference attitude of the UAV in order to redirect the body-fixed force to a direction that reduces translational errors. While there are many ways of achieving this, we augment our reference attitude by performing three successive rotations. We formulate these rotations using a newly defined reference frame, $\mathcal{F}_r$, which is fixed to the body frame associated with the reference attitude, $\mathbf{q}^{ref}$. This triad of rotations is computed using the following equation:

$$\boldsymbol{\Theta} = \hat{\mathbf{f}}_r^{ref} \times (K_{p_p}\mathbf{C}_{ri}(\mathbf{p}_i^{ref} - \mathbf{p}_i) + K_{p_d}(\mathbf{v}_r^{ref} - \mathbf{C}_{ri}\mathbf{C}_{bi}^T\mathbf{v}_b)) \tag{3.1}$$

where the direction of body-fixed force associated with the reference orientation is denoted by $\hat{\mathbf{f}}_r^{ref}$, and the PD control law on translational motion are both resolved in $\mathcal{F}_r$. Note that $\hat{\mathbf{f}}_r^{ref}$ and $\hat{\mathbf{f}}_b$ are component-wise equal. The proportional and derivative gains are denoted by K_{p_p} and K_{p_d} respectively. Eq. (3.1) uses the direction cosine matrices from $\mathcal{F}_i$ to $\mathcal{F}_b$, $\mathbf{C}_{bi}$, and $\mathcal{F}_i$ to $\mathcal{F}_r$, $\mathbf{C}_{ri}$. Eq. (2.5) is used to compute $\mathbf{C}_{bi}$, and $\mathbf{C}_{ri}$ can be computed by replacing the attitude, $\mathbf{q}$, with the reference attitude, $\mathbf{q}^{ref}$, in Eq. (2.5).

The gains K_{p_p} and K_{p_d} are chosen to be small enough that typical errors in position and velocity lead to the components of $\frac{1}{2}\boldsymbol{\Theta}$, $(\frac{\Theta_x}{2},\ \frac{\Theta_y}{2},\ \frac{\Theta_z}{2})$, being small angles. In the chance that any component of $\frac{1}{2}\boldsymbol{\Theta}$ becomes large, we limit each component of $\boldsymbol{\Theta}$ to 45° to ensure the half of each component can be considered small. Without this limitation, very large position errors could cause the components of $\boldsymbol{\Theta}$ to become so large that the augmented orientation no longer points the thruster in a direction that reduces errors in position.

We form three quaternion rotations from $\Theta_x,\ \Theta_y$ and Θ_z:

$$\mathbf{q}^x = [\cos\frac{\Theta_x}{2}, \sin\frac{\Theta_x}{2}, 0, 0]^T \tag{3.2a}$$

$$\mathbf{q}^y = [\cos\frac{\Theta_y}{2}, 0, \sin\frac{\Theta_y}{2}, 0]^T \tag{3.2b}$$

$$\mathbf{q}^z = [\cos\frac{\Theta_z}{2}, 0, 0, \sin\frac{\Theta_z}{2}]^T \tag{3.2c}$$

We can then form our augmented reference orientation by performing three successive rotations of the original reference orientation using the Hamilton quaternion product:

$$\bar{\mathbf{q}}^{ref} = \mathbf{q}^{ref} \odot \mathbf{q}^z \odot \mathbf{q}^y \odot\ \mathbf{q}^x \tag{3.3}$$

which can be interpreted as taking the reference orientation, and then subsequently rotating it about the z axis of $\mathcal{F}_r$, and then rotating by intermediary y axis, and then an intermediary x axis. The order of rotations affects the outcome, and thus this order is chosen based on the body frame definitions discussed in Sec. 3.5.

While this augmented reference orientation could be computed in various other ways, such as treating $\boldsymbol{\Theta}$ as an axis-angle rotation, we elect to use three successive rotations in order to keep the augmented reference orientation close to the original reference orientation. The aircraft's roll angle will not affect the thrust direction, and thus ideally the position controller will not affect this portion of the augmented reference attitude. Performing these three successive rotations has a smaller effect on the roll angle than say, treating $\boldsymbol{\Theta}$ as an axis-angle rotation.

Ultimately, these rotations redirect the thrust in order to reduce translational position and velocity error that are not along the thrust axis; the errors along the thrust axis are corrected by the force controller discussed in Sec. 3.2. This approach is generalized to an arbitrary thrust axis, making it suitable for quadrotors, tailsitters, agile aircraft, or even tilt rotors undergoing transitions. In addition, this methodology has no limitations on the actual or reference orientation, and remains valid for *any* orientation of the UAV.

Another advantage to this approach is the modularity of the architecture. The modularity makes it very easy to turn on and off position control (by setting the gains to zero). This is advantageous for gain tuning, as it is easier to first focus on tuning the attitude controller without use of the position controller, and once the attitude tracking is satisfactory the position controller can then be tuned. Another case where the ability to easily turn off the position controller is advantageous is in extreme maneuvering. Say the higher level goal of a maneuver is to perform a flip, and the position of the UAV is irrelevant, one could simply turn off the position controller during the flip to achieve better attitude tracking. The last scenario where the ability to turn off the position controller is advantageous is in a manual/pilot assist mode, where a pilot is the 'position controller' and specifies the augmented reference attitude with a joystick, and then the inner-loop attitude controller can still track this manually generated augmented reference attitude.

3.2 Force Controller

The goal of the force controller is to track the reference height and velocity of the UAV. The control force is chosen such that accelerates the vehicle according to

$$\mathbf{a}_b^{des} = K_v(\mathbf{C}_{bi}\mathbf{C}_{ri}^T\mathbf{v}_r^{ref} - \mathbf{v}_b) + K_{h_p}\Delta\mathbf{h}_b + K_{h_i}\int \Delta\mathbf{h}_b \, dt \tag{3.4}$$

where

$$\Delta\mathbf{h}_b = \mathbf{C}_{bi}[0, 0, (p_{i,z}^{ref} - p_{i,z})]^T \tag{3.5}$$

The desired acceleration is based on a proportional control law on the velocity error, using proportional velocity gain, K_v, and a proportional-integral control law on the height error, with the proportional and integral height gains denoted by K_{h_p} and K_{h_i} respectively.

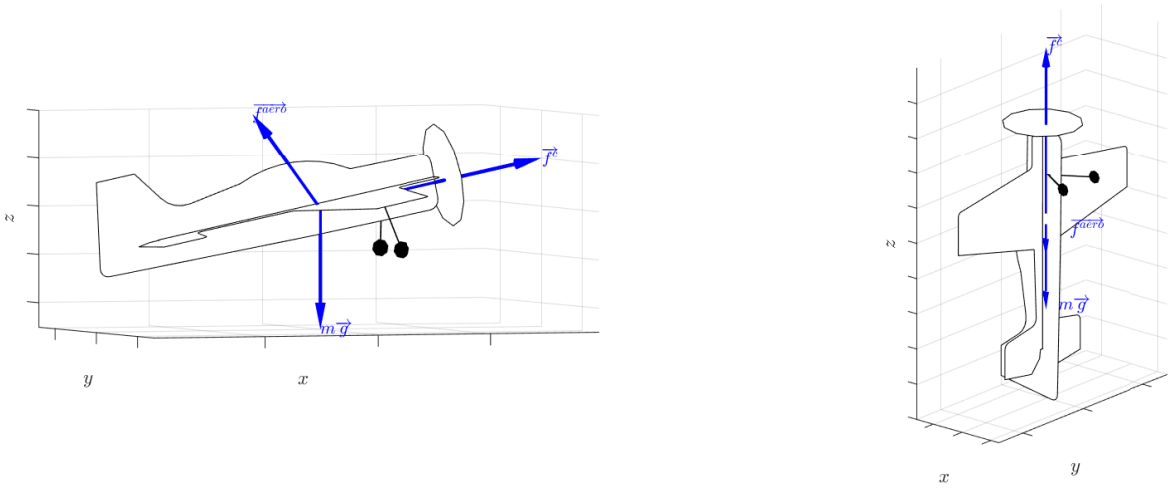
We compute the control force such that feedforward control inputs counteract gravity, lift, and drag, and the remaining force accelerates the vehicle according to our feedback law in Eq. (3.4). The magnitude of the control force, f^c, which is applied along the

body-fixed force axis, is calculated as follows:

$$f^c = (-m\mathbf{C}_{bi}\mathbf{g}_i - \mathbf{f}_b^{aero} + m\mathbf{a}_b^{des})^T \hat{\mathbf{f}}_b \tag{3.6}$$

where the UAV mass is denoted by m, the acceleration due to gravity, expressed in the inertial frame is denoted by $\mathbf{g}_i$, and the aerodynamic force, i.e. the sum of the lift and drag forces, is denoted by $\mathbf{f}_b^{aero}$. Ultimately, the desired total force in the parentheses in Eq. (3.6) would be the desired control force if force could be generated in any direction, however, since the control force can only be generated along the body-fixed force axis, $\hat{\mathbf{f}}_b$, the commanded control force, f^c, is the projection of the desired total force onto this axis.

We show a free body diagram of the aircraft in both level flight and in hover in Fig. 3.2. We can see that in level flight, the aerodynamic force is comprised of lift and drag. When projecting this aerodynamic force on to the direction in which the aircraft can exert thrust, the majority of the projection stems from drag. When the aircraft is hovering, the aircraft generates no lift, but does have a small amount of drag due to the propeller slipstream.



(a) Level Flight (b) Hover

FIGURE 3.2: Aircraft Free Body Diagram

The aerodynamic force is approximated from our simulation. We record the aerodynamic force for various aircraft speeds in steady level flight, as well as in a hover. We then project this force onto $\hat{\mathbf{f}}_b$, the direction in which the aircraft can exert a body-fixed force. We then apply a second-order curve fit to obtain the aerodynamic force as a function of aircraft speed, v, which is shown in Fig. 3.3. In order to account for some discrepancy

between our model and the actual aircraft aerodynamic properties, we scale this curve fit by a gain, K_{aero}, which is tuned in flight.

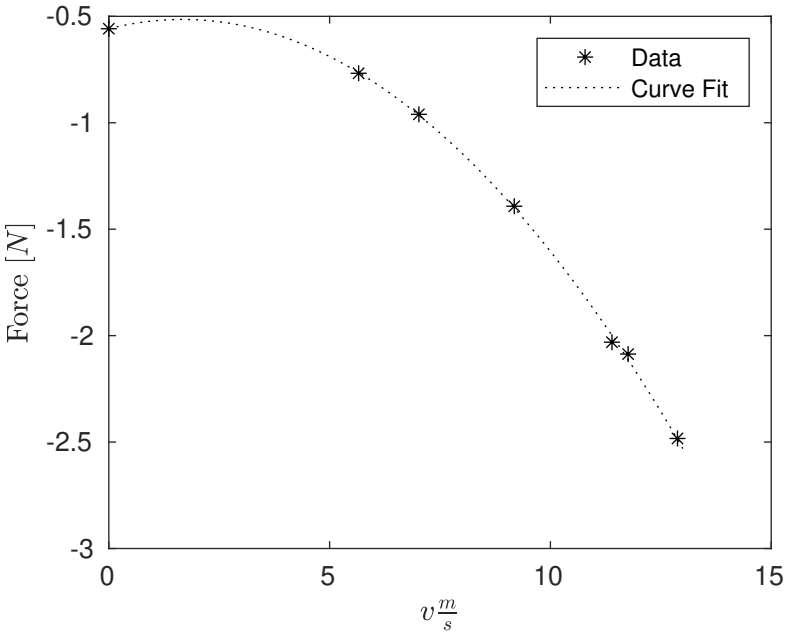


FIGURE 3.3: Aerodynamic Force Approximation. Plot shows the projection of $\mathbf{f}_b^{aero}$ onto $\hat{\mathbf{f}}_b$ and a curve fit of -0.0157v^2 + 0.0524v - 0.5583

We use a simple aerodynamic approximation to generate the feedforward control force, and rely on the robustness of the feedback term to successfully track the aircraft velocity and height. While more sophisticated methods could be used to generate the feedforward control force, such as creating a look-up table based on the aircraft state and the modelling in Chapter 2, we find this simplistic approach sufficient. Furthermore, even the most sophisticated approaches will have significant errors due to the unknown wind conditions and modelling errors.

3.3 Attitude Controller

The goal of the attitude controller is to compute a control moment that will track the augmented reference orientation output from the position controller. The attitude controller computes an error quaternion [23] which is used to obtain angular errors about the body axes, which are in turn, mapped to desired moments.

The error quaternion, $\Delta\mathbf{q}$, describes the rotation from the aircraft attitude, $\mathbf{q}$, to the augmented reference attitude, $\bar{\mathbf{q}}^{ref}$, and thus must satisfy:

$$\mathbf{q} \odot \Delta\mathbf{q} = \bar{\mathbf{q}}^{ref} \tag{3.7}$$

where $\odot$ is the Hamilton quaternion product. The left side corresponds to the aircraft attitude rotated by the rotation describing the actual orientation to augmented reference orientation rotation. This sequence of rotation would equal the augmented reference orientation. By left multiplying by the quaternion inverse, which is just the conjugate for a unit quaternion, $\mathbf{q}^*$, we can obtain the error quaternion:

$$\Delta\mathbf{q} = \mathbf{q}^* \odot \bar{\mathbf{q}}^{ref} \tag{3.8}$$

To ensure the axis-angle parametrization of this rotation has an angle less than 180°, if $\Delta q_0 < 0$, then $\Delta\mathbf{q}$ is replaced with $-\Delta\mathbf{q}$.

The axis-angle representation of the error quaternion can be represented as a rotation of magnitude, β, about a unit vector, $\mathbf{n}$. This axis-angle representation of the error quaternion is equivalent to the angular errors about the body axes, and is computed as:

$$\mathbf{e}_b = \beta\mathbf{n} \tag{3.9}$$

We can also write the error quaternion using the axis-angle representation as:

$$\Delta\mathbf{q} = [\Delta q_0 \ \Delta q_1 \ \Delta q_2 \ \Delta q_3] = [\cos\frac{\beta}{2} \ n_x \sin\frac{\beta}{2} \ n_y \sin\frac{\beta}{2} \ n_z \sin\frac{\beta}{2}] \tag{3.10}$$

By equating the first component on either side of Eq. (3.10) we can obtain the magnitude of rotation from:

$$\beta = 2\cos^{-1}(\Delta q_0) \tag{3.11}$$

We can also equate the other components on the two sides of Eq. (3.10) to solve for the axis of rotation:

$$n_x = \frac{\Delta q_1}{\sin\frac{\beta}{2}}, \ n_y = \frac{\Delta q_2}{\sin\frac{\beta}{2}}, \ n_z = \frac{\Delta q_3}{\sin\frac{\beta}{2}} \tag{3.12}$$

Using a trigonometric identity, Eq. (3.11), and the knowledge that we are using a unit quaternion, we can solve for $\sin\frac{\beta}{2}$:

$$\sin\frac{\beta}{2} = \sqrt{1-\cos^2\frac{\beta}{2}} = \sqrt{1-\Delta q_0^2} = \sqrt{\Delta q_1^2 + \Delta q_2^2 + \Delta q_3^2} \tag{3.13}$$

Finally, we can substitute Eq. (3.13) into Eq. (3.12), and substitute Eq. (3.11) & Eq. (3.12) into Eq. (3.9) to obtain the angular errors about the body frame axes. Combining these equations, we calculate the angular errors about the body frame axes, $\mathbf{e}_b$, using:

$$\mathbf{e}_b = \begin{cases} 2\cos^{-1}(\Delta q_0)\frac{\Delta\mathbf{q}_{1:3}}{||\Delta\mathbf{q}_{1:3}||}, & ||\Delta\mathbf{q}_{1:3}|| \neq 0 \\ \mathbf{0}, & ||\Delta\mathbf{q}_{1:3}|| = 0 \end{cases} \tag{3.14}$$

where $\Delta\mathbf{q}_{1:3}$ refers to the vector part of the error quaternion, and Δq_0 refers to the scalar component. If $||\Delta\mathbf{q}_{1:3}|| = 0$ the error quaternion is [1 0 0 0], the identity quaternion, implying there is no angular error, and $\mathbf{e}_b$ is set to $\mathbf{0}$.

A PD controller is used to calculate the moment needed to correct the angular error. Two gain matrices, $\mathbf{K_{a_p}}$ and $\mathbf{K_{a_d}}$, are used to map an angular error to a desired angular acceleration, which is multiplied by the inertia matrix to obtain a desired control moment about each axis. This multiplication by the inertia matrix could be removed and factored into the control gains, but it allows for easier transitioning to different platforms while using a similar set of control gains. The control moment is calculated by:

$$\mathbf{m}_b^c = \mathbf{I}_b(\mathbf{K_{a_p}}\mathbf{e}_b + \mathbf{K_{a_d}}(\mathbf{C}_{bi}\mathbf{C}_{ri}^T\boldsymbol{\omega}_r^{ref} - \boldsymbol{\omega}_b)) + \mu(\boldsymbol{\omega}_b \times \mathbf{I}_b\boldsymbol{\omega}_b) \tag{3.15}$$

where $\mathbf{m}_b^c$ is the control moment to be applied by the aircraft's actuators. The second term of Eq. (3.15) can precisely cancel the gyroscopic coupling torque by setting $\mu = 1$. However, for the maneuvers demonstrated in simulation, we found there is little difference whether $\mu = 0$ or $\mu = 1$ due to the relatively small inertia tensor and angular velocity values. For this reason, we keep $\mu = 0$, which makes the controller less sensitive to inaccuracies in the estimation of the inertia tensor and angular velocity.

A stability analysis of the attitude controller is shown in Sec. 3.4.1, and a similar attitude controller presented in [16], is shown to be globally stable.

3.4 Stability Analysis

Since a UAV has four control inputs, it is only possible to achieve asymptotic tracking for at most four output states. This leads to most researchers developing controllers for either position tracking flight modes (and one attitude state), or attitude tracking flight modes (and one position state) [21]. However, aerobatic maneuvering requires control over all six states, since the desired maneuver is a trajectory in both attitude and position, and thus foregoing the asymptotic tracking of four states for the ability to have some control over all six states is beneficial. For these reasons, we develop a controller that tries to track all six states, where the controller constantly trades off between position and attitude tracking. While it is not possible to achieve asymptotic tracking of all six states, we are able to demonstrate the attitude controller is asymptotically stable with regards to regulation of the *augmented* reference orientation, which is the original reference orientation, but modified to control position. For the position error stability analysis we assume the aircraft attitude is the augmented reference attitude. This assumption is based on the fast rotational and slow translational dynamics of aircraft [91]. Under this assumption we show the position errors are Lyapunov or asymptotically stable, depending on the reference orientation.

3.4.1 Attitude

We recall the attitude kinematics as

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{q}_0 \\ \dot{\mathbf{q}}_{1:3} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} q_0 \\ \mathbf{q}_{1:3} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega}_b \end{bmatrix}. \tag{3.16}$$

and the error quaternion defined as

$$\Delta\mathbf{q} = \mathbf{q}^* \odot \bar{\mathbf{q}}^{ref}. \tag{3.17}$$

We can obtain our error quaternion kinematics by taking the time derivative of both sides of the equation

$$\Delta\dot{\mathbf{q}} = \dot{\mathbf{q}}^* \odot \bar{\mathbf{q}}^{ref} + \mathbf{q}^* \odot \dot{\bar{\mathbf{q}}}^{ref}. \tag{3.18}$$

We show the stability analysis for regulation, so $\dot{\bar{\mathbf{q}}}^{ref} = \mathbf{0}$. By simplifying and substituting Eq. (3.16) into Eq. (3.18) we obtain

$$\Delta\dot{\mathbf{q}} = \frac{1}{2}\left(\begin{bmatrix} q_0 \\ \mathbf{q}_{1:3} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega}_b \end{bmatrix}\right)^* \odot \bar{\mathbf{q}}^{ref}. \tag{3.19}$$

Using the property for two quaternions $\mathbf{a}$ and $\mathbf{b}$, $(\mathbf{a} \odot \mathbf{b})^{-1} = \mathbf{b}^{-1} \odot \mathbf{a}^{-1}$, and for unit quaternions, $(\mathbf{a} \odot \mathbf{b})^* = \mathbf{b}^* \odot \mathbf{a}^*$, Eq. (3.19) becomes

$$\Delta\dot{\mathbf{q}} = \frac{1}{2}\begin{bmatrix} 0 \\ -\boldsymbol{\omega}_b \end{bmatrix} \odot \begin{bmatrix} q_0 \\ \mathbf{q}_{1:3} \end{bmatrix}^* \odot \bar{\mathbf{q}}^{ref}. \tag{3.20}$$

Recalling our error quaternion definition from Eq. (3.17), Eq. (3.20) becomes

$$\Delta\dot{\mathbf{q}} = -\frac{1}{2}\begin{bmatrix} 0 \\ \boldsymbol{\omega}_b \end{bmatrix} \odot \Delta\mathbf{q}, \tag{3.21}$$

and by multiplying through becomes

$$\Delta\dot{\mathbf{q}} = \begin{bmatrix} \Delta\dot{q}_0 \\ \Delta\dot{\mathbf{q}}_{1:3} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} \boldsymbol{\omega}_b^T \Delta\mathbf{q}_{1:3} \\ -\Delta q_0 \boldsymbol{\omega}_b - \boldsymbol{\omega}_b \times \Delta\mathbf{q}_{1:3} \end{bmatrix}. \tag{3.22}$$

Turning our attention to the attitude dynamics, we recall the rotational equations of motion subject to no disturbances, as

$$\mathbf{I}_b\dot{\boldsymbol{\omega}}_b = \mathbf{m}_b^c - \boldsymbol{\omega}_b \times \mathbf{I}_b\boldsymbol{\omega}_b \tag{3.23}$$

where our control torque $\mathbf{m}_b^c$ comes from Eqs. (3.14 & 3.15). For the regulation task, $\boldsymbol{\omega}_r^{ref} = \mathbf{0}$, and the control torque becomes

$$\mathbf{m}_b^c = \mathbf{I}_b(\mathbf{K}_{\mathbf{a_p}} 2\cos^{-1}(\Delta q_0)\frac{\Delta\mathbf{q}_{1:3}}{||\Delta\mathbf{q}_{1:3}||} - \mathbf{K}_{\mathbf{a_d}}\boldsymbol{\omega}_b) + \mu(\boldsymbol{\omega}_b \times \mathbf{I}_b\boldsymbol{\omega}_b). \tag{3.24}$$

We can replace $||\Delta\mathbf{q}_{1:3}||$ with $\sqrt{1-\Delta q_0^2}$ in Eq. (3.24) since $\Delta\mathbf{q}$ is a unit quaternion, which can be substituted into Eq. (3.23) to obtain the closed-loop attitude dynamics

$$\mathbf{I}_b\dot{\boldsymbol{\omega}}_b = \mathbf{I}_b(\mathbf{K}_{\mathbf{a_p}} 2\cos^{-1}(\Delta q_0)\frac{\Delta\mathbf{q}_{1:3}}{\sqrt{1-\Delta q_0^2}} - \mathbf{K}_{\mathbf{a_d}}\boldsymbol{\omega}_b) + (\mu-1)(\boldsymbol{\omega}_b \times \mathbf{I}_b\boldsymbol{\omega}_b). \tag{3.25}$$

Now consider the following Lyapunov function candidate

$$V = \frac{1}{2}\boldsymbol{\omega}_b^T(\mathbf{I}_b\mathbf{K}_{\mathbf{a_p}})^{-1}\mathbf{I}_b\boldsymbol{\omega}_b + 2[\cos^{-1}(\Delta q_0)]^2 \tag{3.26}$$

At the equilibrium point, $\Delta\mathbf{q} = [1\ 0\ 0\ 0]^T$ and $\boldsymbol{\omega}_b = \mathbf{0}$, $V = 0$. We require $\mathbf{K}_{\mathbf{a_p}} > 0$, which makes $V > 0$ for all error quaternion and angular velocity combinations besides the equilibrium point. We can then take the time derivative of the Lyapunov function candidate to obtain

$$\dot{V} = \boldsymbol{\omega}_b^T(\mathbf{I}_b\mathbf{K}_{\mathbf{a_p}})^{-1}\mathbf{I}_b\dot{\boldsymbol{\omega}}_b - \frac{4\cos^{-1}(\Delta q_0)}{\sqrt{1-\Delta q_0^2}}\Delta\dot{q}_0. \tag{3.27}$$

We can substitute Eqs. (3.22 & 3.25) into Eq. (3.27) to obtain

$$\dot{V} = \boldsymbol{\omega}_b^T(\mathbf{I}_b\mathbf{K}_{\mathbf{a_p}})^{-1}(\mathbf{I}_b(\mathbf{K}_{\mathbf{a_p}}2\cos^{-1}(\Delta q_0)\frac{\Delta\mathbf{q}_{1:3}}{\sqrt{1-\Delta q_0^2}} - \mathbf{K}_{\mathbf{a_d}}\boldsymbol{\omega}_b) + (\mu-1)(\boldsymbol{\omega}_b \times \mathbf{I}_b\boldsymbol{\omega}_b)) - \frac{4\cos^{-1}(\Delta q_0)}{\sqrt{1-\Delta q_0^2}}(\frac{1}{2}\boldsymbol{\omega}_b^T\Delta\mathbf{q}_{1:3}), \tag{3.28}$$

which simplifies to

$$\dot{V} = -\boldsymbol{\omega}_b^T(\mathbf{I}_b\mathbf{K}_{\mathbf{a_p}})^{-1}\mathbf{I}_b\mathbf{K}_{\mathbf{a_d}}\boldsymbol{\omega}_b + \boldsymbol{\omega}_b^T(\mathbf{I}_b\mathbf{K}_{\mathbf{a_p}})^{-1}(\mu-1)(\boldsymbol{\omega}_b \times \mathbf{I}_b\boldsymbol{\omega}_b) + \boldsymbol{\omega}_b^T(\mathbf{I}_b\mathbf{K}_{\mathbf{a_p}})^{-1}(\mathbf{I}_b\mathbf{K}_{\mathbf{a_p}})2\cos^{-1}(\Delta q_0)\frac{\Delta\mathbf{q}_{1:3}}{\sqrt{1-\Delta q_0^2}} - \frac{4\cos^{-1}(\Delta q_0)}{\sqrt{1-\Delta q_0^2}}(\frac{1}{2}\boldsymbol{\omega}_b^T\Delta\mathbf{q}_{1:3}), \tag{3.29}$$

which simplifies to

$$\dot{V} = -\boldsymbol{\omega}_b^T{\mathbf{K}_{\mathbf{a_p}}}^{-1}\mathbf{I}_b^{-1}\mathbf{I}_b\mathbf{K}_{\mathbf{a_d}}\boldsymbol{\omega}_b + \boldsymbol{\omega}_b^T(\mathbf{I}_b\mathbf{K}_{\mathbf{a_p}})^{-1}(\mu-1)(\boldsymbol{\omega}_b \times \mathbf{I}_b\boldsymbol{\omega}_b) + \frac{2\cos^{-1}(\Delta q_0)}{\sqrt{1-\Delta q_0^2}}(\boldsymbol{\omega}_b^T\Delta\mathbf{q}_{1:3} - \boldsymbol{\omega}_b^T\Delta\mathbf{q}_{1:3}). \tag{3.30}$$

If we set $\mu = 1$ we get a precise cancellation of the gyroscopic coupling torque. Alternatively, this term could go to zero by removing the multiplication of $\mathbf{I}_b$ in Eq. 3.24, and forcing ${\mathbf{K}_{\mathbf{a_p}}}^{-1}$ to be a linear combination of $\mathbf{I}_b$ and the identity matrix, which is shown in [16]. We can then further simplify our Lyapunov function derivative to

$$\dot{V} = -\boldsymbol{\omega}_b^T{\mathbf{K}_{\mathbf{a_p}}}^{-1}\mathbf{K}_{\mathbf{a_d}}\boldsymbol{\omega}_b. \tag{3.31}$$

As we can see in Eq. (3.31), if we require $\mathbf{K_{a_p}}^{-1}\mathbf{K_{a_d}} > 0$ our Lyapunov function derivative is negative semi-definite, showing the attitude errors are globally stable. We can show these errors are globally asymptotically stable using Lasalle's invariant set theorem.

We solve for the error states when $\dot{V} = 0$:

$$\dot{V} = -\boldsymbol{\omega}_b^T \mathbf{K_{a_p}}^{-1}\mathbf{K_{a_d}}\boldsymbol{\omega}_b = 0 \Rightarrow \boldsymbol{\omega}_b = \mathbf{0} \Rightarrow \dot{\boldsymbol{\omega}}_b = \mathbf{0}$$

Since $\dot{V} = 0$ implies $\boldsymbol{\omega}_b = \mathbf{0}$ and $\dot{\boldsymbol{\omega}}_b = \mathbf{0}$, then when $\dot{V} = 0$ the closed-loop attitude dynamics, represented by Eq. (3.25), simplifies to

$$\mathbf{0} = \mathbf{I}_b \mathbf{K_{a_p}} 2\cos^{-1}(\Delta q_0)\frac{\Delta \mathbf{q}_{1:3}}{\sqrt{1-\Delta q_0^2}}. \tag{3.32}$$

Since $\mathbf{I}_b > 0$ and $\mathbf{K_{a_p}} > 0$, for this equation to hold $\Delta\mathbf{q} = [1\ 0\ 0\ 0]^T$.

Applying Lasalle's invariance principle, since $\dot{V} \leq 0$ for all $\Delta\mathbf{q}, \boldsymbol{\omega}_b$, and the only solution to $\dot{V} = 0$ exists when $\Delta\mathbf{q} = [1\ 0\ 0\ 0]^T$ and $\boldsymbol{\omega}_b = \mathbf{0}$, then we can conclude $\Delta\mathbf{q} = [1\ 0\ 0\ 0]^T$ and $\boldsymbol{\omega}_b = \mathbf{0}$ is an asymptotically stable equilibrium point. We can combine this with our Lyapunov analysis to conclude the attitude errors are globally asymptotically stable.

3.4.2 Position

We now turn over attention to the stability analysis of position and velocity errors. First we define our position error, resolved in the inertial frame, as

$$\Delta\mathbf{p}_i = \mathbf{p}_i^{ref} - \mathbf{p}_i \tag{3.33}$$

and our velocity error, also resolved in the inertial frame, as

$$\Delta\mathbf{v}_i = \Delta\dot{\mathbf{p}}_i = \dot{\mathbf{p}}_i^{ref} - \dot{\mathbf{p}}_i = \mathbf{v}_i^{ref} - \mathbf{v}_i. \tag{3.34}$$

For our stability analysis we consider constant velocity trajectories, causing the velocity error to propagate according to

$$\Delta\dot{\mathbf{v}}_i = -\dot{\mathbf{v}}_i = -\mathbf{g}_i - \frac{\mathbf{f}_i^{aero}}{m} - \frac{f^c}{m}\hat{\mathbf{f}}_i. \tag{3.35}$$

The direction, resolved in the inertial frame, in which the aircraft can exert thrust, $\hat{\mathbf{f}}_i$, will depend on the aircraft's orientation as shown in

$$\hat{\mathbf{f}}_i = \mathbf{C}_{bi}^T \hat{\mathbf{f}}_b. \tag{3.36}$$

Since aircraft have fast rotational and slow translational dynamics [91], in the position stability analysis we assume the aircraft attitude, represented by both $\mathbf{q}$ and $\mathbf{C}_{bi}$, is the attitude output from the position controller, the augmented reference attitude, which is represented by both $\bar{\mathbf{q}}^{ref}$ and $\mathbf{C}_{\bar{r}i}$. This assumption causes Eq. (3.36) to become

$$\hat{\mathbf{f}}_i = \mathbf{C}_{\bar{r}i}^T \hat{\mathbf{f}}_b. \tag{3.37}$$

We utilize Eqs. (3.1 - 3.3) to compute $\bar{\mathbf{q}}^{ref}$. We can substitute our position and velocity error definitions in Eqs. (3.33 & 3.34) into Eq. (3.1) to obtain

$$\boldsymbol{\Theta} = \hat{\mathbf{f}}_r^{ref} \times \mathbf{C}_{ri}(K_{p_p}\Delta\mathbf{p}_i + K_{p_d}\Delta\mathbf{v}_i). \tag{3.38}$$

As mentioned in Sec. 3.1, the gains K_{p_p} and K_{p_d} are chosen to be small enough that typical errors in position and velocity lead to $\frac{\Theta_x}{2}$, $\frac{\Theta_y}{2}$, and $\frac{\Theta_z}{2}$ being small. Using this small angle assumption allows Eq. (3.2a-3.2c) to be simplified to

$$\mathbf{q}^x = [1, \frac{\Theta_x}{2}, 0, 0]^T \tag{3.39a}$$

$$\mathbf{q}^y = [1, 0, \frac{\Theta_y}{2}, 0]^T \tag{3.39b}$$

$$\mathbf{q}^z = [1, 0, 0, \frac{\Theta_z}{2}]^T \tag{3.39c}$$

We can substitute Eq. (3.38) into Eqs. (3.39a - 3.39c), and substitute Eqs. (3.39a - 3.39c) into Eq. (3.3) to obtain $\bar{\mathbf{q}}^{ref}$, and can then obtain $\mathbf{C}_{\bar{r}i}$ using Eq. (2.5).

For the remainder of the proof, we must select a reference orientation for the aircraft. We proceed with three different reference orientations corresponding to level flight, hover, and knife-edge maneuvers.

3.4.2.1 Level Flight

By using a reference orientation of 0° roll, 0° pitch, and 0° yaw, we can obtain the direction of the thrust in the inertial frame as

$$\hat{\mathbf{f}}_i = \begin{bmatrix} \frac{\left({K_{p_d}}^2 \Delta v_y^2 + 2\, K_{p_d} K_{p_p} \Delta p_y \Delta v_y + {K_{p_p}}^2 \Delta p_y^2 - 4\right)\left({K_{p_d}}^2 \Delta v_z^2 + 2\, K_{p_d} K_{p_p} \Delta p_z \Delta v_z + {K_{p_p}}^2 \Delta p_z^2 - 4\right)}{\left({K_{p_d}}^2 \Delta v_y^2 + 2\, K_{p_d} K_{p_p} \Delta p_y \Delta v_y + {K_{p_p}}^2 \Delta p_y^2 + 4\right)\left({K_{p_d}}^2 \Delta v_z^2 + 2\, K_{p_d} K_{p_p} \Delta p_z \Delta v_z + {K_{p_p}}^2 \Delta p_z^2 + 4\right)} \\ -\frac{4\left(K_{p_p} \Delta p_y + K_{p_d} \Delta v_y\right)\left({K_{p_d}}^2 \Delta v_z^2 + 2\, K_{p_d} K_{p_p} \Delta p_z \Delta v_z + {K_{p_p}}^2 \Delta p_z^2 - 4\right)}{\left({K_{p_d}}^2 \Delta v_y^2 + 2\, K_{p_d} K_{p_p} \Delta p_y \Delta v_y + {K_{p_p}}^2 \Delta p_y^2 + 4\right)\left({K_{p_d}}^2 \Delta v_z^2 + 2\, K_{p_d} K_{p_p} \Delta p_z \Delta v_z + {K_{p_p}}^2 \Delta p_z^2 + 4\right)} \\ \frac{4\left(K_{p_p} \Delta p_z + K_{p_d} \Delta v_z\right)}{{K_{p_d}}^2 \Delta v_z^2 + 2\, K_{p_d} K_{p_p} \Delta p_z \Delta v_z + {K_{p_p}}^2 \Delta p_z^2 + 4} \end{bmatrix}. \tag{3.40}$$

By neglecting higher order terms because of the small K_{p_p} and K_{p_d} assumption, Eq. 3.40 simplifies to

$$\hat{\mathbf{f}}_i = \begin{bmatrix} 1 \\ K_{p_p}\Delta p_y + K_{p_d}\Delta v_y \\ K_{p_p}\Delta p_z + K_{p_d}\Delta v_z \end{bmatrix}, \tag{3.41}$$

which can be substituted into Eq. (3.35) to obtain the velocity error dynamics

$$\Delta\dot{\mathbf{v}}_i = \begin{bmatrix} -\frac{f^c + f_x^{aero}}{m} \\ -\frac{f^c}{m}(K_{p_p}\Delta p_y + K_{p_d}\Delta v_y) \\ -g - \frac{f_z^{aero}}{m} - \frac{f^c}{m}(K_{p_p}\Delta p_z + K_{p_d}\Delta v_z) \end{bmatrix}. \tag{3.42}$$

As we can see, aerodynamic force only acts in the x (drag) and z (lift) directions, because of the 0° roll, 0° pitch, and 0° yaw reference orientation. We assume the reference orientations are chosen such that the lift and weight forces cancel out. The control force, f^c, can be obtained from Eqs. (3.4 & 3.6), which simplifies to $f^c = K_v \Delta v_x - f_x^{aero}$ with this reference orientation and neglecting higher-order terms because of the small K_{p_p} and K_{p_d} assumption. We also must realize the control force is the magnitude of an actuator output (usually a propeller) which has a physical limitation, denoted by $f^{c,max}$. This bounds f^c: $0 \leq f^c \leq f^{c,max}$. We can further simplify our velocity error dynamics to

$$\Delta\dot{\mathbf{v}}_i = \begin{bmatrix} -\frac{K_v \Delta v_x}{m} \\ -\frac{f^c}{m}(K_{p_p}\Delta p_y + K_{p_d}\Delta v_y) \\ -\frac{f^c}{m}(K_{p_p}\Delta p_z + K_{p_d}\Delta v_z) \end{bmatrix}. \tag{3.43}$$

By not substituting the control law for f^c into the y and z components, and allowing it to be any value such that $0 \leq f^c \leq f^{c,max}$, the closed-loop translational dynamics become linear, and can be written in the form

$$\begin{bmatrix} \Delta\dot{\mathbf{p}}_i \\ \Delta\dot{\mathbf{v}}_i \end{bmatrix} = \mathbf{A} \begin{bmatrix} \Delta\mathbf{p}_i \\ \Delta\mathbf{v}_i \end{bmatrix}, \tag{3.44}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{K_v}{m} & 0 & 0 \\ 0 & -\frac{K_{p_p} f^c}{m} & 0 & 0 & -\frac{K_{p_d} f^c}{m} & 0 \\ 0 & 0 & -\frac{K_{p_p} f^c}{m} & 0 & 0 & -\frac{K_{p_d} f^c}{m} \end{bmatrix}. \tag{3.45}$$

We can compute the closed-loop eigenvalues of $\mathbf{A}$ to obtain

$$\lambda_1 = 0 \tag{3.46}$$

$$\lambda_2 = \frac{-K_v}{m} \tag{3.47}$$

$$\lambda_{3,4} = \frac{-K_{p_d} f^c \pm \sqrt{-f^c(4K_{p_p} m - K_{p_d}^2 f^c)}}{2m} \tag{3.48}$$

$$\lambda_{5,6} = \frac{-K_{p_d} f^c \pm \sqrt{-f^c(4K_{p_p} m - K_{p_d}^2 f^c)}}{2m} \tag{3.49}$$

If $-f^c(4K_{p_p} m - K_{p_d}^2 f^c) \leq 0$, then the real part of all eigenvalues of the closed-loop $\mathbf{A}$ matrix are less than or equal to zero. Since $f^c \geq 0$, we need $4K_{p_p} m - K_{p_d}^2 f^c \geq 0$. This leads to $\frac{4K_{p_p} m}{K_{p_d}^2} \geq f^c$, which can be guaranteed if the control gains are chosen such that $\frac{4K_{p_p} m}{K_{p_d}^2} \geq f^{c,max}$. Assuming the gains meet this criterion, the closed-loop system is Lyapunov stable. We could achieve asymptotic stability by including Δp_x and Δp_y in the force controller, but we leave out these terms so the aircraft can achieve path following as opposed to position tracking.

3.4.2.2 Hover

We can go through the same steps in hovering flight as done in level flight. For hover we specify a reference orientation of 0° roll, 90° pitch, and 0° yaw. We can obtain the direction of the thrust in the inertial frame as

$$\begin{bmatrix} \frac{4\,K_{p_p}\,\Delta p_x+4\,K_{p_d}\,\Delta v_x}{{K_{p_d}}^2\,{\Delta v_x}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_x\,\Delta v_x+{K_{p_p}}^2\,{\Delta p_x}^2+4} \\ -\frac{4\left(K_{p_p}\,\Delta p_y+K_{p_d}\,\Delta v_y\right)\left({K_{p_d}}^2\,{\Delta v_x}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_x\,\Delta v_x+{K_{p_p}}^2\,{\Delta p_x}^2-4\right)}{\left({K_{p_d}}^2\,{\Delta v_x}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_x\,\Delta v_x+{K_{p_p}}^2\,{\Delta p_x}^2+4\right)\left({K_{p_d}}^2\,{\Delta v_y}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_y\,\Delta v_y+{K_{p_p}}^2\,{\Delta p_y}^2+4\right)} \\ -\frac{\left({K_{p_d}}^2\,{\Delta v_x}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_x\,\Delta v_x+{K_{p_p}}^2\,{\Delta p_x}^2-4\right)\left({K_{p_d}}^2\,{\Delta v_y}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_y\,\Delta v_y+{K_{p_p}}^2\,{\Delta p_y}^2-4\right)}{\left({K_{p_d}}^2\,{\Delta v_x}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_x\,\Delta v_x+{K_{p_p}}^2\,{\Delta p_x}^2+4\right)\left({K_{p_d}}^2\,{\Delta v_y}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_y\,\Delta v_y+{K_{p_p}}^2\,{\Delta p_y}^2+4\right)} \end{bmatrix} \tag{3.50}$$

By neglecting higher order terms because of the small K_{p_p} and K_{p_d} assumption, Eq. 3.50 simplifies to

$$\hat{\mathbf{f}}_i = \begin{bmatrix} K_{p_p}\Delta p_x + K_{p_d}\Delta v_x \\ K_{p_p}\Delta p_y + K_{p_d}\Delta v_y \\ -1 \end{bmatrix}, \tag{3.51}$$

which can be substituted into Eq. (3.35) to obtain the velocity error dynamics

$$\Delta\dot{\mathbf{v}}_i = \begin{bmatrix} -\frac{f^c}{m}(K_{p_p}\Delta p_x + K_{p_d}\Delta v_x) \\ -\frac{f^c}{m}(K_{p_p}\Delta p_y + K_{p_d}\Delta v_y) \\ \frac{f^c}{m} - g \end{bmatrix}. \tag{3.52}$$

As we can see, there is no aerodynamic force (lift and drag) while hovering, because the aircraft is stationary. As done previously, the control force can be obtained from Eqs. (3.4 & 3.6), which simplifies to $f^c = mg - K_{h_p}\Delta p_z - K_v\Delta v_z$ with this hover reference orientation and neglecting higher-order terms because of the small K_{p_p} and K_{p_d} assumption. We can further simplify our velocity error dynamics to

$$\Delta\dot{\mathbf{v}}_i = \begin{bmatrix} -\frac{f^c}{m}(K_{p_p}\Delta p_x + K_{p_d}\Delta v_x) \\ -\frac{f^c}{m}(K_{p_p}\Delta p_y + K_{p_d}\Delta v_y) \\ -\frac{1}{m}(K_{h_p}\Delta p_z + K_v\Delta v_z) \end{bmatrix}. \tag{3.53}$$

Similar to the level flight analysis, by not substituting the control law for f^c into the x and y components, and allowing it to be any value such that $0 \leq f^c \leq f^{c,max}$, the closed-loop translational dynamics become linear, and can be written in the form

$$\begin{bmatrix} \Delta\dot{\mathbf{p}}_i \\ \Delta\dot{\mathbf{v}}_i \end{bmatrix} = \mathbf{A} \begin{bmatrix} \Delta\mathbf{p}_i \\ \Delta\mathbf{v}_i \end{bmatrix} \tag{3.54}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -\frac{K_{p_p} f^c}{m} & 0 & 0 & -\frac{K_{p_d} f^c}{m} & 0 & 0 \\ 0 & -\frac{K_{p_p} f^c}{m} & 0 & 0 & -\frac{K_{p_d} f^c}{m} & 0 \\ 0 & 0 & -\frac{K_{h_p}}{m} & 0 & 0 & -\frac{K_v}{m} \end{bmatrix} \tag{3.55}$$

We can compute the closed-loop eigenvalues of $\mathbf{A}$ to obtain

$$\lambda_{1,2} = \frac{-K_v f^c \pm \sqrt{K_v^2 - 4K_{h_p} m}}{2m} \tag{3.56}$$

$$\lambda_{3,4} = \frac{-K_{p_d} f^c \pm \sqrt{-f^c(4K_{p_p} m - K_{p_d}^2 f^c)}}{2m} \tag{3.57}$$

$$\lambda_{5,6} = \frac{-K_{p_d} f^c \pm \sqrt{-f^c(4K_{p_p} m - K_{p_d}^2 f^c)}}{2m} \tag{3.58}$$

Following the same analysis as in the level flight case, we require $\frac{4K_{p_p} m}{K_{p_d}^2} \geq f^{c,max}$. In addition, $K_v^2 - 4K_{h_p} m \leq 0$, which implies $\frac{K_v^2}{4m} \leq K_{h_p}$. Thus all closed-loop eigenvalues are less than zero, proving the system is asymptotically stable in hover conditions.

3.4.2.3 Knife-Edge

Similarly to the level flight and hover analysis, we can obtain the direction of the thrust in the inertial frame for the knife-edge maneuver using a reference orientation of 90° roll, 0° pitch, and 0° yaw:

$$\begin{bmatrix} \frac{\left({K_{p_d}}^2\,{\Delta v_y}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_y\,\Delta v_y+{K_{p_p}}^2\,{\Delta p_y}^2-4\right)\left({K_{p_d}}^2\,{\Delta v_z}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_z\,\Delta v_z+{K_{p_p}}^2\,{\Delta p_z}^2-4\right)}{\left({K_{p_d}}^2\,{\Delta v_y}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_y\,\Delta v_y+{K_{p_p}}^2\,{\Delta p_y}^2+4\right)\left({K_{p_d}}^2\,{\Delta v_z}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_z\,\Delta v_z+{K_{p_p}}^2\,{\Delta p_z}^2+4\right)} \\ \frac{4\,K_{p_p}\,\Delta p_y+4\,K_{p_d}\,\Delta v_y}{{K_{p_d}}^2\,{\Delta v_y}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_y\,\Delta v_y+{K_{p_p}}^2\,{\Delta p_y}^2+4} \\ -\frac{4\left(K_{p_p}\,\Delta p_z+K_{p_d}\,\Delta v_z\right)\left({K_{p_d}}^2\,{\Delta v_y}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_y\,\Delta v_y+{K_{p_p}}^2\,{\Delta p_y}^2-4\right)}{\left({K_{p_d}}^2\,{\Delta v_y}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_y\,\Delta v_y+{K_{p_p}}^2\,{\Delta p_y}^2+4\right)\left({K_{p_d}}^2\,{\Delta v_z}^2+2\,K_{p_d}\,K_{p_p}\,\Delta p_z\,\Delta v_z+{K_{p_p}}^2\,{\Delta p_z}^2+4\right)} \end{bmatrix} \tag{3.59}$$

By neglecting higher order terms because of the small K_{p_p} and K_{p_d} assumption, Eq. 3.59 simplifies to

$$\hat{\mathbf{f}}_i = \begin{bmatrix} 1 \\ K_{p_p}\Delta p_y + K_{p_d}\Delta v_y \\ K_{p_p}\Delta p_z + K_{p_d}\Delta v_z \end{bmatrix} \tag{3.60}$$

which is identical to the level flight case. Carrying on with the rest of this proof is identical to the level flight analysis, which concludes the position and velocity errors are Lyapunov stable.

3.4.3 Stability Analysis Remarks

We are able to provide a stability analysis for steady-state trajectories with constant orientation and velocity. While our mathematical analysis is limited to these types of trajectories, we are able to demonstrate successful tracking of time-varying attitude and velocity trajectories in the simulation and flight testing discussed in both Chapters 4 & 6.

3.5 Control Allocation

We now need to map the control moment and body-fixed force to individual actuator commands. The effects of the actuators are modelled in the high-fidelity simulation described in Chapter 2. However, using the high-fidelity model at each control time-step would be too complex to implement on flight hardware. Therefore, we use simplified physics-based models of how the thruster and control surfaces generate forces and moments in order to determine the appropriate control surface deflections and motor speed. In the following section, we outline these simplified models, which will be used in inverse form to determine the individual actuator commands.

An agile fixed-wing aircraft has two types of actuators: a thruster (motor/propeller), and control surfaces.

3.5.1 Actuators

We denote the commanded actuator signal with u_j^s for the system's j^{th} actuator, and this signal corresponds to a force and torque, denoted by $\mathbf{u}_j^f$ and $\mathbf{u}_j^\tau$, which are resolved in the body frame.

3.5.1.1 Thruster

For a propeller, we consider the input signal, u_j^s, to be the rotational speed of the j^{th} propeller in RPM, and this can be mapped to a thrust and torque as follows:

$$\mathbf{u}_j^f = k_{t_j}(J_j)u_j^{s^2}\hat{\mathbf{f}}_b \tag{3.61a}$$

$$\mathbf{u}_j^\tau = \pm k_{q_j}(J_j)u_j^{s^2}\hat{\mathbf{f}}_b \tag{3.61b}$$

where the propeller thrust and torque coefficients are denoted by k_{t_j} and k_{q_j}. Simplified thrust and torque models assume k_{t_j} and k_{q_j} are constant, which is valid for stationary propellers (i.e. hovering flight). However, an agile fixed-wing flies at high speeds, and thus the propeller is not stationary. As the aircraft flies faster the same propeller rotational speed will produce less force, as the difference in airflow velocity entering and leaving the propeller lessens. We account for this phenomenon by modeling the thrust and torque coefficients as a function of advance ratio, J_j, using the model presented in [92], shown in Fig. 3.4, where the advance ratio is defined as

$$J_j = \frac{||\mathbf{v}_b||}{2R\frac{u_{j,prev}^s}{60}} \tag{3.62}$$

where $u_{j,prev}^s$ is the propeller rotational speed at the previous time step, in RPM, and R is the propeller radius. The advance ratio is bounded such that $0 \leq J_j \leq 0.5$, which ensures the propeller is in its normal working state [9].

3.5.1.2 Control Surface

Air flowing over a deflected surface changes direction, causing a change in momentum of the air, which consequently exerts a force on the aircraft. For the same magnitude of

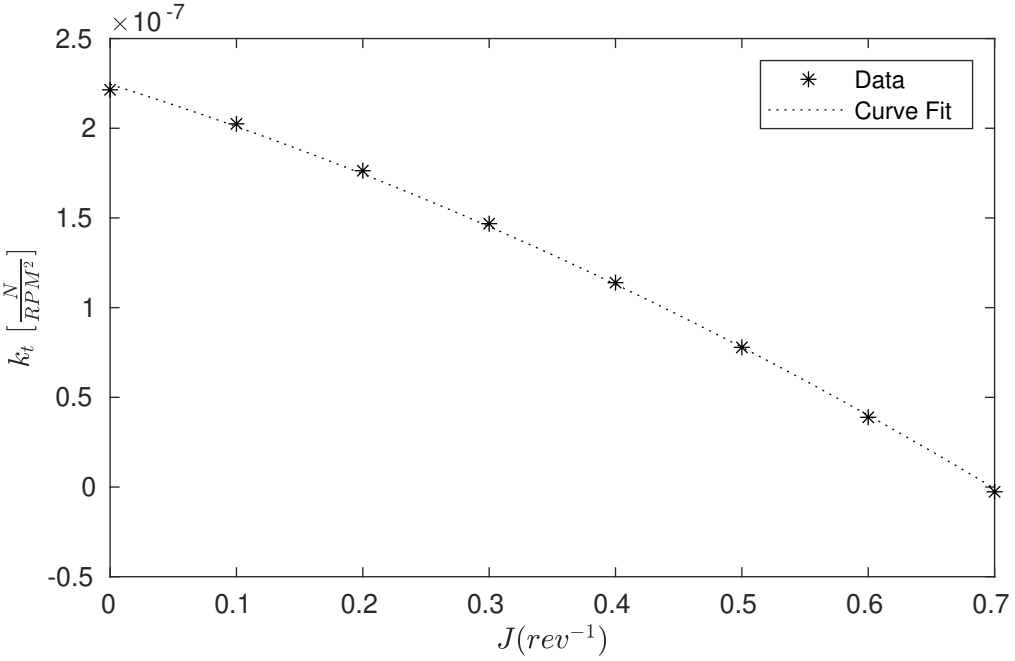


FIGURE 3.4: Thruster Coefficient vs Advance Ratio for Electrify PowerFlow 10 x 4.5 Propeller. The curve fit $k_t = (-1.439J^2 - 2.212J + 2.245) * 10^{-7}$.

deflection, faster flowing air undergoes a greater change in momentum, producing a larger force. These forces are typically small in magnitude but far from the aircraft center of mass; thus are ultimately used to exert moments on the aircraft. We model the force and torque generated by a control surface as follows:

$$\mathbf{u}_j^f = c_j v_{s,j}^2 \hat{\mathbf{d}}_j u_j^s \tag{3.63a}$$

$$\mathbf{u}_j^\tau = \mathbf{0} \tag{3.63b}$$

where the input signal is the deflection angle and the direction of the force is $\hat{\mathbf{d}}_j$. The constant specific to the control surface and atmospheric conditions is denoted by c_j, and can be extracted from bench tests performed in [9]. In the aircraft community it is common to scale the control surface effectiveness with the square of the airspeed [93]. In conventional aircraft, the speed of the airflow over the control surfaces is equivalent to the speed of the aircraft. However, for small agile aircraft the propeller slipstream effects must also be considered. For example, a hovering agile aircraft is stationary but yet generates all of its control authority from the propeller slipstream. Thus we correlate the control surface effectiveness with a slipstream speed approximation, $v_{s,j}$, instead of the aircraft speed. We can estimate this slipstream speed, $v_{s,j}$, using momentum theory [94]:

$$v_{s,j} = \sqrt{(\mathbf{v}_b \cdot \hat{\mathbf{f}}_b)^2 + \frac{2||\mathbf{u}_k^f||}{\rho \pi R^2}} \tag{3.64}$$

where the air density is denoted by ρ, the aircraft longitudinal speed is the projection of the velocity onto the thrust axis ($\mathbf{v}_b \cdot \hat{\mathbf{f}}_b$), and the commanded thrust force corresponding to the thruster inducing the slipstream is denoted by $||\mathbf{u}_k^f||$ (denoted by subscript k, because the index corresponds to thruster actuator, not control surface). Using this simplified model the slipstream approximation is the same over every control surface because an agile fixed-wing aircraft has one thruster. The approximated slipstream is bounded to always be greater than the slipstream in a hover (calculated using Eq. (3.64), and setting $\mathbf{v}_b \cdot \hat{\mathbf{f}}_b = 0$ and $||\mathbf{u}_k^f|| = \frac{mg}{\text{number of thrusters}}$), to avoid excessive control action at low slipstream values. In addition, we filter the approximated slipstream using a second-order low-pass filter with a 2 Hz natural frequency and .707 damping ratio, since a noisy approximated slipstream will cause abrupt changes in control surface deflections. The low-pass filter introduces some delay, but considering it takes some time for the flow created by the propeller to reach the control surfaces downstream, adding this delay is consistent with the slipstream we are modeling.

3.5.2 Obtaining Actuator Commands

We can now turn our attention to the question of how to generate appropriate control surface deflections and thruster speed to obtain the desired forces and moments, based on our simplified models. We denote the position vector from the UAV's center of mass to the j^{th} actuator's applied force as $\mathbf{r}_j$, which is resolved in the body frame. We can then compute the forces and torque's created by the control inputs as:

$$\mathbf{m}_b^c = \sum_{j=1}^{m} \mathbf{r}_j \times \mathbf{u}_j^f + \mathbf{u}_j^\tau \tag{3.65a}$$

$$\mathbf{f}_b^c = f^c \hat{\mathbf{f}}_b = \sum_{j=1}^{m} \mathbf{u}_j^f \tag{3.65b}$$

for a UAV with m actuators. An agile fixed-wing aircraft has four control inputs, one propeller rotational speed (u_1^s), an aileron (u_2^s), an elevator (u_3^s), and a rudder (u_4^s) deflection. The propeller control input is in RPM, while the control surface deflections are expressed in degrees. The aileron is made up of two control surfaces driven by one servomotor; thus each surface deflects equal and opposite causing the input signal u_2^s to produce two forces, one on each side of the plane and denoted by subscripts $2l$ and $2r$. Our test platform, Mcfoamy is shown in Fig. 3.5.

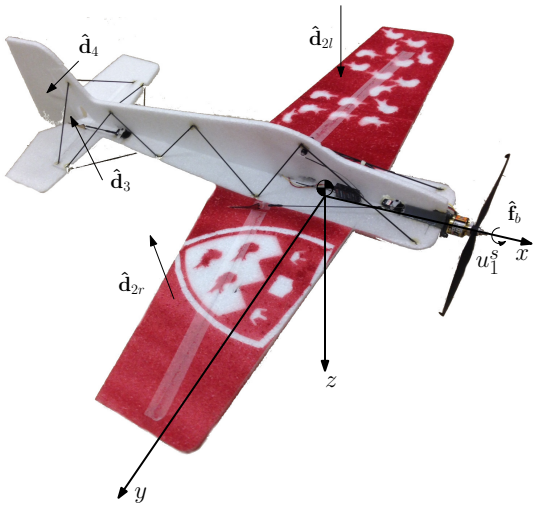


FIGURE 3.5: McFoamy Agile Fixed-Wing

For this aircraft, using the actuator models presented in Eq. (3.61) & Eq. (3.63) with Eq. (3.65) we can obtain:

$$\mathbf{m}_b^c = (\mathbf{r}_{2l} \times c_2 v_s^2 \hat{\mathbf{d}}_{2l} + \mathbf{r}_{2r} \times c_2 v_s^2 \hat{\mathbf{d}}_{2r}) u_2^s + \mathbf{r}_3 \times c_3 v_s^2 \hat{\mathbf{d}}_3 u_3^s + \mathbf{r}_4 \times c_4 v_s^2 \hat{\mathbf{d}}_4 u_4^s \tag{3.66}$$

$$f^c \hat{\mathbf{f}}_b = k_t u_1^{s^2} \hat{\mathbf{f}}_b \quad \Rightarrow f^c = k_t u_1^{s^2} \tag{3.67}$$

which give the applied moment and body fixed-force as a function of the input signal. Note that we assume the forces generated by the control surfaces are negligible compared to the thruster force, and that the torque generated by the propeller is negligible compared to the torque generated by the control surfaces. For this airframe, $\mathbf{r}_1 \times k_t \hat{\mathbf{f}}_b = \mathbf{0}$ which is why it is dropped in the equation. We drop the j subscript in v_{s_j},k_{t_j}, and k_{q_j}, since there is only one propeller and we assume the airflow over all the control surfaces is the same. We can rewrite Eq. (3.66) and Eq. (3.67) in matrix form, and invert it to obtain the input signal as a function of moment and body-fixed force:

$$\begin{bmatrix} u_1^{s^2} \\ u_2^s \\ u_3^s \\ u_4^s \end{bmatrix} = \begin{bmatrix} k_t & 0 & 0 & 0 \\ \mathbf{0} & (\mathbf{r}_{2l} \times c_2 v_s^2 \hat{\mathbf{d}}_{2l} + \mathbf{r}_{2r} \times c_2 v_s^2 \hat{\mathbf{d}}_{2r}) & \mathbf{r}_3 \times c_3 v_s^2 \hat{\mathbf{d}}_3 & \mathbf{r}_4 \times c_4 v_s^2 \hat{\mathbf{d}}_4 \end{bmatrix}^{-1} \begin{bmatrix} f^c \\ \mathbf{m}_b^c \end{bmatrix} \tag{3.68}$$

When further simplifying Eq. (3.68), the aircraft geometry causes the relationship between actuators and force/moments to decouple, causing each control surface to only affect the moment about one axis, and the throttle to only affect the body-fixed force, as given by

$$\begin{bmatrix} u_1^{s^2} \\ u_2^s \\ u_3^s \\ u_4^s \end{bmatrix} = \begin{bmatrix} k_t & 0 & 0 & 0 \\ 0 & \bar{c}_2 v_s^2 & 0 & 0 \\ 0 & 0 & \bar{c}_3 v_s^2 & 0 \\ 0 & 0 & 0 & \bar{c}_4 v_s^2 \end{bmatrix}^{-1} \begin{bmatrix} f^c \\ m_{b,x}^c \\ m_{b,y}^c \\ m_{b,z}^c \end{bmatrix}. \tag{3.69}$$

The specific values for the parameters used in control allocation are shown in Table 3.1.

TABLE 3.1: Agile Fixed-Wing Control Parameters

Variable	Unit	Agile Fixed-Wing
$\hat{\mathbf{f}}_b$	-	$[1,\ 0,\ 0]^T$
k_t	N/RPM^2	$(-1.439J^2 - 2.212J + 2.245)10^{-7}$
c_2	$N/(^\circ m^2/s^2)$	1.0665×10^{-4}
c_3	$N/(^\circ m^2/s^2)$	4.0034×10^{-4}
c_4	$N/(^\circ m^2/s^2)$	4.4899×10^{-4}
$\bar{c}_2$	$Nm/(^\circ m^2/s^2)$	-5.2292×10^{-5}
$\bar{c}_3$	$Nm/(^\circ m^2/s^2)$	-2.1618×10^{-4}
$\bar{c}_4$	$Nm/(^\circ m^2/s^2)$	-2.6939×10^{-4}
$\hat{\mathbf{d}}_{2l}$	-	$[0,\ 0,\ 1]^T$
$\hat{\mathbf{d}}_{2r}$	-	$[0,\ 0,\ -1]^T$
$\hat{\mathbf{d}}_3$	-	$[0,\ 0,\ -1]^T$
$\hat{\mathbf{d}}_4$	-	$[0,\ 1,\ 0]^T$
$\mathbf{r}_1$	m	$[0.25,\ 0,\ 0]^T$
$\mathbf{r}_{2l}$	m	$[0,\ -0.24,\ 0]^T$
$\mathbf{r}_{2r}$	m	$[0,\ 0.24,\ 0]^T$
$\mathbf{r}_3$	m	$[-0.54,\ 0,\ 0]^T$
$\mathbf{r}_4$	m	$[-0.60,\ 0,\ 0]^T$

Most of the time the thrust command is simply the force command. However, in the case of a saturated control surface, the thruster can also be used to generate more slipstream to produce a larger moment. We can compute a desired slipstream speed based on the desired moment and control surface characteristics, and then use this desired slipstream coupled with momentum theory to calculate a desired thrust force. We outline these calculations for a saturated elevator, although similar calculations could be made for a saturated aileron or rudder:

$$v_{s_{des}} = \sqrt{\frac{m_{b,y}^c}{\bar{c}_3 u_{3_{max}}^s}} \tag{3.70}$$

$$f^{additional} = \frac{\rho \pi R^2}{2}(v_{s_{des}}^2 - (\mathbf{v}_b \cdot \hat{\mathbf{f}}_b)^2) \tag{3.71}$$

The new total thrust force is obtained by summing the component to correct height and speed errors, f^c, and the component to increase the control authority when necessary (i.e. a saturated control surface), $f^{additional}$, given by Eq. (3.71). The new propeller rotational speed can be recomputed using Eq. (3.69), but replacing f^c with $f^c + f^{additional}$.

3.6 Extension to Other Platforms

The goal of our agile fixed-wing aircraft control system is to track position and orientation (six degrees of freedom); which coincides with the goal of any UAV controller. While different UAVs have various arrangements and types of actuators, for the majority of UAVs, the final effect of the actuators is the *same*: they produce a force along a body-fixed axis, and moments about three linearly independent axes, which is equivalent to a moment about an arbitrary axis. The reason for this similarity is most applications require complete control of the UAV orientation, thus requiring the ability to exert a moment about an arbitrary axis. The UAV requires the ability to generate a force in order to counteract gravity, and accelerate the vehicle in a desired direction. Although the control problem would be simplified if this force could be directed in an arbitrary manner, achieving this would require additional actuators, which would in turn add weight and cost to the platform. In order to save this weight and reduce complexity, most UAV platforms are built with a body-fixed direction of force, and the under-actuated system controls position by re-orienting itself to re-direct this force.

With exception of the control allocation discussed in Sec. 3.5, the control system developed for the agile fixed-wing aircraft can be applied to this class of UAVs that can exert a body-fixed force and a moment in any direction, which includes multi-copters, conventional fixed-wing, agile fixed-wing, most tailsitters, some tilt-rotor/wing platforms, and some flapping-wing vehicles. A universal controller has many advantages; including portability between different platforms.

In practice, the ideal UAV platform is dependent on the mission, as UAV designs typically trade off maneuverability and flight efficiency, as shown in Fig. 3.6. For missions in confined cluttered spaces, rotorcraft are the ideal platform. For missions requiring long range flight in uncluttered environments, fixed-wing aircraft are the ideal platform. In missions requiring both long range flight and flight in confined cluttered spaces, agile fixed-wing aircraft or tailsitters may be the most suitable platform.

In this section, we present control allocation approaches suitable for a quadrotor, a tail-sitter, a flapping-wing, and tilt-wing aircraft. These approaches, combined with the

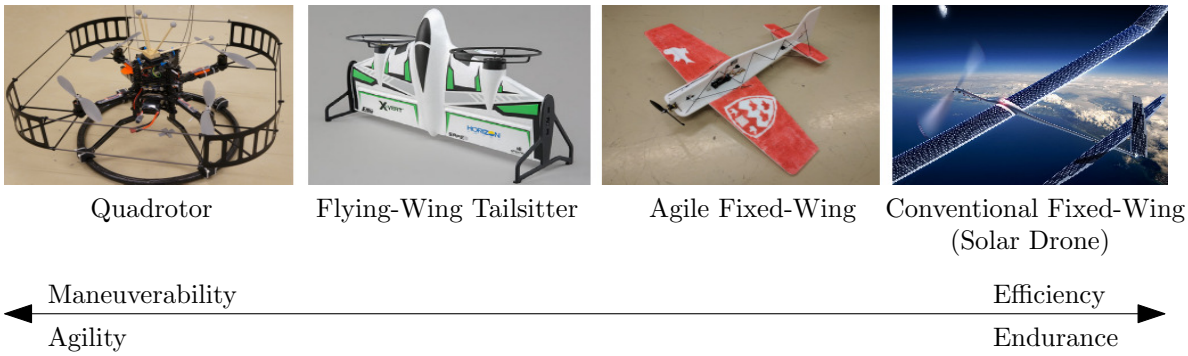


FIGURE 3.6: Types of Unmanned Aerial Vehicles

controller described in Sec. 3.1 - 3.3 allow a unified control methodology to be used for the widely different platforms.

3.6.1 Quadrotor

Consider a quadrotor with an 'X' configuration, such as the Pleiades Spiri shown in Fig. 3.7.

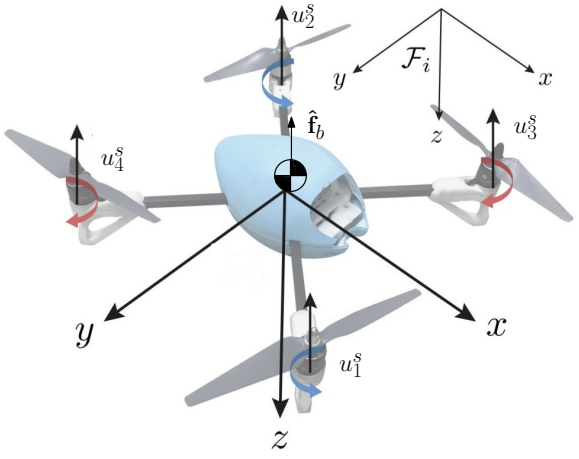


FIGURE 3.7: Spiri Quadrotor

A quadrotor platform has four propellers (which are all the same type, so the j subscript gets dropped in k_{t_j} and k_{q_j}); we substitute the propeller model in Eq. (3.61) into Eq. (3.65) for $j = 1, 2, 3, 4$ to obtain:

$$\mathbf{m}_b^c = \sum_{j=1}^{4} \mathbf{r}_j \times k_t u_j^{s^2} \hat{\mathbf{f}}_b \pm k_q u_j^{s^2} \hat{\mathbf{f}}_b = \sum_{j=1}^{4} (\mathbf{r}_j \times k_t \hat{\mathbf{f}}_b \pm k_q \hat{\mathbf{f}}_b) u_j^{s^2} \tag{3.72}$$

$$f^c\hat{\mathbf{f}}_b = \sum_{j=1}^{4} k_t u_j^{s^2}\hat{\mathbf{f}}_b \quad \Rightarrow f^c = \sum_{j=1}^{4} k_t \tag{3.73}$$

where we can obtain the applied moment and body fixed-force as a function of the input signal. We also note that the force equation simplifies to scalar form. We can rewrite Eq. (3.72) and Eq. (3.73) in matrix form, and invert it to obtain the input signal as a function of moment and body-fixed force:

$$\begin{bmatrix} u_1^{s^2} \\ u_2^{s^2} \\ u_3^{s^2} \\ u_4^{s^2} \end{bmatrix} = \begin{bmatrix} k_t & k_t & k_t & k_t \\ (\mathbf{r}_1 \times k_t\hat{\mathbf{f}}_b - k_q\hat{\mathbf{f}}_b) & (\mathbf{r}_2 \times k_t\hat{\mathbf{f}}_b - k_q\hat{\mathbf{f}}_b) & (\mathbf{r}_3 \times k_t\hat{\mathbf{f}}_b + k_q\hat{\mathbf{f}}_b) & (\mathbf{r}_4 \times k_t\hat{\mathbf{f}}_b + k_q\hat{\mathbf{f}}_b) \end{bmatrix}^{-1} \begin{bmatrix} f^c \\ \mathbf{m}_b^c \end{bmatrix} \tag{3.74}$$

3.6.2 Tailsitter

Consider the twin-thruster flying-wing tailsitter shown in Fig. 3.8, which has four control inputs: two propellers (which are all the same type, so the j subscript gets dropped in k_{t_j} and k_{q_j}), u_1^s & u_2^s, and two control surface, u_3^s & u_4^s, called elevons. Using the actuator

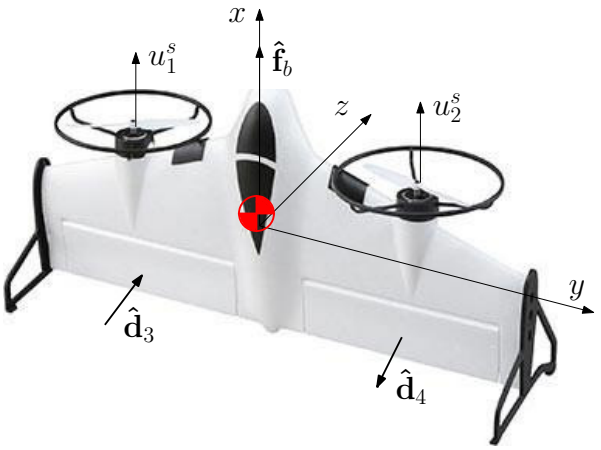


FIGURE 3.8: Tailsitter

models in Eq. (3.61) & Eq. (3.63) with Eq. (3.65) we can obtain:

$$\mathbf{m}_b^c = (\mathbf{r}_1 \times k_t\hat{\mathbf{f}}_b + k_q\hat{\mathbf{f}}_b)u_1^{s^2} + (\mathbf{r}_2 \times k_t\hat{\mathbf{f}}_b - k_q\hat{\mathbf{f}}_b)u_2^{s^2} + \mathbf{r}_3 \times c_3 v_{s_3}^2\hat{\mathbf{d}}_3 u_3^s + \mathbf{r}_4 \times c_4 v_{s_4}^2\hat{\mathbf{d}}_4 u_4^s \tag{3.75}$$

$$f^c\hat{\mathbf{f}}_b = (k_t u_1^{s^2} + k_t u_2^{s^2})\hat{\mathbf{f}}_b \quad \Rightarrow f^c = k_t u_1^{s^2} + k_t u_2^{s^2} \tag{3.76}$$

where once again, we assume the force generated by the control surfaces is negligible compared to the thruster forces in order to classify this vehicle as one that can apply force in a body-fixed direction. We can write this in matrix form and invert it to obtain the control inputs as a function of moment and body-fixed force:

$$\begin{bmatrix} u_1^{s^2} \\ u_2^{s^2} \\ u_3^s \\ u_4^s \end{bmatrix} = \begin{bmatrix} k_t & k_t & 0 & 0 \\ (\mathbf{r}_1 \times k_t \hat{\mathbf{f}}_b + k_q \hat{\mathbf{f}}_b) & (\mathbf{r}_2 \times k_t \hat{\mathbf{f}}_b - k_q \hat{\mathbf{f}}_b) & \mathbf{r}_3 \times c_3 v_{s_3}^2 \hat{\mathbf{d}}_3 & \mathbf{r}_4 \times c_4 v_{s_4}^2 \hat{\mathbf{d}}_4 \end{bmatrix}^{-1} \begin{bmatrix} f^c \\ \mathbf{m}_b^c \end{bmatrix} \tag{3.77}$$

Unlike the agile fixed-wing aircraft, the tailsitter has two thrusters, one positioned on each side of the aircraft, implying that the slipsteam approximation for control surface 3 should be based on thruster 1, while the slipstream approximation for control surface 4 should be based on thruster 2.

3.6.3 Flapping-Wing

Consider the Delfly flapping-wing UAV shown in Fig. 3.9, which has four control inputs, one flapping-wing (u_1^s), an aileron (u_2^s), an elevator (u_3^s), and a rudder (u_4^s) deflection. The aileron is made up of two control surfaces driven by one servomotor; thus each surface deflects equal and opposite causing the input signal u_2^s to produce two forces, one on each side of the plane and denoted by subscripts $2l$ and $2r$.

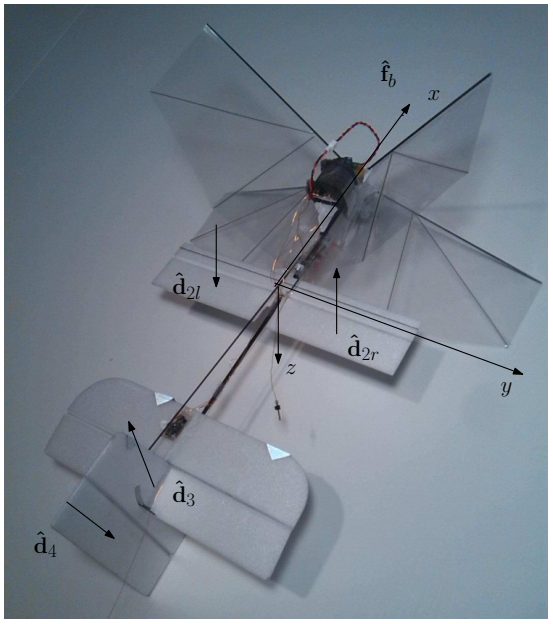


FIGURE 3.9: Delfly Flapping-Wing [10]

We assume that we can experimentally obtain a function relating the flapping frequency of a flapping surface to the mean thrust it generates. Using this assumption we can generally model a flapping surface as:

$$\mathbf{u}_j^f = g(u_j^s)\hat{\mathbf{f}}_b \tag{3.78a}$$

$$\mathbf{u}_j^\tau = \mathbf{0} \tag{3.78b}$$

where we assume this function g can be obtained experimentally, and u_j^s is the frequency of the flapping surface.

Using the actuator models presented in Eq. (3.63) & Eq. (3.78) with Eq. (3.65) we can obtain:

$$\mathbf{m}_b^c = (\mathbf{r}_{2l} \times c_2 v_s^2 \hat{\mathbf{d}}_{2l} + \mathbf{r}_{2r} \times c_2 v_s^2 \hat{\mathbf{d}}_{2r}) u_2^s + \mathbf{r}_3 \times c_3 v_s^2 \hat{\mathbf{d}}_3 u_3^s + \mathbf{r}_4 \times c_4 v_s^2 \hat{\mathbf{d}}_4 u_4^s \tag{3.79}$$

$$f^c \hat{\mathbf{f}}_b = g(u_1^s)\hat{\mathbf{f}}_b \quad \Rightarrow f^c = g(u_1^s) \tag{3.80}$$

which give the applied moment and body fixed-force as a function of the input signal. Similar to the previously mentioned vehicles, we assume the force generated by the control surfaces are negligible compared to the flapping-wing thrust. For this airframe, $\mathbf{r}_1 \times k_t \hat{\mathbf{f}}_b = \mathbf{0}$ which is why it is dropped in the equation. We also note that the force equation simplifies to scalar form. We can rewrite Eq. (3.79) in matrix form, and invert it to obtain the input signal as a function of the desired moment:

$$\begin{bmatrix} u_2^s \\ u_3^s \\ u_4^s \end{bmatrix} = \begin{bmatrix} (\mathbf{r}_{2l} \times c_2 v_s^2 \hat{\mathbf{d}}_{2l} + \mathbf{r}_{2r} \times c_2 v_s^2 \hat{\mathbf{d}}_{2r}) & \mathbf{r}_3 \times c_3 v_s^2 \hat{\mathbf{d}}_3 & \mathbf{r}_4 \times c_4 v_s^2 \hat{\mathbf{d}}_4 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{m}_b^c \end{bmatrix} \tag{3.81}$$

To generate the desired body-fixed force, we assume that the relationship of Eq. (3.80) is invertible, leading to

$$u_1^s = g^{-1}(f^c) \tag{3.82}$$

3.6.4 Tilt-Wing

Consider the Vahana tilt-wing aircraft, a single-passenger flying taxi proposed by Airbus A^3 [11], shown in Fig. 3.10. This aircraft has eight propellers—four on the front wing, and four on the rear wing—and two control surfaces on the rear wing. The wing tilt angle, γ, is the angle between the direction of the thrust force and the body x-axis.

Potentially, this angle gives an additional degree of freedom to the controller. However, since the wing tilt is likely to take place at a much slower rate than variations of thrust or control surface deflections, we can assume that the variation of γ will be determined by the trajectory generator, rather than by the feedback controller. In this case, the angle γ can be viewed by the controller as a prescribed variable, rather than an unknown, and the control allocation can be obtained by combining the actuator models of Eq. (3.61) & Eq. (3.63) with Eq. (3.65) to obtain:

$$\mathbf{m}_b^c = \sum_{j=1}^{8}(\mathbf{r}_j \times k_t\hat{\mathbf{f}}_b + (-1)^n k_q\hat{\mathbf{f}}_b)u_j^{s^2} + \sum_{j=9}^{10}(\mathbf{r}_j \times c_j v_{s_j}^2 \hat{\mathbf{d}}_j u_j^s) \tag{3.83}$$

$$f^c\hat{\mathbf{f}}_b = \sum_{j=1}^{8} k_t u_j^{s^2}\hat{\mathbf{f}}_b \quad \Rightarrow f^c = \sum_{j=1}^{8} k_t u_j^{s^2} \tag{3.84}$$

Another difference between the Vahana platform and the preceding platforms is that it is redundantly-actuated—there are ten actuators to generate four force/moments. This results in infinite possible sets of actuator commands that can be used to generate the applied moment and body-fixed force. This indeterminacy is best resolved by formulating and solving an optimization problem, and using Eq. (3.83) and Eq. (3.84) as constraints to that problem. For example, one could minimize an objective consisting of the weighted norm of the vector of control inputs $\sum_{j=1}^{8}(w_j u_j^{s^2})^2 + \sum_{j=9}^{10}(w_j u_j^s)^2$. In this case, the resulting optimization problem would have a quadratic objective and linear constraints in the design variables, and would be solvable in real-time. Another advantage of this optimization approach is that inequality constraints can be included to represent actuator limits, such as maximum control surface deflections and maximum propeller speeds.

(a) $\gamma = 0°$ (b) $\gamma = 90°$

FIGURE 3.10: Vahana Tilt-Wing [11]

3.6.5 Other Platform Properties

We show the values of the platform specific properties in Table 3.2. Not all the values are applicable to every platform, and those values are denoted by N/A. For the flapping-wing and tilt-wing we cannot specify some values because we do not have the platforms, which is denoted by NI (no info).

TABLE 3.2: Other Platform Control Parameters

Variable	Unit	Quadrotor	Tailsitter	Flapping-Wing	Tilt-Wing
$\hat{\mathbf{f}}_b$	-	$[0,\ 0,\ -1]^T$	$[1,\ 0,\ 0]^T$	$[1,\ 0,\ 0]^T$	$[\cos(\gamma),\ 0,\ -\sin(\gamma)]^T$
k_t	N/RPM^2	6.91×10^{-8}	NI	NI	NI
k_q	Nm/RPM^2	1.12×10^{-9}	NI	N/A	NI
c_2	$N/(^\circ m^2/s^2)$	N/A	N/A	NI	N/A
c_3	$N/(^\circ m^2/s^2)$	N/A	NI	NI	N/A
c_4	$N/(^\circ m^2/s^2)$	N/A	NI	NI	N/A
$\hat{\mathbf{d}}_{2l}$	-	N/A	N/A	$[0,\ 0,\ 1]^T$	N/A
$\hat{\mathbf{d}}_{2r}$	-	N/A	N/A	$[0,\ 0,\ -1]^T$	N/A
$\hat{\mathbf{d}}_3$	-	N/A	$[0,\ 0,\ 1]^T$	$[0,\ 0,\ -1]^T$	N/A
$\hat{\mathbf{d}}_4$	-	N/A	$[0,\ 0,\ -1]^T$	$[0,\ 1,\ 0]^T$	N/A
$\hat{\mathbf{d}}_9$	-	N/A	N/A	N/A	$[\sin(\gamma),\ 0,\ \cos(\gamma)]^T$
$\hat{\mathbf{d}}_{10}$	-	N/A	N/A	N/A	$[-\sin(\gamma),\ 0,\ -\cos(\gamma)]^T$
$\mathbf{r}_1$	m	$[0.1626,\ 0.1626,\ 0]^T$	$[0.05,\ -0.15,\ 0]^T$	NI	NI
$\mathbf{r}_2$	m	$[-0.1626,\ -0.1626,\ 0]^T$	$[0.05,\ 0.15,\ 0]^T$	N/A	NI
$\mathbf{r}_{2l}$	m	N/A	N/A	NI	N/A
$\mathbf{r}_{2r}$	m	N/A	N/A	NI	N/A
$\mathbf{r}_3$	m	$[0.1626,\ -0.1626,\ 0]^T$	$[-0.15,\ -0.13,\ 0]^T$	NI	NI
$\mathbf{r}_4$	m	$[-0.1626,\ 0.1626,\ 0]^T$	$[-0.15,\ 0.13,\ 0]^T$	NI	NI

Chapter 4

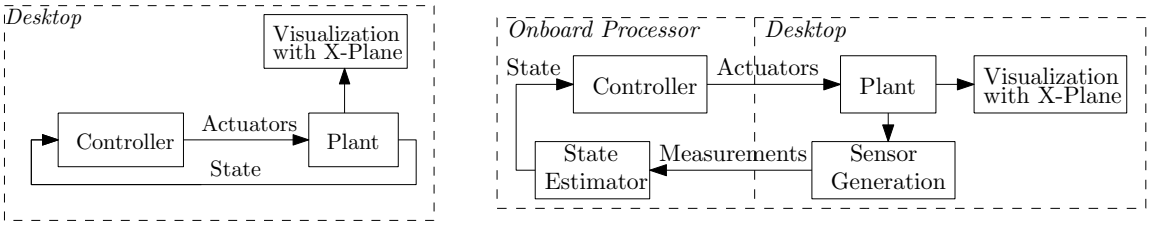
Controller Validation

In this chapter we discuss the validation of the control system developed in Chapter 3. We discuss our simulation, maneuver generation, agile fixed-wing test platform, and then present results from simulations, indoor flights, and outdoor flights. Finally, we validate the extension to other platforms using a quadrotor.

4.1 Simulation

We utilize the dynamics model presented in Chapter 2 to validate the control system prior to flight testing. We first use a conventional simulation, depicted in Fig. 4.1a, which is implemented in real-time in MATLAB/Simulink and is visualized using X-Plane [84]. The conventional simulation is used for initial evaluation of the control algorithm and to adjust control gains. It has been found easiest to tune the gains from the inside out: the thrust can be temporarily set to a constant value, and the position control gains can be set to zero, while the attitude controller is tuned first. The position control gains can then be increased until satisfactory position tracking is achieved. Finally, the thrust control can be tuned to track longitudinal speed and height. Using this approach, we are able to converge to a set of control gains which lead to successful autonomous aerobatic flight in this conventional simulation environment.

Conventional simulations are a useful tool for evaluating the performance of a control system before testing in an actual flight. However, they typically lack the ability to entirely model many of the potential issues associated with real-time implementation on flight hardware. Phenomena such as sensor noise, state estimation errors, controller



(a) Conventional Simulation

(b) HIL Simulation

FIGURE 4.1: Simulation Environments

discretization, memory overflow, and timing delays, can all be easily overlooked in a conventional simulation, yet can make the aircraft unstable in real flight. An extra measure to reduce the likelihood of a crash during flight testing is to validate the control system in a hardware-in-the-loop (HIL) simulation [95, 96]. In an HIL simulation, depicted in Fig. 4.1b, the flight simulator sends artificial sensor measurements to the on-board processor, which is running as if it were in a real flight. The on-board processor executes its state estimation and control algorithms, and sends artificial actuator commands back to the flight simulator. Since the control algorithm is implemented on the flight hardware, the HIL simulation allows the effect of the aforementioned real-time processor issues to be evaluated in a simulation environment before testing in flight.

The flight controller and open-source firmware used on the test platform, the Pixhawk and PX4 respectively, come with a built-in HIL environment which uses the X-Plane [84] physics engine as the plant. However, as we believe our dynamics model to be more realistic than X-Plane's, we modify this setup to replace the X-Plane physics engine with our in-house Simulink dynamics model. This modification is discussed in detail in Appendix A.

The initial tuning of the control gains in the conventional simulation typically leads to large control gains which perform well. Applying these same gains in the HIL simulation will often result in unstable aircraft motion due to the large amplification of state estimation errors caused by sensor noise. Thus, the control gains need to be significantly reduced in order for the aircraft to become stable in the HIL simulation. To demonstrate this we run simulations using the initial set of high control gains, as well as the re-tuned lowered control gains to control the aircraft in level flight in both the conventional and HIL simulation environment. All four scenarios are shown in Fig. 4.2, where the aircraft is flying in the x direction aiming to keep y and z zero. We can see the aircraft is unstable using the high gains in the HIL, while remaining stable in the conventional simulation. By lowering the gains, the aircraft achieves stable flight in both simulations.

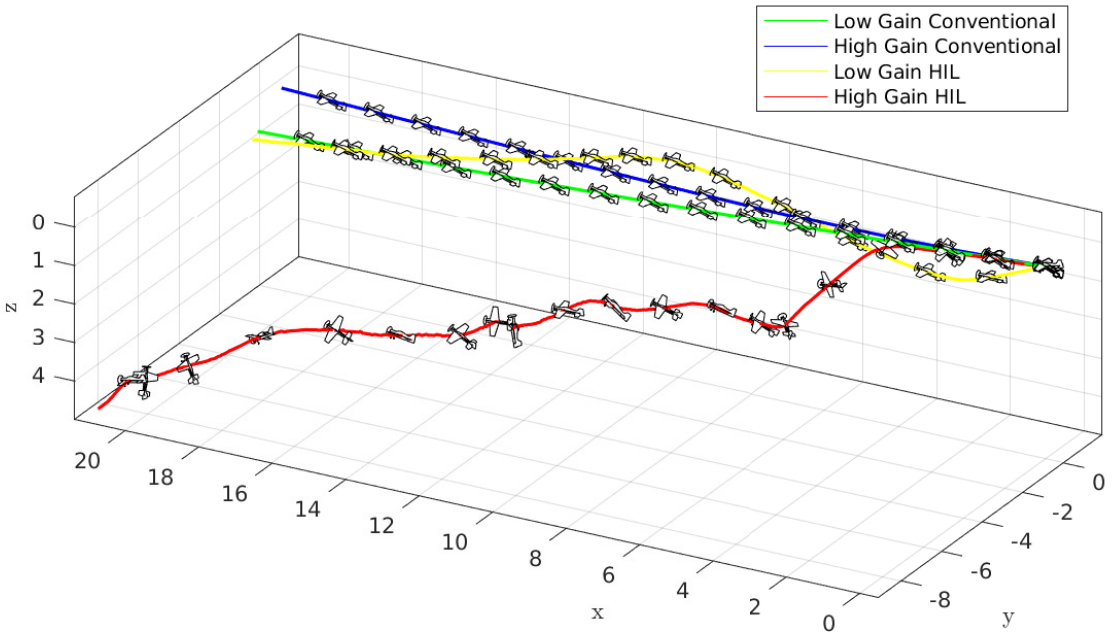


FIGURE 4.2: Control Gain Comparison

The absence of phenomena unaccounted for in the traditional simulation, but present in an HIL simulation, lead to large control gains in the traditional simulation. This would be very problematic and likely cause crashes if an attempt was made to transition directly to flight testing. As well, tracing and resolving these issues during flight testing would be a difficult process. Using the HIL simulation as an intermediate step allows the resolution of these issues in the lab, avoiding many crashes. The first autonomous flight test was successful without any modifications to the code used in the HIL simulation. The aircraft never crashed during a flight test validating the control logic, largely due to the use of the HIL simulation in pre-flight planning.

4.2 Platform Description

The experimental platform is an off-the-shelf RC aircraft, the WM Parkflyers McFoamy, which is retrofitted with additional carbon fiber reinforcements and a custom 3D printed motor mount. The aircraft is made of EPP foam, and is equipped with a 50C 3S 850mA LiPo battery, 3 HiTEC HS-65MG metal gear feather servos, the Electrify SS-25 Brushless ESC, the Great Planes Rimfire 400 brushless motor and the Electrify PowerFlow 10 x 4.5

propeller. A Pixhawk flight controller enables autonomous flight. The platform used in the indoor controller validation experiments uses the original Pixhawk, and the platform used in the outdoor controller validation tests uses the 3DR Pixhawk Mini flight controller [97]. Upgrading the flight controller to the Pixhawk Mini reduced the aircraft's mass from 480 grams to 450 grams. In both cases, the hardware runs the open source PX4 flight stack that allows data logging, sensor integration, control and state estimation. The default PX4 state estimator, the EKF2, fuses the Pixhawk's embedded IMU, barometer, as well as a GPS module with compass to provide an estimate of the position, attitude, velocity, and body rate. The default PX4 control module is replaced with the controller presented in Chapter 3, and is executed at 200 Hz. The McFoamy controller validation aircraft is displayed in Fig. 3.5 and the physical properties are shown in Table 4.1.

TABLE 4.1: Aircraft Properties

Parameter	Symbol	Value	Unit
Mass	m	0.45/0.48	kg
Moments of Inertia	I_x	3.922×10^{-3}	kg m^2
	I_y	1.594×10^{-2}	kg m^2
	I_z	1.934×10^{-2}	kg m^2
Non-zero Products of Inertia	I_{xz}	3.03×10^{-4}	kg m^2
Wing Area	–	0.143	m^2
Wing Span	–	0.864	m
Mean Aerodynamic Chord	–	0.21	m
Maximum Aileron Deflection	$u^s_{2_{max}}$	52	deg
Maximum Elevator Deflection	$u^s_{3_{max}}$	59	deg
Maximum Rudder Deflection	$u^s_{4_{max}}$	49	deg
Propeller Radius	R	0.127	m

The controller generates three control surface deflections, and one propeller rotational speed. The control surfaces are attached to control rods driven by servo motors while the propeller is driven by a brushless DC motor controlled by an electronic speed controller (ESC). The servos and the ESC are all commanded using pulse-width-modulated (PWM) signals. We therefore had to experimentally characterize the servo linkages and the ESC in order to ensure that the appropriate PWM signals were generated and that the desired control surface deflections and propeller speed were obtained. These characteristics were implemented as curve-fits to translate the controller outputs to PWM signals during operation of the system.

4.3 Maneuver Generator

In the preceding chapter, we developed a control algorithm capable of tracking a given reference trajectory. In Chapter 5 the reference trajectory is specified through the obstacle avoidance algorithm. However, we validate the controller prior to validating the obstacle avoidance, and generate aerobatic reference trajectories via a heuristic 'maneuver generator'. The maneuver generator specifies a time history of reference motion variables: reference orientation, reference position, and reference longitudinal speed, $v_{r,x}^{ref} (= \mathbf{v}_r^{ref} \cdot \hat{\mathbf{f}}_r^{ref})$. The reference translational and angular velocities are not specified in the maneuver generator because the controller validation used an earlier version of the control architecture, which used proportional-derivative control on position and orientation, as opposed to proportional control on position, orientation, translational velocity, and angular velocity.

The reference trajectories generated are not always kinematic and dynamically feasible in order to test the limits of the control structure. While it is not possible for the aircraft to perfectly track the output of the maneuver generator, we have found that this approach does allow aerobatic maneuvers to be accomplished efficiently.

4.3.1 Reference Position

Before discussing the generation of specific maneuvers, we address the generation of the reference position, as this applies to multiple maneuvers. Generating the reference position can be done using two approaches, depending on whether the controller aims to maintain a specified *position*, or if it aims to follow a specified *path*. In the first approach, the user simply specifies that reference position. The second approach requires a more detailed explanation, best illustrated by an example.

Consider an aircraft at position $\mathbf{p}_i$, aiming to fly along a line, as shown in the top view of Fig. 4.3.

We define the desired flight path as the line extending from the aircraft position at the initiation of the maneuver, $\mathbf{p}_i^0$, extending in the direction of the reference yaw, ψ^{ref}, represented by the dashed line in Fig. 4.3. The reference position sent to the position controller is the point on the desired flight path closest to the aircraft, which can be calculated as follows:

$$\mathbf{p}_{i,xy}^{ref} = \mathbf{p}_{\parallel} = (\mathbf{p}_{i,xy} - \mathbf{p}_{i,xy}^0) \bullet \begin{bmatrix} \cos\psi^{ref} \\ \sin\psi^{ref} \end{bmatrix}) \begin{bmatrix} \cos\psi^{ref} \\ \sin\psi^{ref} \end{bmatrix} + \mathbf{p}_{i,xy}^0 \tag{4.1}$$

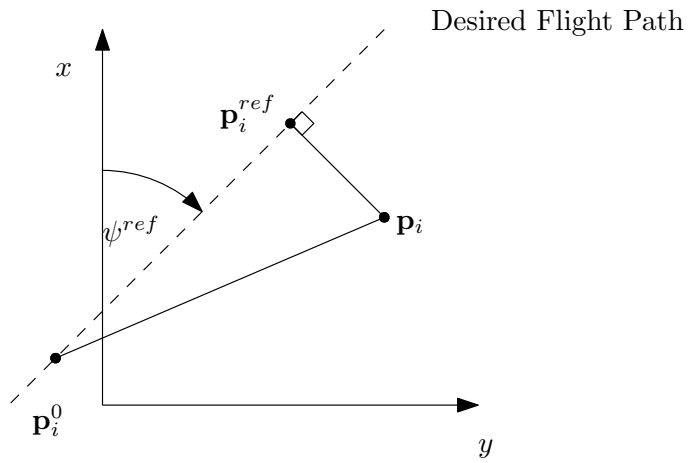


FIGURE 4.3: Line Following Example (Top-Down View)

where the x and y components of the variable are denoted by $()_{xy}$. In Eq. (4.1), the term in parentheses represents the length of $(\mathbf{p}_{i,xy} - \mathbf{p}^0_{i,xy})$ projected onto the desired flight path. This is then multiplied by the unit vector in the direction of the desired flight path to obtain the vector from the initial position to the reference position. It should be noted that $\mathbf{p}^{ref}_{i,z}$ is left free to be defined according to the maneuver, as will be discussed later in this section.

4.3.2 Straight and Level

While level flight is not considered an aerobatic maneuver, we nevertheless discuss how it is specified because it is often used to transition between other aerobatic maneuvers. When flying straight and level, the aim of the aircraft is to fly level while tracking a straight line at constant altitude. This desired motion implies a zero reference roll angle. We allow the reference longitudinal speed and heading to be arbitrarily chosen. The reference pitch angle is chosen to be consistent with the specified reference longitudinal speed, which is derived in [1]. Thus, the maneuver generator for level flight is as follows:

$$\mathbf{q}^{ref} = \text{EulToQuat}(\phi = 0°, \theta = \theta^{ref}, \psi = \psi^{ref})$$
$$\mathbf{p}_i^{ref} = [\mathbf{p}_{\|},\ p^0_{i,z}]$$

where Euler angles are converted to their equivalent quaternion representation.

4.3.3 Knife-Edge

The knife-edge maneuver is useful for flying between obstacles when the passage is narrower than the aircraft's wingspan. The goal of a knife-edge is to maintain 90° roll while tracking a straight line at constant altitude. An autonomous knife-edge maneuver is depicted in Fig. 4.4. Similarly to level flight, we allow the reference longitudinal speed and heading to be user-specified, and determine the reference pitch based on the reference speed [1]. The maneuver generator for knife-edge flight is as follows:

$$\mathbf{q}^{ref} = \text{EulToQuat}(\phi = 90°, \theta = \theta^{ref}, \psi = \psi^{ref})$$
$$\mathbf{p}_i^{ref} = [\mathbf{p}_{\|},\ p_{i,z}^0]$$

FIGURE 4.4: Knife-Edge Image Sequence

4.3.4 Rolling Harrier

In a rolling Harrier maneuver, the aircraft flies along a constant altitude line, while maintaining a constant roll rate. An autonomous Rolling Harrier is demonstrated in Fig. 4.5. While this maneuver has little practical utility, it does demonstrate the aircraft's extreme flight capability and allows us to demonstrate the versatility of our control system. As

in the two previous maneuvers, the reference longitudinal speed and heading are user-specified, and the reference pitch angle is chosen as a function of the reference longitudinal speed [1]. The maneuver generator for a rolling Harrier is as follows:

$$\mathbf{q}^{ref} = \text{EulToQuat}(\phi = \Omega t, \theta = \theta^{ref}, \psi = \psi^{ref})$$
$$\mathbf{p}_i^{ref} = [\mathbf{p}_{\|},\ p_{i,z}^0]$$

where the desired roll rate is denoted by Ω, and the time in seconds since the initiation of the maneuver is denoted by t.

FIGURE 4.5: Rolling Harrier Image Sequence

4.3.5 Hover

Hovering can be useful for surveillance tasks, as the aircraft remains stationary in a vertical orientation. An autonomous hover is depicted in Fig. 4.6. Given the nature of the maneuver, the reference pitch is 90° and the reference longitudinal speed is zero. To ensure a smooth transition into the hover, the reference heading is unchanged from the preceding maneuver (most likely level flight) and the reference roll angle is set to zero. This allows the transition to only occur along the pitch axis.

Ideally, we would like the aircraft to stop instantaneously when commanding a hover; thus the reference position is initially set to the initial position of the maneuver. In order to avoid the aircraft backtracking to that position, we change the reference position to the aircraft's current position once the vertical orientation has been achieved. This new reference position, $\mathbf{p}_i^f$, is kept for the remainder of the hover. The maneuver generator for the hover maneuver is as follows:

$$v_{r,x}^{ref} = 0$$
$$\mathbf{q}^{ref} = \text{EulToQuat}(\phi = 0, \theta = 90°, \psi = \psi^{ref})$$
$$\mathbf{p}_i^{ref} = \begin{cases} \mathbf{p}_i^0 & \text{prior to vertical orientation} \\ \mathbf{p}_i^f & \text{otherwise} \end{cases}$$

FIGURE 4.6: Hover Image Sequence

4.3.6 Aggressive Turnaround

Some situations may require the aircraft to reverse course quickly in a small space. We design the maneuver assuming the aircraft is initially in level flight, and must end the maneuver in level flight with opposite heading. We first command the aircraft into a hover. Once the pitch angle exceeds 45°, we command inverted flight with a heading opposite that at the start of the maneuver. Finally, once the aircraft pitches down to less than the reference pitch angle, the aircraft is commanded to roll back to level flight. Transitions between stages are unidirectional – i.e. once the maneuver proceeds to the next stage, it cannot go back to a previous stage, regardless of its orientation. An autonomous aggressive turnaround maneuver is demonstrated in Fig. 4.7. The maneuver generator for an aggressive turnaround is defined as follows:

Stage 1 $\theta : \theta_0 \rightarrow 45°$
$\mathbf{q}^{ref} = \text{EulToQuat}(\phi = 0, \theta = 90°, \psi = \psi^{ref})$
$\mathbf{p}_i^{ref} = \mathbf{p}_i^0$

Stage 2 $\theta : 45° \rightarrow 90° \rightarrow \theta^{ref}$
$\psi^{ref} \leftarrow \psi^{ref} + 180°$
$\mathbf{q}^{ref} = \text{EulToQuat}(\phi = 180°, \theta = \theta^{ref}, \psi = \psi^{ref})$
$\mathbf{p}_{i,xy}^{ref} = \mathbf{p}_{\|}$

Stage 3 otherwise
$\mathbf{q}^{ref} = \text{EulToQuat}(\phi = 0, \theta = \theta^{ref}, \psi = \psi^{ref})$
$\mathbf{p}_{i,xy}^{ref} = \mathbf{p}_{\|}$

FIGURE 4.7: Aggressive Turnaround Image Sequence

4.4 Results

We validate the control logic in traditional simulation, hardware-in-the-loop simulation, indoor flight tests, and outdoor flight tests. The indoor flight tests took place in the Concordia Stinger Dome, Montreal, Canada, whose roof is GPS transparent, allowing full GPS signal in a wind-free environment. The flying area is a mini soccer field, of dimension 30 $m \times 55$ m. The outdoor flight tests took place at the West Island Model Aeronautics Club, Montreal, Canada, which has a flight field of dimension 100 $m \times 100$ m. During the tests, the average wind gusts recorded at a nearby airport were 10-12 knots [98]. All the autonomous flight modes tested were airborne maneuvers, i.e. takeoff and landing were performed manually by a professional RC pilot. Once airborne, control authority was switched from the pilot to the control algorithm. For landings, control authority was switched back to the pilot who would manually land the aircraft. A video from an indoor flight is shown in https://youtu.be/yRMwQVy9tHo, and a video from an outdoor flight is shown in https://youtu.be/w0JX8BKc3wAWe.

To demonstrate the effectiveness of the controller in flight tests, and to clearly demonstrate differences between the simulations and experiments, each maneuver is performed in the conventional simulation, the HIL simulation, and actual flight. The flight data is shown in Figs. 4.8 - 4.19. For each maneuver we present three figures, one corresponding

to the indoor flight, one corresponding to the outdoor flight at 5 $\frac{m}{s}$, and one corresponding to the outdoor flight at 9 $\frac{m}{s}$. In each figure, we overlay the corresponding conventional and HIL simulation data onto the actual flight data. We use the same control gains to perform every maneuver, as well as use the same control gains in the corresponding conventional and HIL simulation. The control gains were slightly adjusted when transitioning from indoor to outdoor flight testing (also a new airframe was used), which are shown in Table 4.2.

TABLE 4.2: Controller Gains

Gain	Symbol	Value (Indoor)	Value (Outdoor)	Unit
Position Proportional	K_{p_p}	0.1	0.08	rad/m
Position Derivative	K_{p_d}	0.1	0.1	$rad/\frac{m}{s}$
Attitude Proportional	$\mathbf{K_{a_p}}$	130 $\mathbf{1}_{3x3}$	160 $\mathbf{1}_{3x3}$	$\frac{rad}{s^2}/rad$
Attitude Derivative	$\mathbf{K_{a_d}}$	8 $\mathbf{1}_{3x3}$	8 $\mathbf{1}_{3x3}$	$\frac{rad}{s^2}/\frac{rad}{s}$
Speed Proportional	K_v	5	3	$\frac{m}{s^2}/\frac{m}{s}$
Height Proportional	K_{h_p}	15	5	$\frac{m}{s^2}/m$
Height Integral	K_{h_i}	1	0.5	$\frac{m}{s^2}/ms$

The initial implementation of the controller tested in the indoor flights and corresponding simulations contained a feedforward component to the attitude controller. This feedforward term degraded the controller performance in the indoor experiments, and was thus subsequently removed before the outdoor flight testing. In addition, the aerodynamic force used in the controller was initially (in the indoor flights) approximated as a gain (chosen to be 0.1) multiplied by the square of the aircraft velocity. This was improved for the outdoor experiments using the method described in Sec. 3.2, with $K_{aero} = 2$.

During indoor flight testing, we evaluate each maneuver using a reference longitudinal speed of 5 $\frac{m}{s}$, and outdoors we use both 5 $\frac{m}{s}$ & 9 $\frac{m}{s}$ as the reference longitudinal speed. For consistency in the initial conditions for the aerobatic maneuver in each environment, we command level flight prior to the initiation of each maneuver. While only one set of results is shown here for each maneuver, the flight tests were repeated multiple times, each time achieving similar results. An exception to this is for the hover maneuver outdoors, where the wind has the largest effect on the aircraft, because the wings present a large surface area to the wind. Although the hover always remained stable, at times there were larger drifts in position.

For each maneuver, the raw data has been altered such that the maneuver starts at the origin, and the reference heading is along the x axis. This allows motion in the y-direction to be viewed as cross-track error, and motion in the z-direction to be viewed as altitude error. For easier comparisons between experiments and simulations, the horizontal axis in

the figures corresponds to the x axis in flight, not time. In order to see the transition into the maneuver, two meters of level flight are shown prior to the initiation of the maneuver (i.e. the plots starts at $x = -2\ m$). To evaluate the performance of the attitude tracker, the orientation angles (roll (ϕ), pitch (θ), and yaw (ψ)) are displayed along with both the reference and augmented reference orientation.

In each environment, the aircraft is commanded to perform the same maneuver, for the same distance covered. The augmented reference orientation for each environment differs because the position controller uses errors in position to determine the augmented reference orientation, and each environment has different position errors.

4.4.1 Hover

4.4.1.1 Indoors

As shown in Fig. 4.8, the hover maneuver is performed efficiently in indoor experiments and both simulations. In the indoor flight, the maneuver incurred $\approx 0.5\ m$ increase in altitude, less than $0.5\ m$ of cross-track error, while coming to a stop in about $3.5\ m$. The results in the traditional simulation are similar, but in the HIL simulation, the aircraft has more cross-track error. The poorer performance in the HIL is likely due to the HIL being run on a non-real-time operating system, which can lead to the control loop operating at a lower frequency than expected.

Turning our attention to the pitch results, we note that the augmented reference pitch angle first jumps to 90° at $x = 0$, and then gradually decreases as the aircraft moves further from the reference position (the origin), since the aircraft is commanded to pitch further backwards. Beyond $x \approx 2.5\ m$, the pitch climbs to and then remains around 90°. The roll and yaw angles are not shown because they are non-intuitive due to the singularity of this Euler angle representation when the pitch is near 90°.

4.4.1.2 Outdoors

As shown in Figs. 4.9 & 4.10, we successfully transition from level flight to hover at both 5 $\frac{m}{s}$ and 9 $\frac{m}{s}$, and hold the hover for 10 s in the conventional simulator, in the HIL simulator, and in the outdoor flights. The position control is noticeably worse in comparison to both simulations, and the indoor flights, likely because the wind has a large

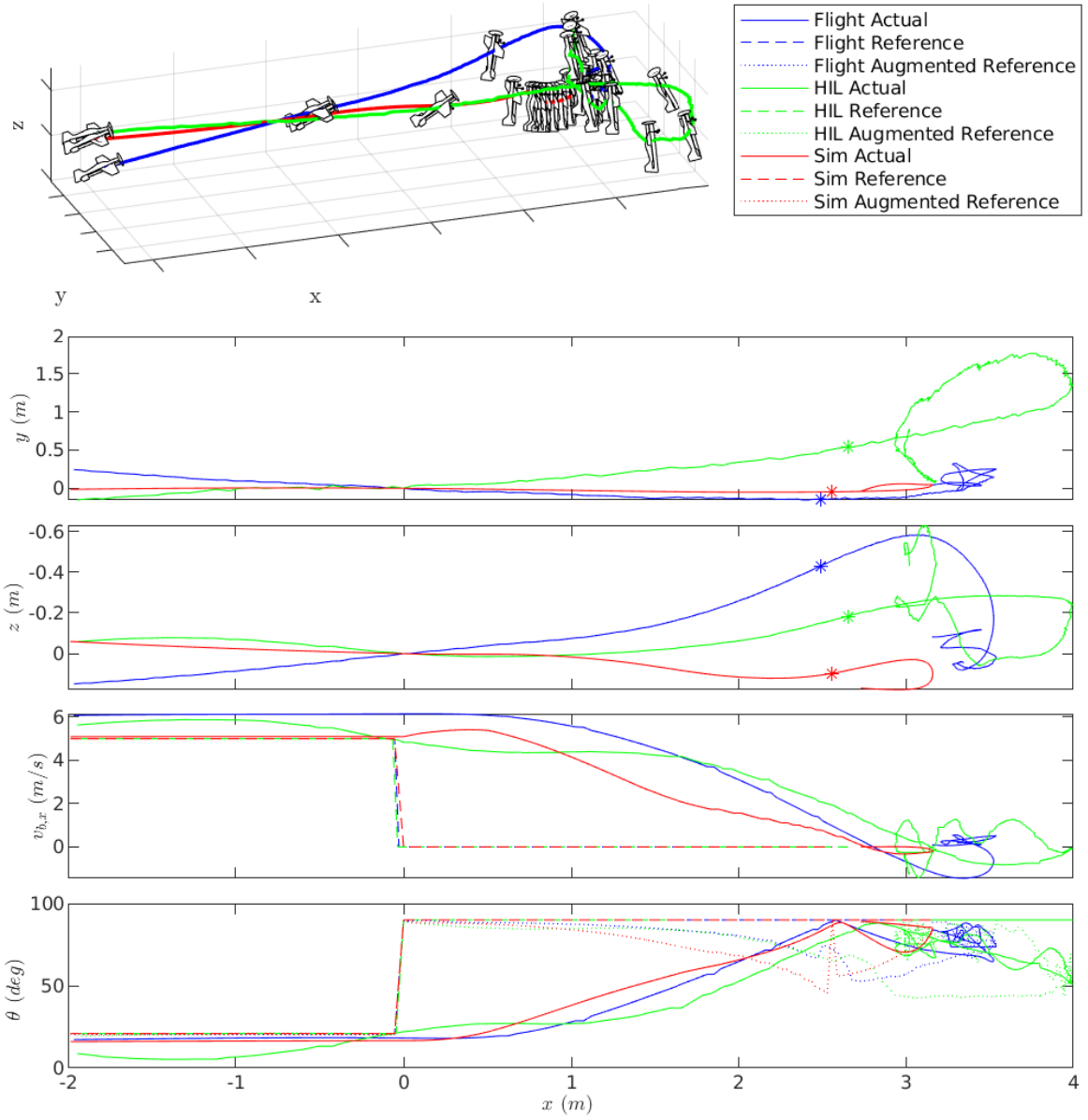


FIGURE 4.8: Indoor Hover

effect on a hovering aircraft, as the wings generate large drag, and there is no actuator to directly oppose the wind.

The aircraft traveling at 5 $\frac{m}{s}$ pitches up into a hover, and as the wings become more exposed to the wind gusts, the aircraft gets pushed away from the reference position, with about 5 m cross-track error, 3 m in the direction of flight but with less deviation in altitude. The aircraft eventually finds an orientation which points its nose into the wind, and remains stationary. As seen in Fig. 4.9, the pitch angle reduces to $40° - 50°$ to

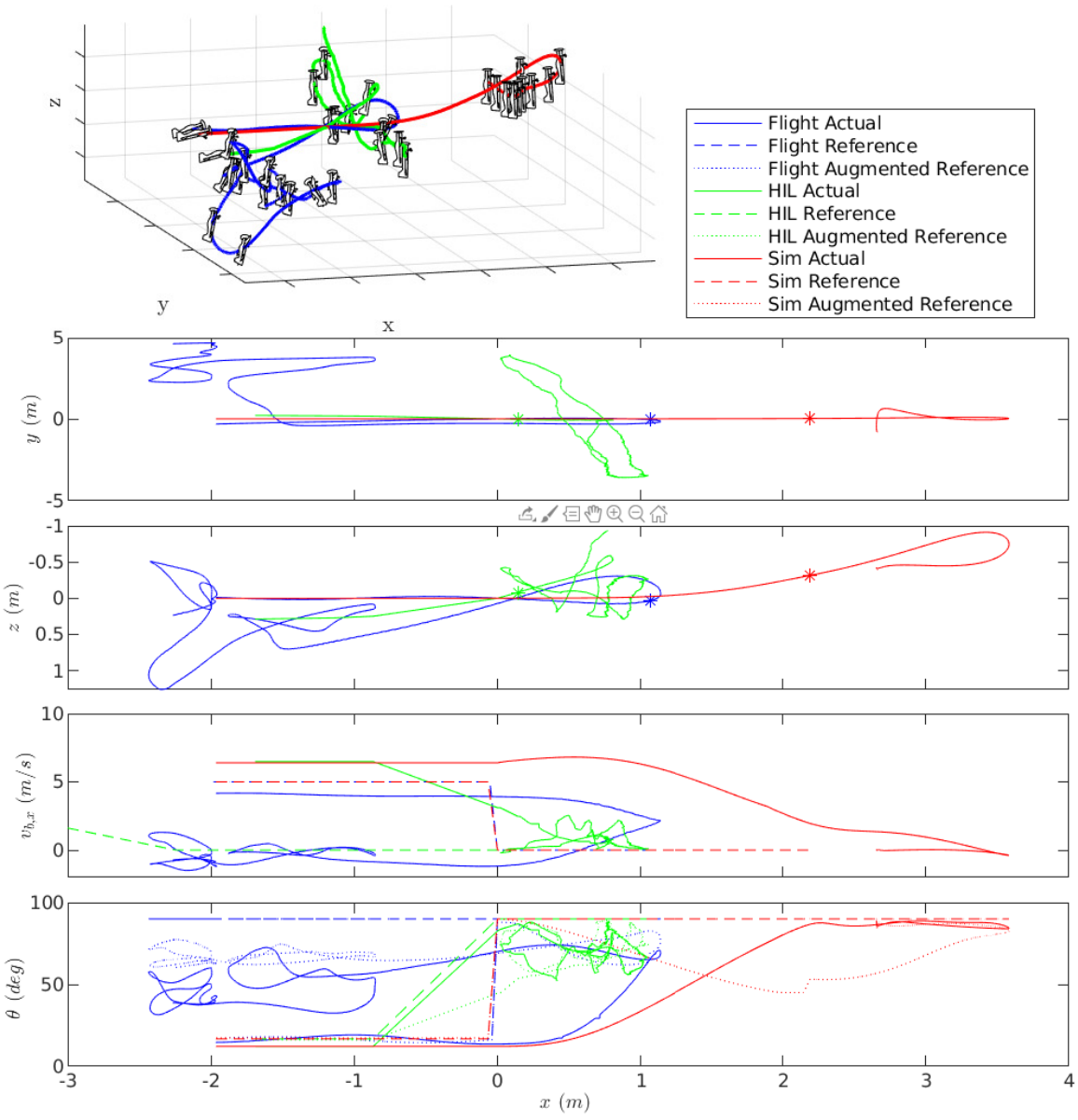


FIGURE 4.9: Outdoor Hover at 5 $\frac{m}{s}$

counteract the wind, whereas in both simulations without wind, the pitch remains closer to 90°.

For the transition to hover at 9 $\frac{m}{s}$, it is likely the winds were calmer in the x-direction as the aircraft does not get pushed backwards, and the steady-state pitch angle is higher. However, there is 4 m of cross-track error, and the aircraft climbs 3 m. The aircraft climbs higher when transitioning from a faster speed, as there is more kinetic energy to dissipate.

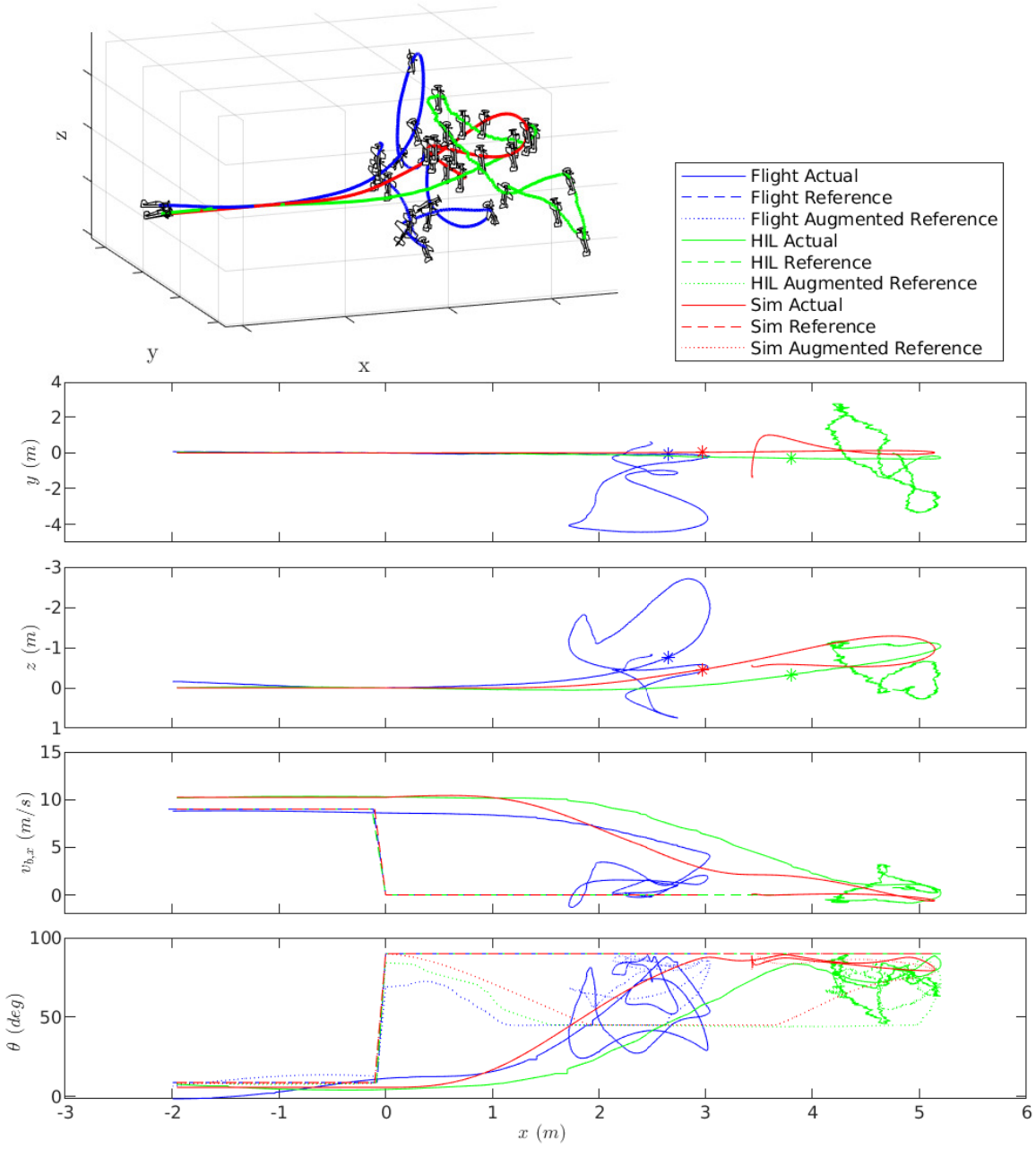


FIGURE 4.10: Outdoor Hover at 9 $\frac{m}{s}$

4.4.2 Aggressive Turnaround

Figs. 4.11 - 4.13 show that the control system performs the aggressive turnaround effectively during both simulations and all the flight tests. Indoors, the aircraft is able to reverse the heading of the aircraft in about 2 s, covering $\approx$ 2 m in the direction flight, with $\approx$ 2 m of cross-track error, and a 1 m gain in altitude during the maneuver, followed by a drop of 1 m at the end of the maneuver. The results for the traditional and HIL

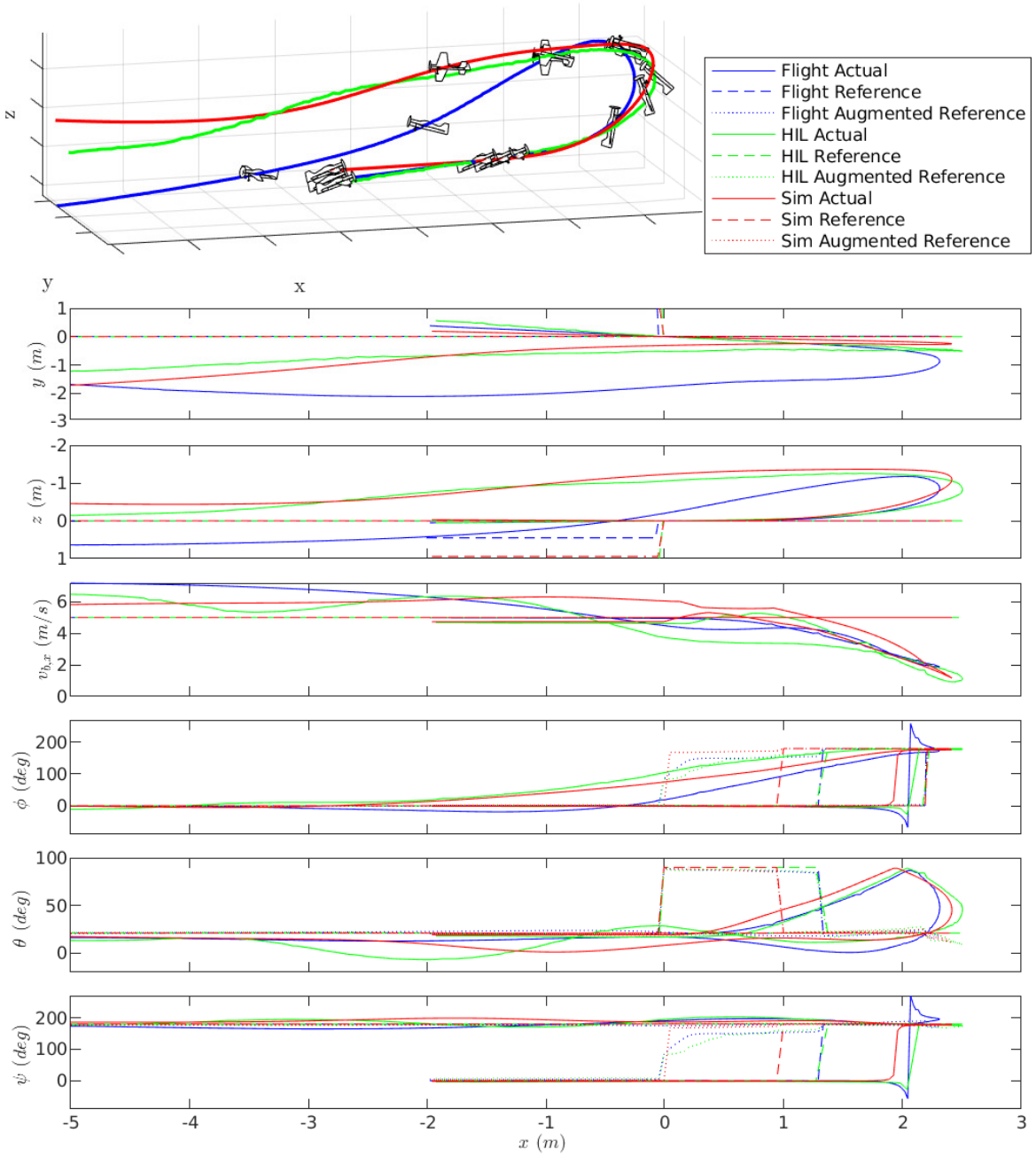


FIGURE 4.11: Indoor Aggressive Turnaround

simulations are quite similar to the experimental results, though they drop less in altitude at the end of the maneuver. During outdoors experiments, at both speeds, the aircraft is able to reverse the heading of the aircraft in about 1 s, covering ≈ 2 m in the direction flight, with ≈ 0.5 m of cross-track error, and a 1 m gain in altitude during the maneuver. It is somewhat surprising that the aircraft turns around in the same distance when traveling at a higher speed, as one would expect the aircraft to travel further and have a larger increase in altitude, as is the case in the conventional and HIL simulations. A

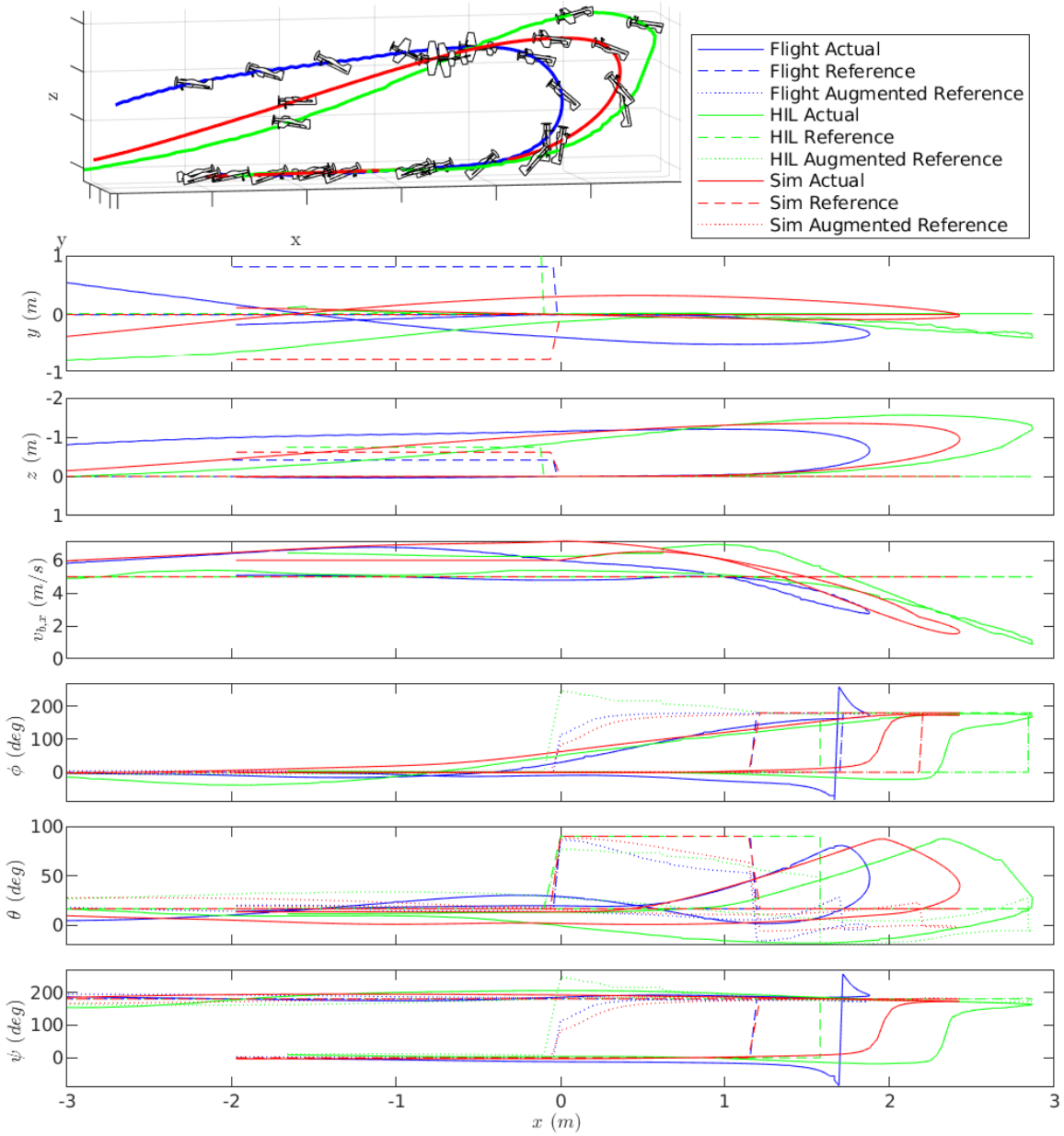


FIGURE 4.12: Outdoor Aggressive Turnaround at 5 $\frac{m}{s}$

plausible explanation could be that the aircraft was flying against a stronger wind during the 9 $\frac{m}{s}$ case.

Turning our attention to the orientation results, the pitch angle initially climbs to 90°. At this point, the roll and yaw angle instantaneously jump to 180° due to the Euler angle convention. The pitch angle then decreases, followed by the roll angle going from 180° to

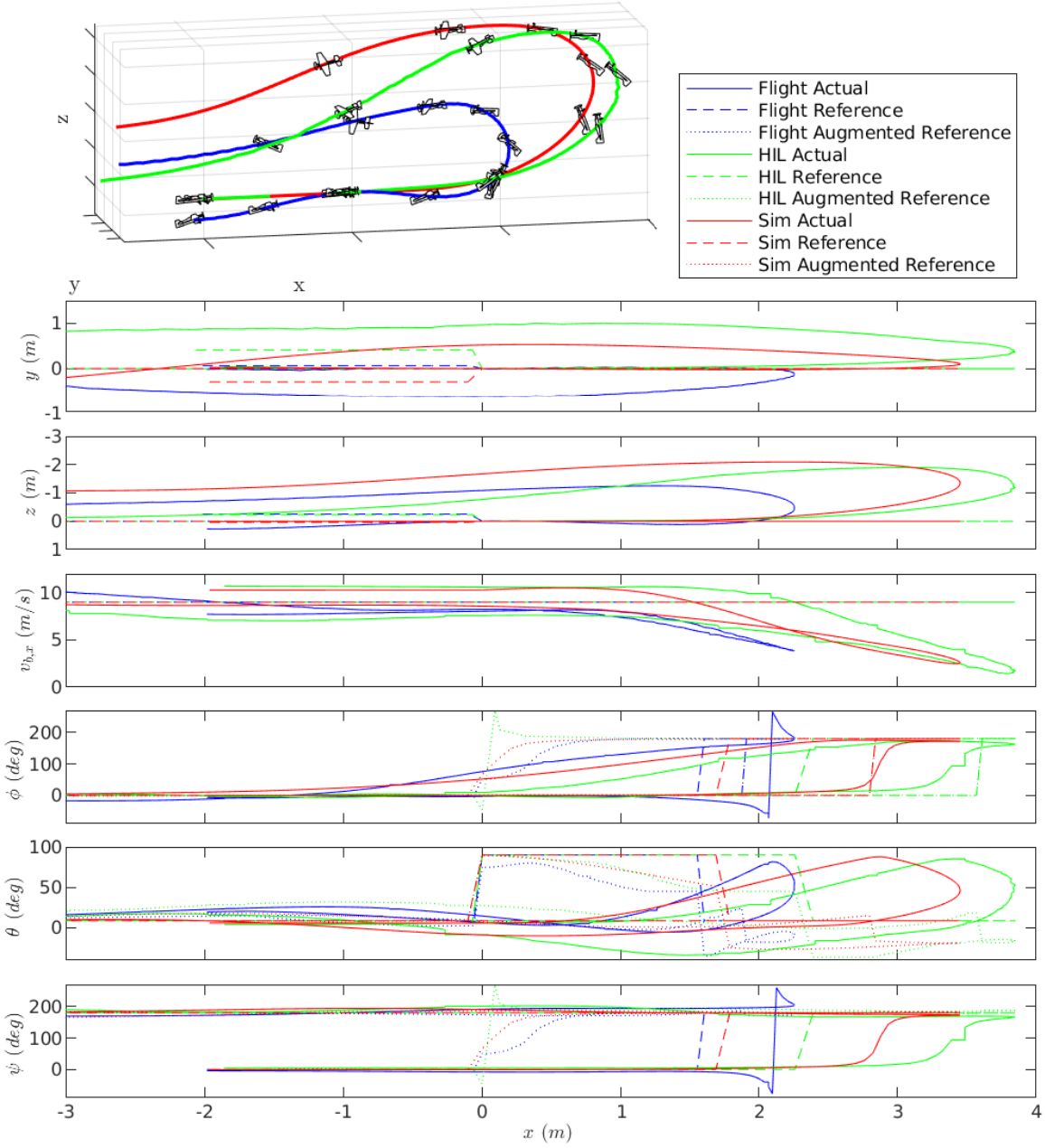


FIGURE 4.13: Outdoor Aggressive Turnaround at 9 $\frac{m}{s}$

0°. At the end of the maneuver, the aircraft is in level flight with reversed heading. This motion is similar during all the simulations and all the flight tests.

4.4.3 Knife-Edge

As shown in Fig. 4.14, during the indoor flight the aircraft takes about 5 m (< 1 s) to roll into the knife-edge, with about 1 m of altitude and cross-track error during flight. In

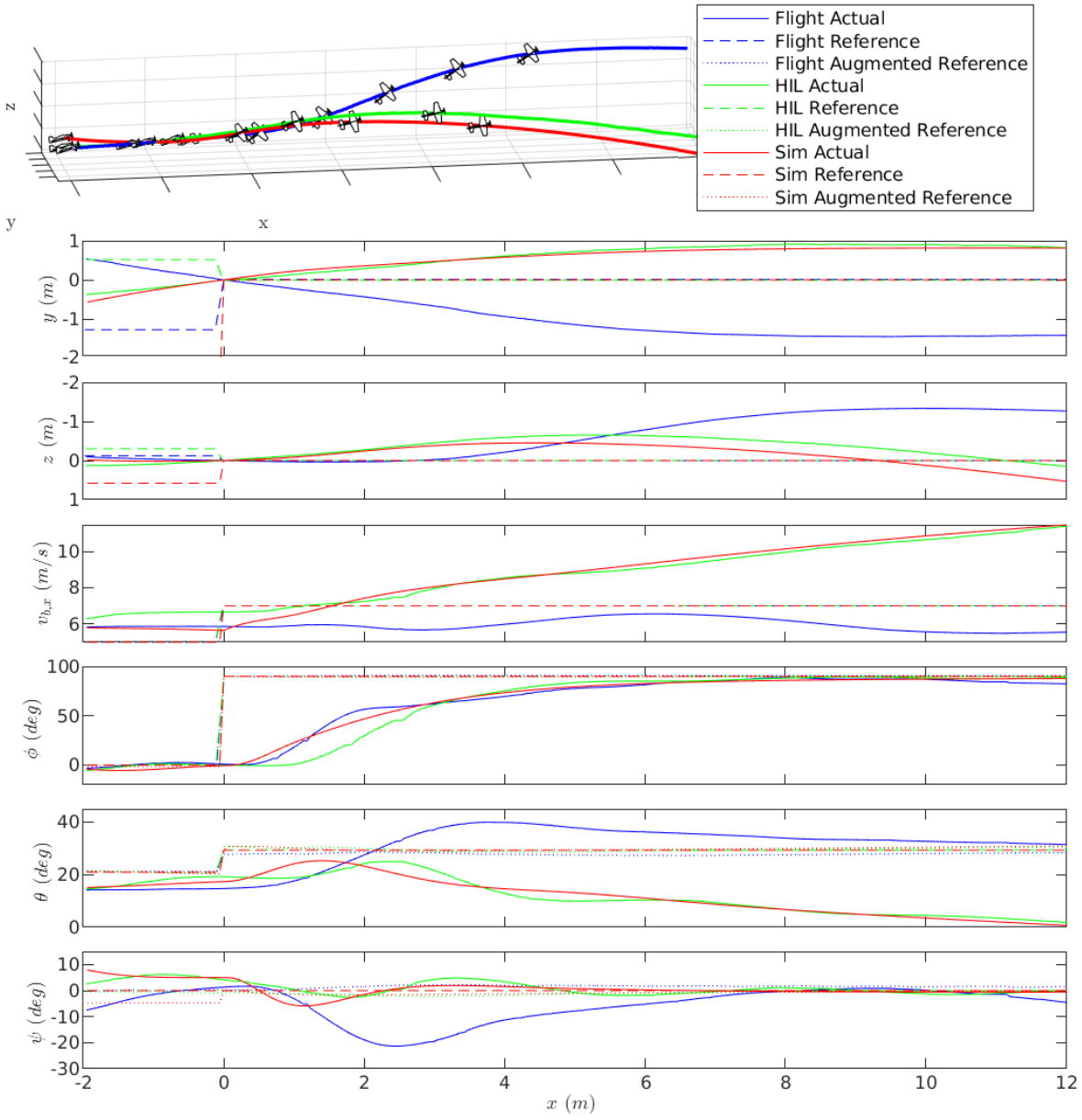


FIGURE 4.14: Indoor Knife-Edge

Fig. 4.15, when the aircraft is traveling at 5 $\frac{m}{s}$ during the outdoor flight, the aircraft takes about 1.8 m ($\frac{1}{3}s$) to roll into the knife-edge, with about a 1 m drop in altitude and less than 2 m cross-track error during flight. At 9 $\frac{m}{s}$ outdoors, shown in Fig. 4.16, the aircraft takes about 2.5 m ($\frac{1}{4}s$) to roll into the knife-edge, with an initial 1 m climb followed by a 1 m drop in altitude, and 1 m cross-track error during flight. We expect a faster traveling aircraft to perform the roll in less time, as the aircraft has more control authority, but also expect this roll to occur over a greater distance, as the aircraft is moving faster.

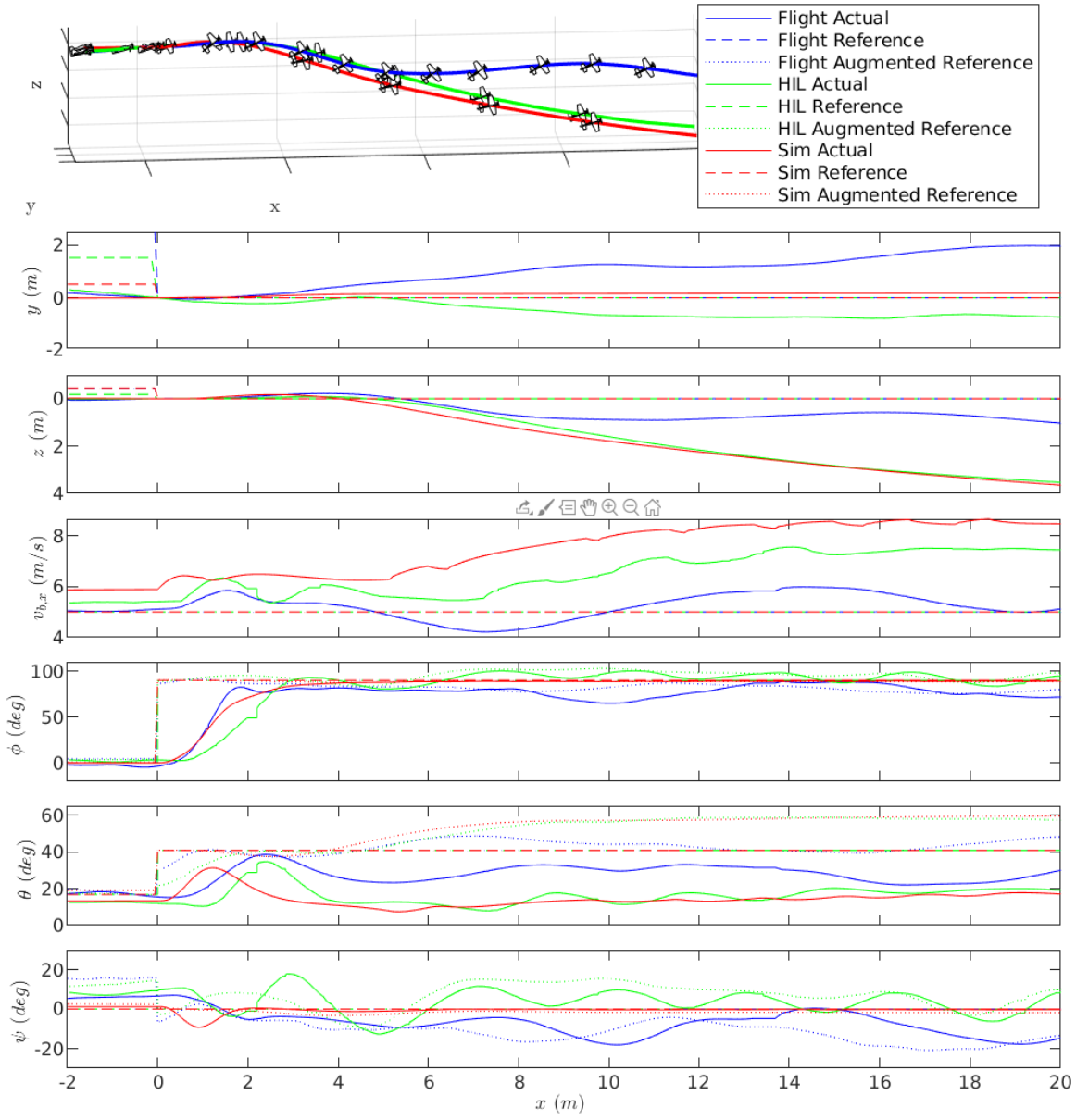


FIGURE 4.15: Outdoor Knife-Edge at 5 $\frac{m}{s}$

During flight testing, once the maneuver is in steady state, all variables stabilize to near-constant values in each testing scenario. The experimental results are mostly consistent with the conventional and HIL simulations. One noticeable difference between simulation and experiment is the aircraft's pitch angle, which also affects the longitudinal speed and altitude. In simulation, the aircraft has difficulty pitching up. This initially lead to executing the knife-edge maneuver with a 7 $\frac{m}{s}$ reference longitudinal speed because the aircraft could not pitch up enough at slower speeds; and using a feedforward rudder

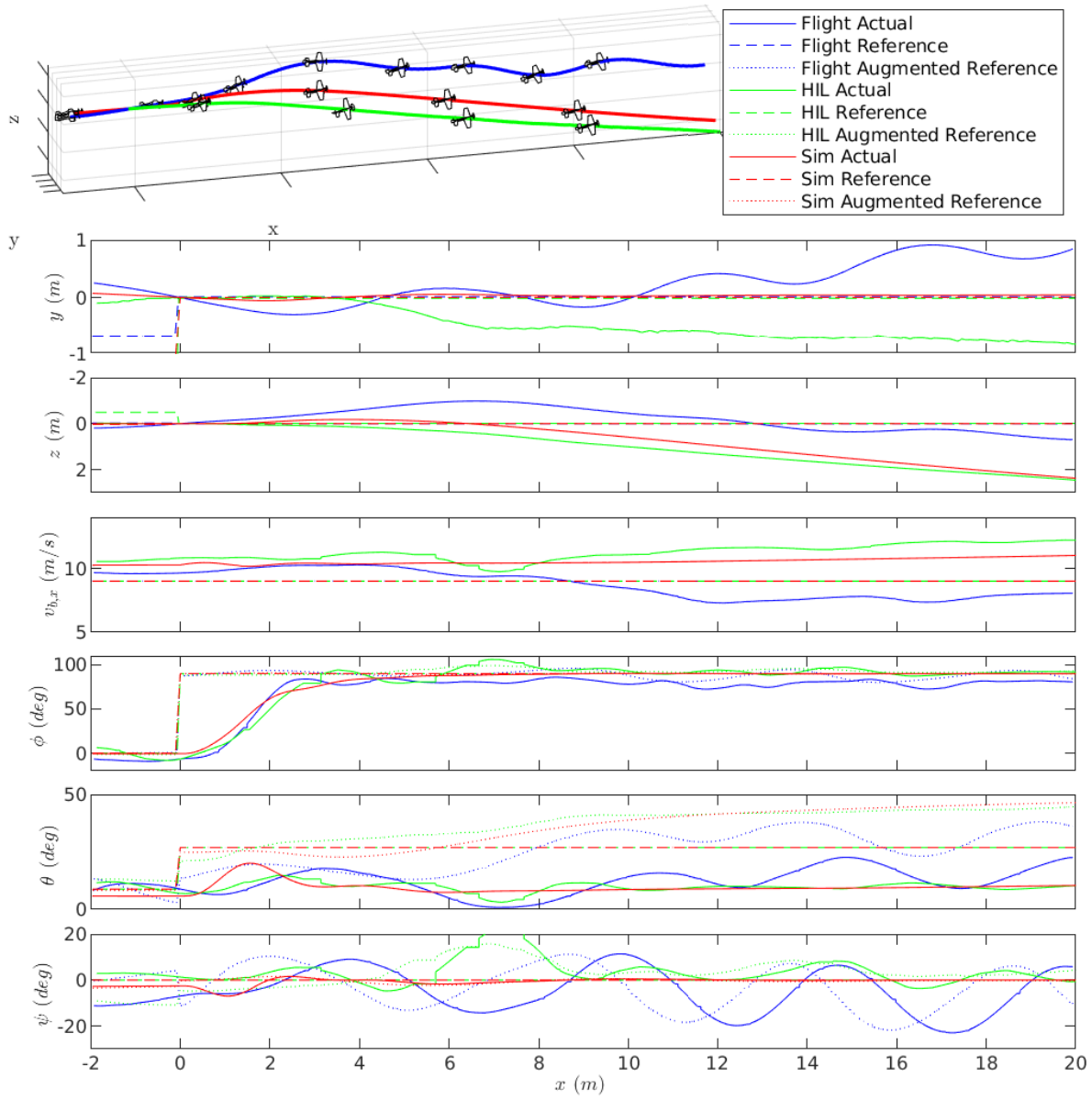


FIGURE 4.16: Outdoor Knife-Edge at 9 $\frac{m}{s}$

input to keep the aircraft pitched up. Modelling inaccuracies lead to the feedforward term dominating the motion, and ultimately pitching the aircraft too much during the indoor experiment. Prior to the outdoor flight testing, this feedforward component was removed, and pitch control during the outdoor flight test is improved. Removing this feedforward term subsequently causes even worse pitch control in the simulations corresponding to the outdoor testing, which leads to the aircraft dropping in altitude.

4.4.4 Rolling harrier

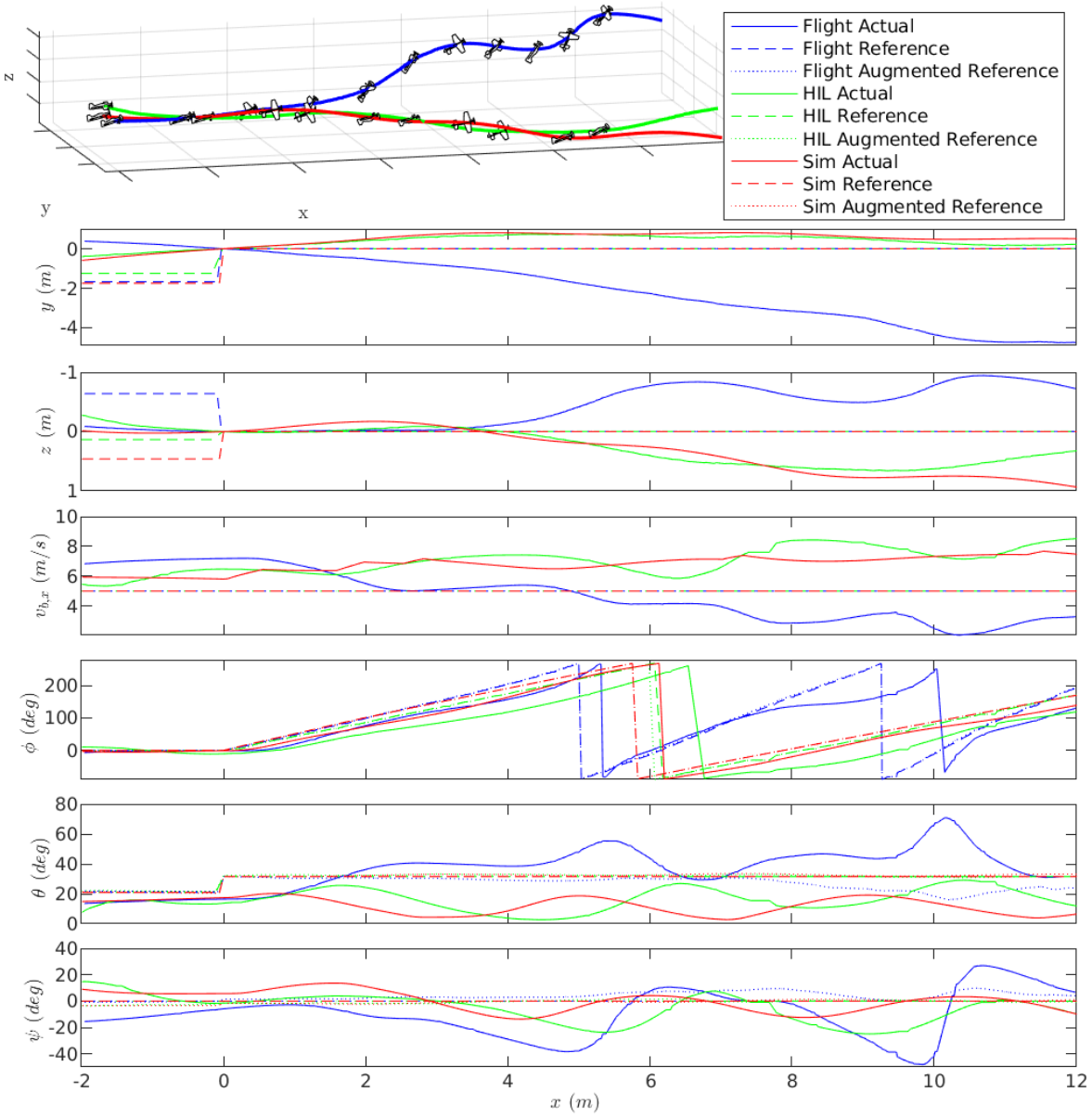


FIGURE 4.17: Indoor Rolling Harrier

We successfully perform a rolling Harrier maneuver with a desired roll rate of $\Omega = 5\ \frac{rad}{s}$, indoors at 5 $\frac{m}{s}$ and outdoors at both 5 $\frac{m}{s}$ and 9 $\frac{m}{s}$ shown in Figs. 4.17 - 4.19. The indoor flight test is accomplished with minimal altitude error, while having about 5 m of cross-track error. At 5 $\frac{m}{s}$ outdoors, the flight has about 4 m of cross-track error and less than 2 m drop in altitude, while at 9 $\frac{m}{s}$, the flight has about 1 m of cross-track error and 1 m

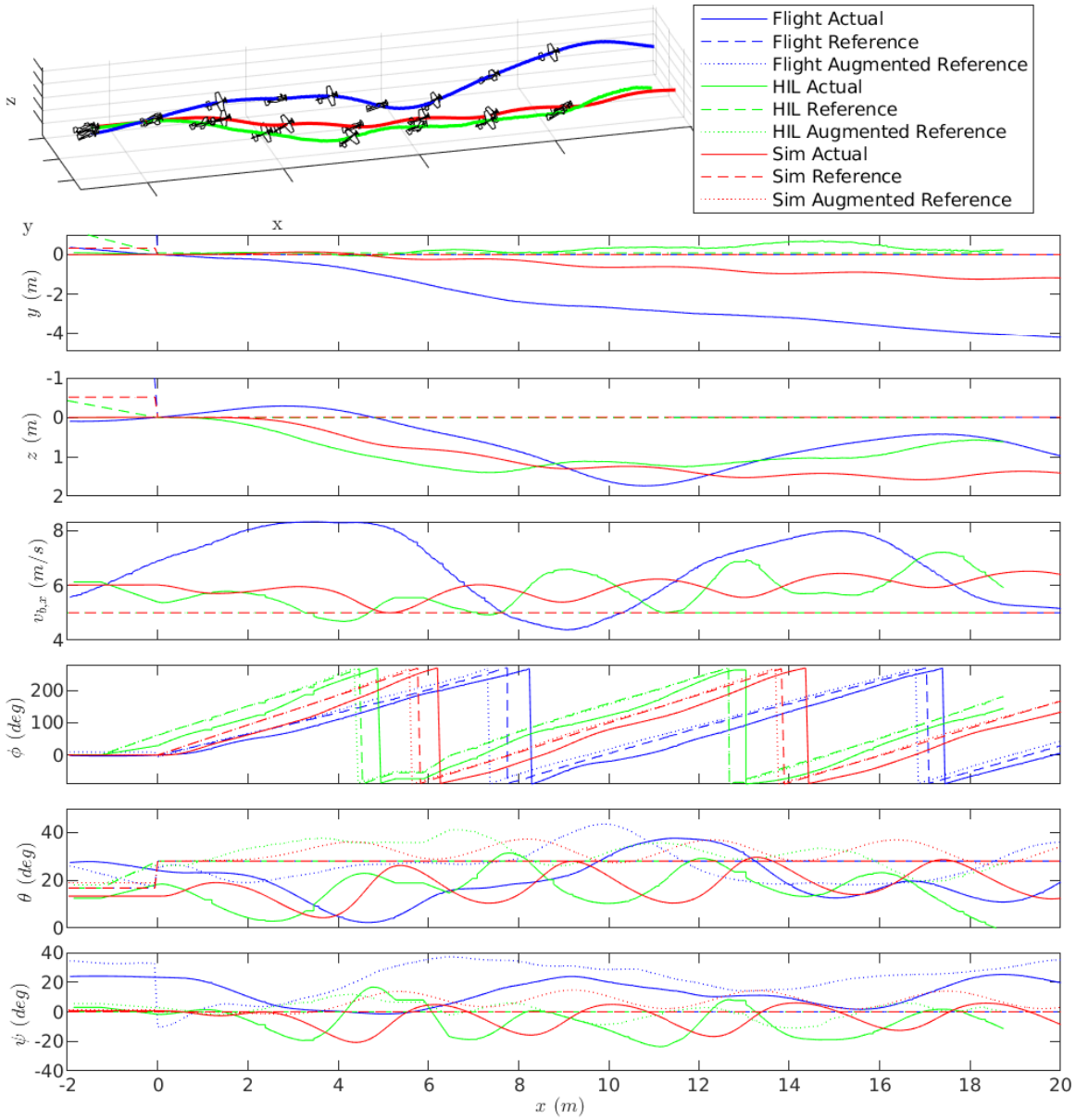


FIGURE 4.18: Outdoor Rolling Harrier at 5 $\frac{m}{s}$

drop in altitude. For both speeds, the behavior in the conventional and HIL simulations are similar to experiment, with the exception that the cross-track error at 5 $\frac{m}{s}$ is less in simulation. In each testing scenario, the roll angle lags the desired roll, but by less than 0.1 s. Overall, the control system efficiently flies the autonomous rolling Harrier in each testing scenario.

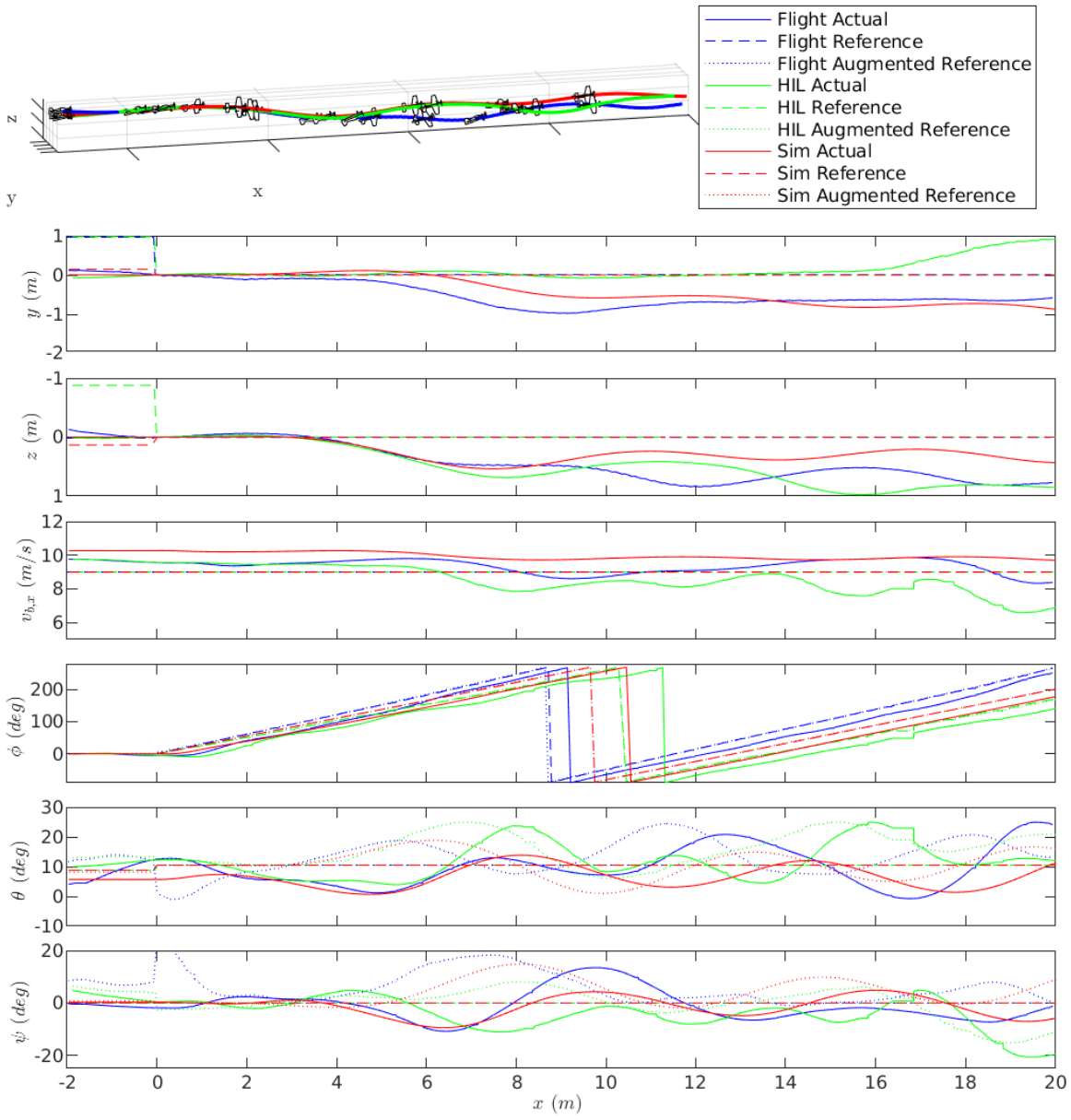


FIGURE 4.19: Outdoor Rolling Harrier at 9 $\frac{m}{s}$

4.5 Extension to Other Platforms

We validate the extension of our control approach to other platforms by demonstrating autonomous acrobatic flight of a quadrotor in simulation and in outdoor flight tests. In addition, variants of the control approach have been used by colleagues to control a quadrotor with bidirectional thrust in [5], a tailsitter in [99], and a passively-coupled hybrid aircraft in [100].

4.5.1 Quadrotor Simulation

We first test the control algorithm in a Matlab/Simulink simulation environment using the modelling techniques described in [101]. To demonstrate the versatility of the controller, we prescribe a rolling flip as the reference trajectory, which has been designed heuristically and is *not* completely dynamically feasible. Furthermore, we use the *same* gains from the outdoor flight experiments with the agile fixed-wing aircraft, shown in Table 4.2.

The motion of the quadrotor is shown in Fig. 4.20a, where the top of the quadrotor is red, the bottom is blue, and the trajectory line starts as blue and becomes lighter with time. For more detailed results, we show the time histories of pose and reference pose in Fig. 4.20b. It is worth emphasizing that the reference pose is not completely dynamically feasible, and thus the tracking of the reference pose is not perfect, although we are still able to achieve the higher-level goal of each maneuver.

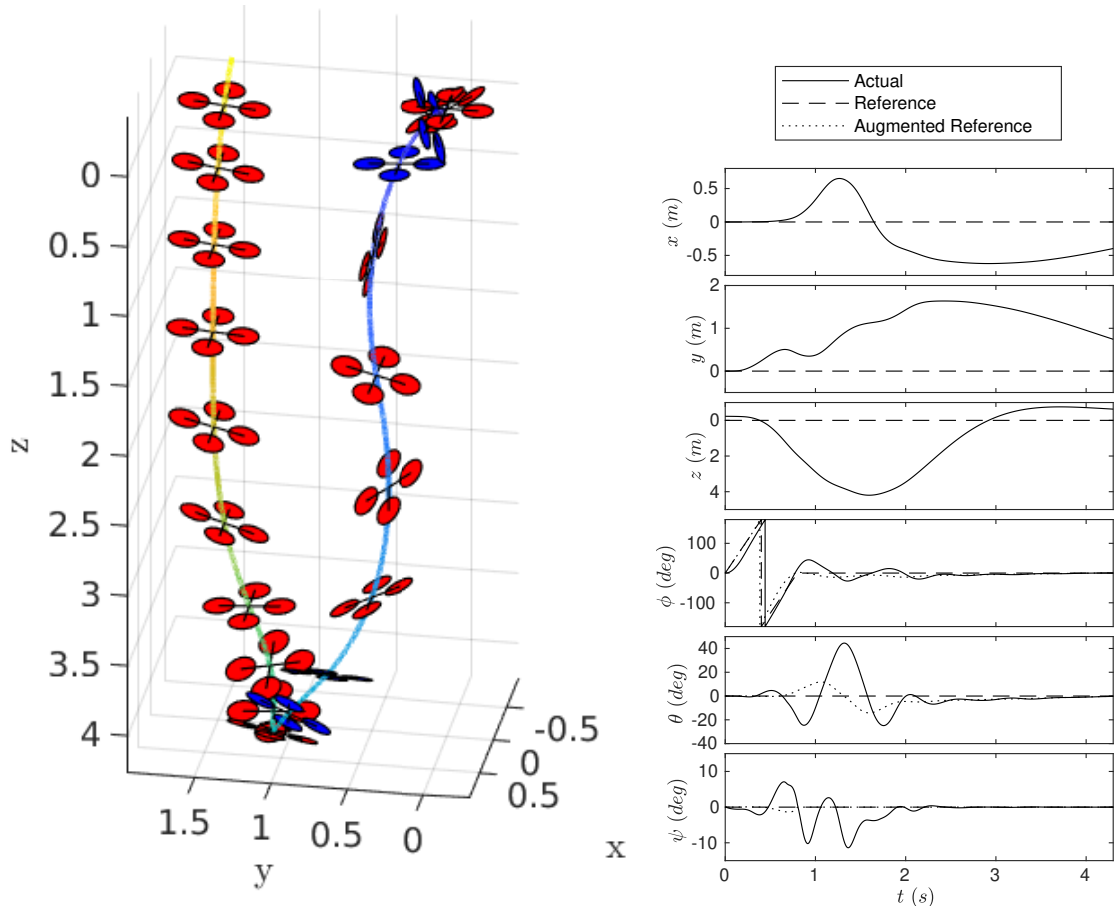
We command a fixed position, pitch, and yaw, while commanding the roll angle to increase at a rate of $2.5\pi \ \frac{rad}{s}$ for one flip. This is obviously not a feasible trajectory as rolling the quadrotor will cause the quadrotor to move laterally and drop in altitude.

However, as shown in Fig. 4.20b, the quadrotor is able to perform the rolling flip, dropping about 4 m in altitude while the quadrotor's thrusters are pointing downwards, and moves laterally about 0.5 m in x and 1.5 m in y. Once the flip is achieved and the quadrotor is stabilized, the quadrotor directs itself back to the reference position and remains stationary.

A hardware-in-the-loop (HIL) simulation is then used to initially test the hardware implementation, prior to testing on the real vehicle. We use the built-in PX4 HIL simulation using RotorS/Gazebo [102], with slight modifications to reflect our systems' inertial and thruster characteristics. In the HIL simulation, we performed the same maneuver as in the conventional simulation, and obtained similar results. These results are not shown here, for brevity. We focus instead on presenting the experimental results which we consider to be more important.

4.5.2 Quadrotor Experiment

After satisfactory results were obtained in the HIL simulations, we performed outdoor autonomous flight tests with the Pleiades Spiri, depicted in Fig. 3.7. The Spiri quadrotor has a mass of 1.02 kg and a thrust-to-weight ratio of 2.9. All of the sensing and computation is done on-board, using a Pixhawk flight computer running the open-source



(a) Motion (b) Data

FIGURE 4.20: Quadrotor Simulation

PX4 flight stack. State estimates of both UAVs are obtained from the default Extended Kalman Filter (EKF2) in PX4 that fuses the IMU, barometer, and GPS measurements. The control loop is executed at 200 Hz.

Our quadrotor flight tests took place at the McGill University Forbes Field, a large outdoor soccer field in Montreal, Canada. We only tested autonomous airborne maneuvers—we manually flew the vehicle up to altitude, and then put the vehicle in autonomous mode, performed various maneuvers autonomously, and then manually landed the vehicle. We recorded flight data over numerous tests and days, with varying wind conditions. We were able to autonomously execute several maneuvers, and were able to perform these maneuvers reliably and repeatedly. We specifically analyze two cases of of rolling flips, both attempted on a day with winds of 4-6 knots [98].

While the *same* gains used in the outdoor fixed-wing flights could be used for the quadrotor experiments, which is done in simulation, better performance is achieved by tuning gains due to the uncertainty in the allocation of control forces and moments to actuator

signals. We change the attitude control gains relative to Table 4.2: the proportional $\mathbf{K_{a_p}} = 110\ \mathbf{1}_{3x3}$, and the derivative $\mathbf{K_{a_d}} = 20\ \mathbf{1}_{3x3}$. The rest of the control gains are kept the same as in the quadrotor simulation (and fixed-wing outdoor flight).

4.5.2.1 Quadrotor Rolling Flip Case 1

We successfully perform a rolling flip with a quadrotor, which is shown in https://youtu.be/VJTqTj8c5s8, as well as a sequence of overlayed images in Fig. 4.21.

FIGURE 4.21: Quadrotor Rolling Flip Case 1 Image Sequence: The flip begins on the right side of the image, performs the flip while losing altitude, and then returns to the start of the maneuver

For quantitative analysis of the maneuver we refer to Fig. 4.22 which compares the state estimates of position and orientation to the reference values, and also shows the control inputs. The control inputs are shown with normalized values, where zero corresponds to the idle motor PWM signal, and one corresponds to the maximum motor PWM signal.

We show one second of hovering flight, and then start the flip maneuver at $t = 0s$. Our reference trajectory is formulated by setting a constant reference position and yaw angle to that at the start of the maneuver (causing the step in reference position at $t = 0s$), a zero pitch angle, and a roll angle which starts at zero, has a constant acceleration, followed by a coast at maximum velocity, concluding with a braking phase of constant deceleration until the roll angle reaches 360 degrees.

This reference trajectory is not dynamically feasible, as it is not possible to perform a flip while remaining in the same position, although we are still able to achieve the higher-level requirements of the maneuver. As shown in Fig. 4.22, the quadrotor drops about 3 m in altitude while the quadrotor's thrusters are pointing downwards, and moves laterally

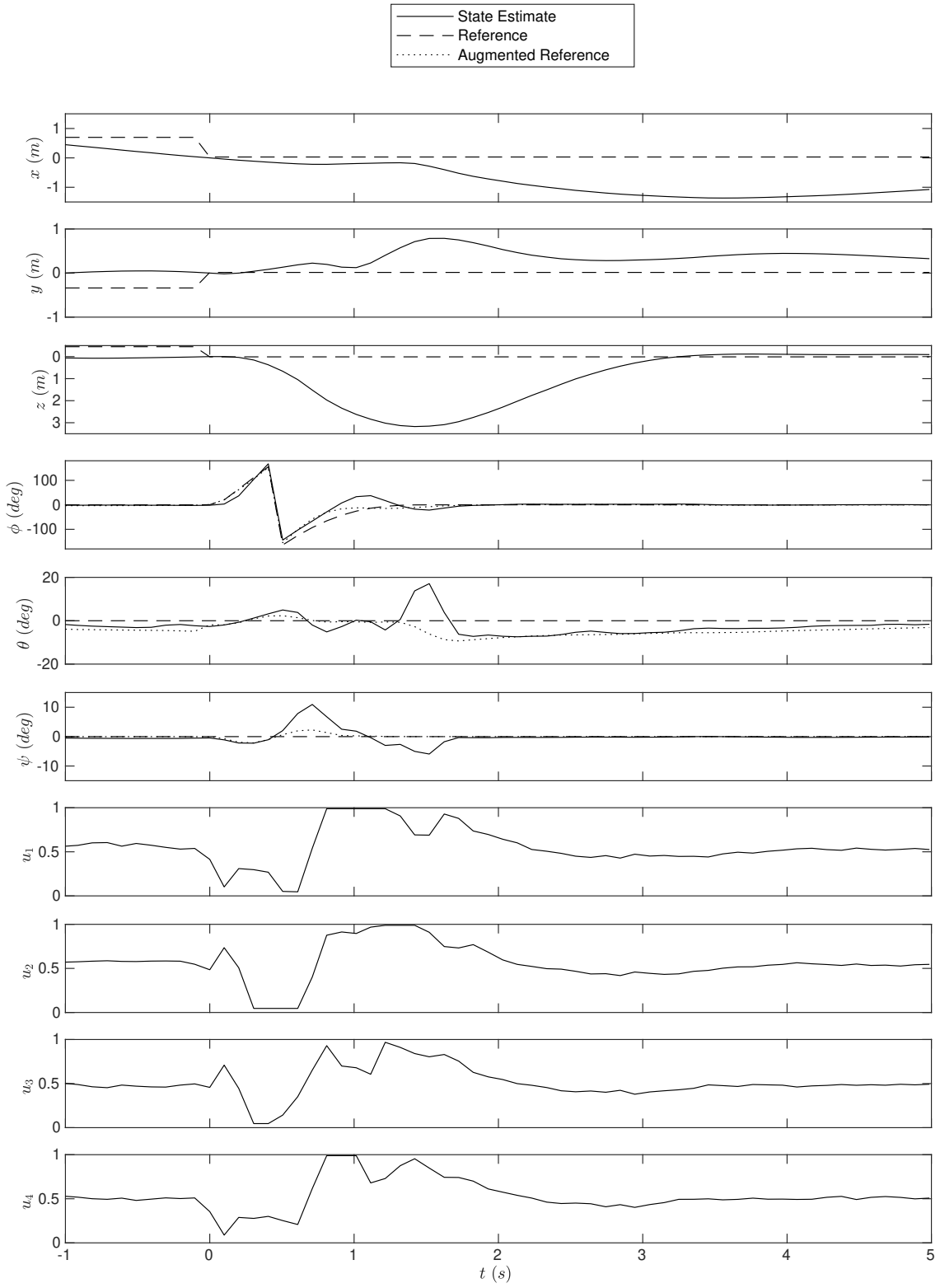


FIGURE 4.22: Quadrotor Rolling Flip Case 1 Flight Data

about 1 m in x and in y. Once the flip is achieved and the quadrotor is stabilized, the quadrotor directs itself back to the reference position and remains stationary. There is a small steady-state error in the x & y position, which can be attributed to the moderate wind gusts and lack of integral term in the control law.

Considering the aggressiveness of the flip maneuver, the attitude is tracked very well. During the flip, the maximum pitch and yaw error remain less than 20° and 10° respectively. The roll angle initially lags the reference, which is followed by a slight overshoot at the end of the flip. After the flip is complete, all of the Euler angles are tracked with less than 2° error.

The actuators behave as expected, where initially the speed of rotors two and three increase, while rotors one and four decrease, causing a moment in the x direction needed to roll the vehicle. As the flip is completing, a moment in the opposite direction is needed to reduce the angular velocity, causing rotors two and three to decrease speed, while rotors one and four are increased.

4.5.2.2 Quadrotor Rolling Flip Case 2

We aim to perform the same rolling flip maneuver described in Sec. 4.5.2.1, except this time with a more aggressive roll reference, which ultimately speeds up the execution of maneuver, making the quadrotor lose less altitude. Two changes were made to achieve this more aggressive flip.

First, we change the proportional attitude gain to $\mathbf{K_{a_p}} = 110\ diag(1, 1, 0.2)\frac{rad}{s^2}/rad$, and the derivative attitude gain to $\mathbf{K_{a_d}} = 20\ diag(1, 1, 0.2)\frac{rad}{s^2}/\frac{rad}{s}$. This change keeps the gains associated with the roll and pitch axes the same, but reduce the gains associated with the yaw axis by a factor of 5. This reduction in yaw gain is theoretically not necessary, however, due to the simple control allocation strategy used for our quadrotor it is necessary to lower the yaw gains to prevent the motors from saturating for too long during this more aggressive flip, and effectively losing control authority in roll and pitch.

The second change made to turn off the position controller while the quadrotor is flipping, and turning it back on once the roll reference is back to zero. This is done in [21] and is an example of one of the benefits of a modular control architecture—the ability to easily turn on and off the position controller as discussed in Sec. 3.1. In this example, the simple reference trajectory in position contradicts the reference attitude at times, and during these times the position controller can degrade the performance of the maneuver. For example, initially the reference roll increases, causing the aircraft to roll and increase in

y position, but then the position controller wants to decease the y position, making the augmented reference roll less than reference roll, ultimately slowing the flip down.

Both of these changes allow the quadrotor to track a faster and more aggressive flip. This maneuver is also shown in https://youtu.be/VJTqTj8c5s8, as well as a sequence of overlayed images in Fig. 4.23.

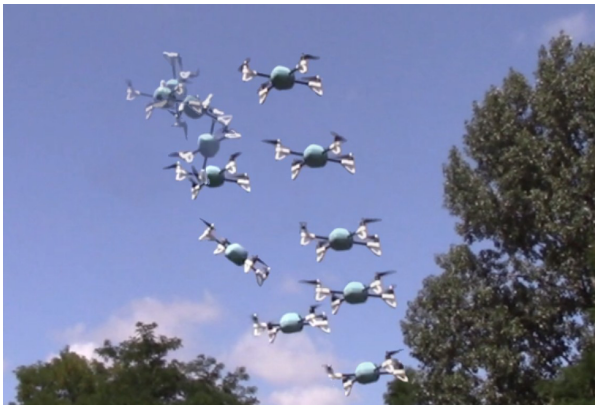


FIGURE 4.23: Quadrotor Rolling Flip Case 2 Image Sequence: The flip begins on the left side of the image, performs the flip quickly, loses altitude, and then returns to the start of the maneuver

For quantitative analysis of the maneuver we refer to Fig. 4.24. For the most part, the analysis of the maneuver is the same as in Case 1. The major difference is the flip is completed in about 0.75 s opposed to 1 s, reducing the altitude drop to only 1.5 m rather than 3 m.

Turning off the position controller for the duration of the flip can be seen in attitude plots, where the augmented reference attitude is the same as the reference attitude from $t = 0$ s to $t = 0.75$ s. After $t = 0.75$ s the position controller is turned back on, and the reference and augmented reference attitudes are no longer the same.

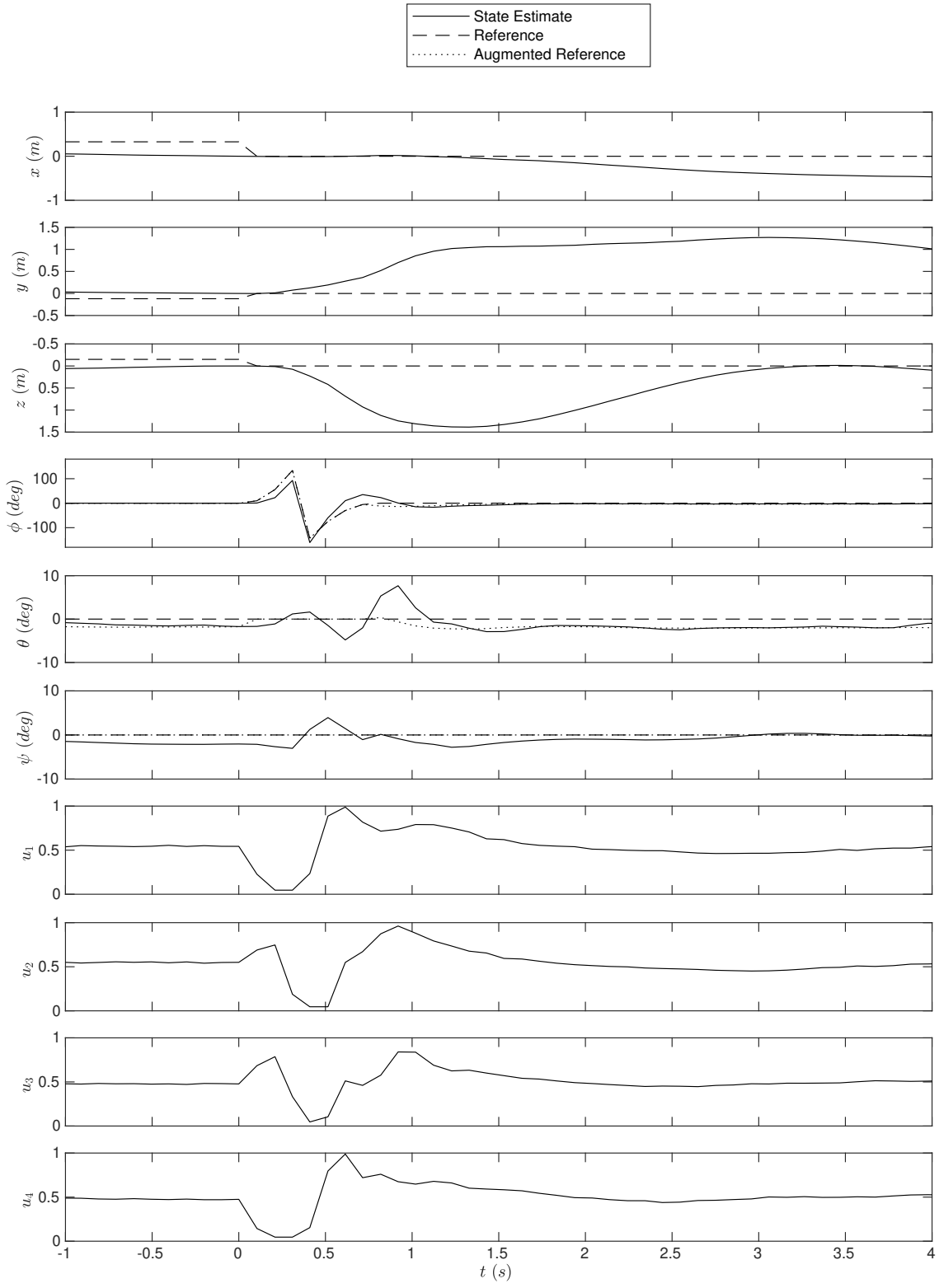


FIGURE 4.24: Quadrotor Rolling Flip Case 2 Flight Data

Chapter 5

Obstacle Avoidance

Navigating a robot through an unknown environment is a complex problem that spans multiple research topics including perception, motion planning, control theory, state estimation, and dynamics modeling. The typical approach consists of a robot moving through an environment with various sensors, which are used to map the environment and localize within it (SLAM). A motion planner then uses this map to generate a collision-free reference trajectory that reaches the goal, and the controller outputs actuator signals to track this reference trajectory.

This problem is significantly more difficult in the context of unmanned aerial vehicles due to their limited payloads and complex six degrees of freedom dynamics. The limited payload restricts the robot's computational power and sensing capabilities. The complex dynamics complicates the motion planning problem when considering real-time implementation with limited computational power. The computational complexity of executing a SLAM algorithm and generating motion plan on an on-board processor results in update rates that are too slow to avoid obstacles during high speed flight.

These issues are the reason autonomous flight through cluttered environments is an active research problem. Most of the success in achieving autonomous high-speed flight in unknown cluttered environments has been demonstrated with rotorcraft; and the issue of limited computational power has been addressed by separating the motion planning problem into two parts: a slower global motion planner that requires a map of the environment to be built while flying, and a faster local reactive motion planner that only uses sensor data [44, 45].

The global motion planner is responsible for finding a collision-free trajectory to the goal if one exists; and the local motion planner is responsible for avoiding collisions with newly

perceived obstacles, and is henceforth referred to as the obstacle avoidance algorithm. The complete hierarchy of the motion planner can be shown in Fig. 5.1. The global planner utilizes the map of the environment, and gives the obstacle avoidance algorithm a local goal, which is a position along the global path. The primary objective of the obstacle avoidance is to avoid obstacles, and reaching the local goal is secondary since the global planner will eventually update this local goal. Furthermore, it is not possible to guarantee the obstacle avoidance will reach the local goal, since it only operates on instantaneous point cloud data.

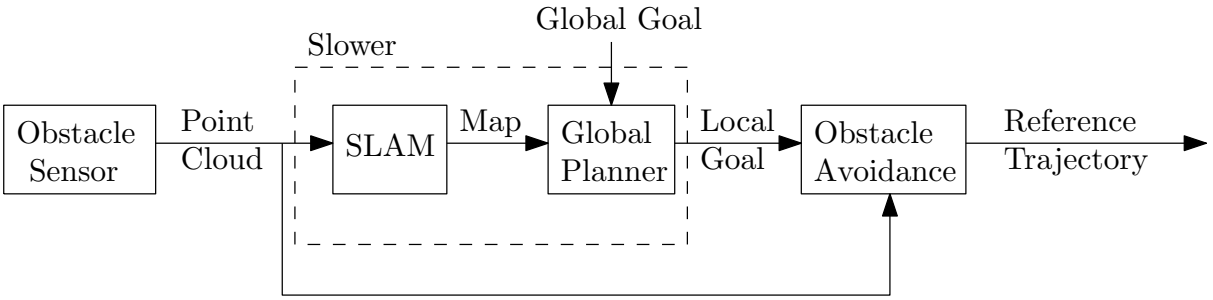


FIGURE 5.1: Block Diagram of Complete Motion Planner

Autonomously flying through unknown cluttered environments with fixed-wing aircraft is even more complicated than rotor-craft due to the nature and complexity of their dynamics. The majority of researchers tackling this problem simplify it by assuming a *known* map of the environment. Previous work within the research group [12] makes this assumption, and focuses on trajectory generation and global motion planning with agile fixed-wing UAVs in static known environments. The global motion planner presented in [12] utilizes a library of motion primitives built off-line, which can then be used in a real-time RRT motion planning framework to generate collision-free dynamically feasible trajectories to a goal region.

To the best of our knowledge, the only work which presents autonomous fixed-wing aircraft flight in a densely cluttered *unknown* environment using only on-board sensing and control is in [79]. While this work is extremely impressive and the most advanced to date, during their outdoor flight testing campaign their aircraft successfully avoided obstacles 16 times, but failed to do so 10 times.

5.1 Obstacle Avoidance Overview

We limit the scope of the motion planning aspect of this thesis to reactive obstacle avoidance, as is done in [79], with the notion that it would be combined with a SLAM

implementation and the global planner in [12] in the future. The scenario addressed in this thesis is represented by a block diagram in Fig. 5.2, where the obstacle avoidance algorithm generates reference trajectories solely based on an instantaneous point cloud and goal location.

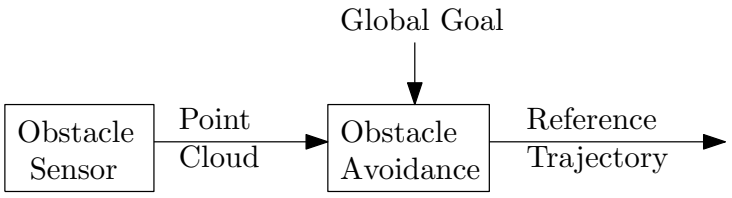


FIGURE 5.2: Block Diagram of Obstacle Avoidance

In this situation, the primary purpose of the obstacle avoidance algorithm is to avoid obstacles, and reaching the goal is secondary, since it only operates on instantaneous point cloud data.

We base our methodology on a point cloud representation of obstacles because most obstacle detection sensors output a point cloud, such as a LiDAR or stereo camera. Our experimentation uses a stereo camera since it is the only sensor that fits within the aircraft's payload capacity.

As mentioned previously, the work in [12] utilizes a library of motion primitives built off-line, which are selected in real-time to navigate through a *known* environment. In this work, we utilize the same library of motion primitives, which is briefly discussed in Sec. 5.2. Unlike in [12], in this work trajectories are selected from the library in conjunction with on-board obstacle detection, which ultimately enable the aircraft to autonomously avoid collisions and reach the goal location in *unknown* environments. This selection process is detailed in Sec. 5.3.

5.2 Trajectory Generation

Many motion planning algorithms propagate the robot's dynamics model several times before obtaining a reference trajectory [103], but this is not feasible in real-time, with limited computational power and a complex dynamics model. The most common approaches are to reduce the complexity of the dynamics model so that it can be executed in real-time or to run the computationally expensive dynamics model off-line, and store it in a library for real-time use. The former approach enables the motion planner to propagate the dynamics as needed, at the cost of inaccurate modeling due to the approximation, while the latter approach enables the motion planner to use the complex dynamics model

but limits the motion plan to only those options that have been stored in the trajectory library.

We have at our disposal a highly accurate, but computationally expensive, dynamics model developed in our research group [9]. As well as existing methodologies pertaining to agile fixed-wing aircraft have successfully utilized a pre-computed trajectory library for real-time global motion planning in known environments in [12], and real-time obstacle avoidance in unknown environments in [79]. We therefore elect to use a pre-computed trajectory library for our obstacle avoidance methodology. The library of motion primitives computed off-line enables the strategy to exploit a large portion of the aircraft's flight envelope without solving complex equations of motion in real-time.

The reference trajectories used for collision avoidance should be dynamically feasible with respect to the modelling discussed Chapter 2. In Chapter 4, we use a 'maneuver generator' to heuristically generate aerobatic trajectories. These trajectories are not dynamically feasible, and thus we elected to not use this approach for generating trajectories for collision avoidance. Instead, we build our library by solving trajectory optimization problems, using the method presented in detail in [12, 56], and briefly discussed in this section.

We obtain our trajectory library using a MATLAB optimal control software, GPOPS-II [104]. Each trajectory in the library is the solution of a trajectory optimization problem that is constrained to the vehicle dynamics model in Chapter 2 and physical actuator limits. We generate two classes of trajectories within the library: finite-time trajectories, referred to as agile primitives, and infinite-time trajectories (or steady-state), which are referred to as trim primitives.

5.2.1 Trim Primitives

In addition to the constraints mentioned above, trim primitives are constrained to constant speed, roll, pitch, and control inputs. They are also constrained to having no sideslip, and the optimization objective is to minimize the control effort. Each trim primitive is defined by its speed, yaw rate and climb rate, which are all constant throughout the trajectory.

Trim primitives make up the majority of the library. Specifying various combinations of yaw rate and climb rate result in straight and level flight, climbs, descents, banked turns, and helical banked turns. For this work, we limit the collision-avoidance strategy to constant speed flight, except during agile maneuvers, and build a library of trim

primitives. In [12], all of the validation is performed using a constant speed of 7 m/s, and motion primitives are obtained with yaw rates of -110 to 110 $^\circ/s$ in increments of 10 $^\circ/s$, and climb/descent rates from -2 to 2 m/s in increments of 1 m/s to form a library of 115 trim trajectories consisting of straight and level flight, 2 climbs, 2 descents, 22 banked turns, and 88 helical banked turns. These ranges of yaw rates and climb rates span the majority of the steady-state flight regime of these aircraft. A top-down view of the motion primitives for different yaw rates with a zero climb rate are depicted in Fig. 5.3, and a side view showing different climb/descent rates with a zero yaw rate are depicted in Fig. 5.4. Fig. 5.5 shows the particular case of a helical turn trim primitive, with a yaw rate of 110°/s and a decent rate of 2 m/s.

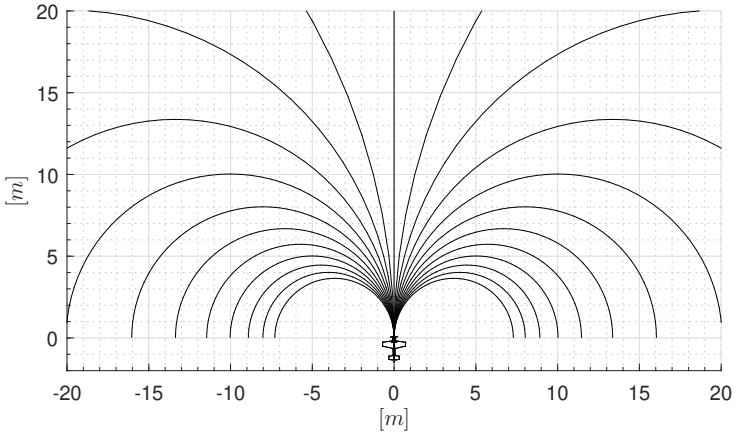


FIGURE 5.3: Trim Primitive Top-down View [12]

The work in this thesis considers operation at various speeds, and thus a trim primitives library is generated for speeds of 7 m/s, 9 m/s, 11 m/s and 13 m/s. Solving these trajectory optimization problems at multiple speeds shows that, as the aircraft is flying faster, the maximum yaw rate decreases. Thus, the full trajectory library of yaw rates from -110 to 110 $^\circ/s$ and climb/descent rates from -2 to 2 m/s could not be found at higher speeds. Further investigation shows that this reduction in maximum yaw rate is caused by throttle saturation, since increasing the maximum thrust by 50% enables the trajectory optimization to find a full trajectory library at all the speeds tested. This phenomenon is consistent with intuition since, at constant speed, a turning aircraft requires more thrust than flying straight, due to the loss of lift while banking. Table. 5.1 summarizes the feasible combinations of yaw rate and climb/descent rate, based on speed.

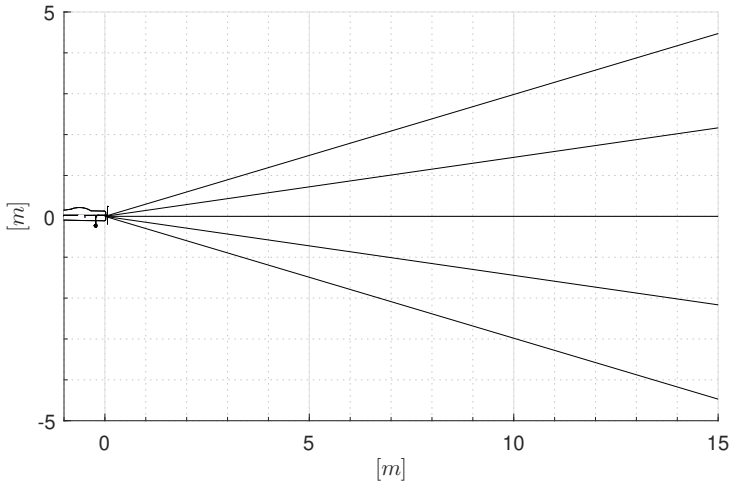


FIGURE 5.4: Trim Primitive Side View [12]

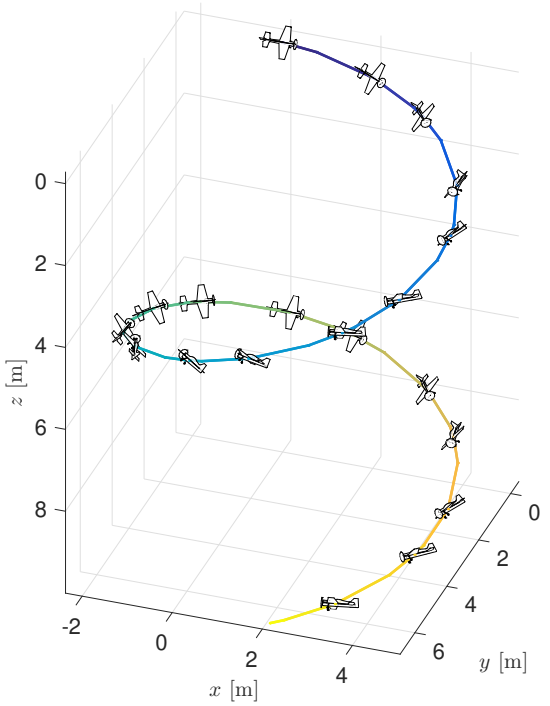


FIGURE 5.5: Helical Turn [12]

While Table. 5.1 contains various yaw rates, climb rates, and speeds, only one speed is used for the duration of a flight (i.e. one column of Table. 5.1).

$\dot{\psi}^{ref}$ $(^\circ/s)$	$v_{i,z}^{ref}$ (m/s)	$\|\|\mathbf{v}_i^{ref}\|\| = 7\ m/s$	$\|\|\mathbf{v}_i^{ref}\|\| = 9\ m/s$	$\|\|\mathbf{v}_i^{ref}\|\| = 11\ m/s$	$\|\|\mathbf{v}_i^{ref}\|\| = 13\ m/s$
0	-2	✓	✓	✓	✓
0	-1	✓	✓	✓	✓
0	0	✓	✓	✓	✓
0	1	✓	✓	✓	✓
0	2	✓	✓	✓	✓
10	-2	✓	✓	✓	✓
10	-1	✓	✓	✓	✓
10	0	✓	✓	✓	✓
10	1	✓	✓	✓	✓
10	2	✓	✓	✓	✓
20	-2	✓	✓	✓	✓
20	-1	✓	✓	✓	✓
20	0	✓	✓	✓	✓
20	1	✓	✓	✓	✓
20	2	✓	✓	✓	✓
30	-2	✓	✓	✓	✓
30	-1	✓	✓	✓	✓
30	0	✓	✓	✓	✓
30	1	✓	✓	✓	✓
30	2	✓	✓	✓	✓
40	-2	✓	✓	✓	
40	-1	✓	✓	✓	
40	0	✓	✓	✓	
40	1	✓	✓	✓	✓
40	2	✓	✓	✓	✓
50	-2	✓	✓		
50	-1	✓	✓	✓	
50	0	✓	✓	✓	
50	1	✓	✓	✓	✓
50	2	✓	✓	✓	✓
60	-2	✓	✓		
60	-1	✓	✓	✓	
60	0	✓	✓	✓	
60	1	✓	✓	✓	
60	2	✓	✓	✓	✓
70	-2	✓	✓		
70	-1	✓	✓		
70	0	✓	✓	✓	
70	1	✓	✓	✓	
70	2	✓	✓	✓	✓
80	-2	✓			
80	-1	✓	✓		
80	0	✓	✓		
80	1	✓	✓	✓	
80	2	✓	✓	✓	
90	-2	✓			
90	-1	✓	✓		
90	0	✓	✓		
90	1	✓	✓	✓	
90	2	✓	✓	✓	
100	-2				
100	-1	✓			
100	0	✓	✓		
100	1	✓	✓		
100	2	✓	✓	✓	
110	-2				
110	-1				
110	0	✓			
110	1	✓	✓		
110	2	✓	✓		

TABLE 5.1: Trajectory feasibility with varying speed

5.2.2 Agile Primitives

Agile motion primitives are obtained by solving a trajectory optimization problem which are only constrained to the vehicle's dynamics model, physical actuator limits, and specified boundary conditions. The objective function of the optimization is to minimize a weighted sum of the duration of the trajectory and the time derivative of the control inputs throughout the trajectory. This results in aggressive trajectories with smooth control inputs. Varying the boundary conditions results in different agile maneuvers. Three agile trajectories are presented in [12]: an aggressive turnaround (see Fig. 5.6), which is used to rapidly change the aircraft's heading, a hover-to-cruise maneuver (see Fig. 5.7), and a cruise-to-hover maneuver (see Fig. 5.8).

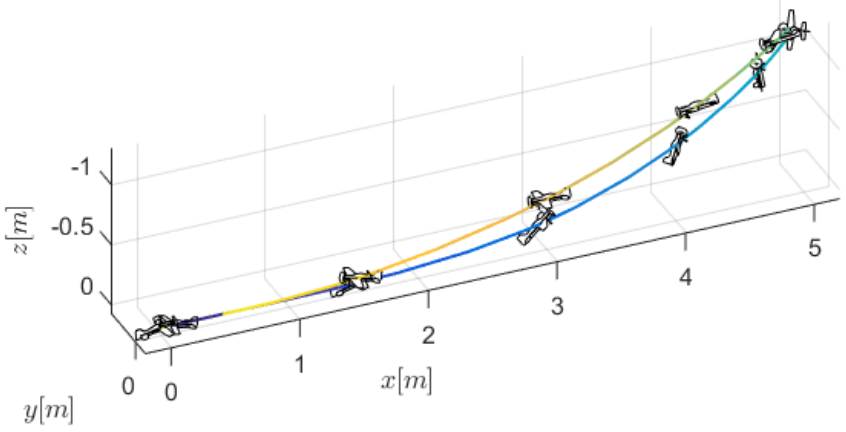


FIGURE 5.6: Aggressive Turnaround [12]

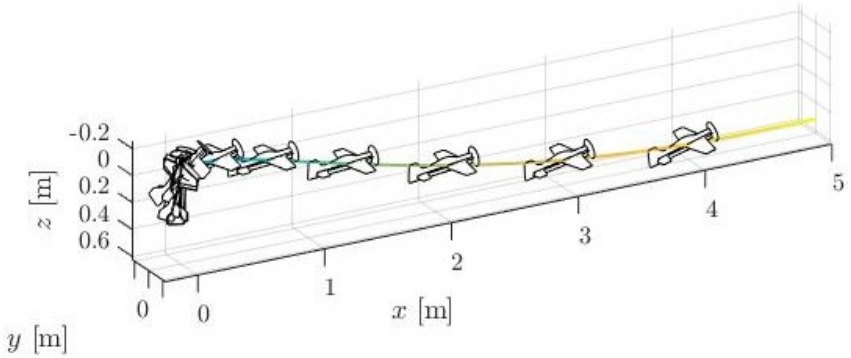


FIGURE 5.7: Hover-to-Cruise [12]

While in Chapter 4 we generated the knife-edge and rolling harrier maneuvers heuristically, we could also generate dynamically feasible knife-edge and rolling harriers via trajectory optimization. However, we elect to not generate these maneuvers because they are not useful within the trajectory selection method presented in Sec. 5.3.

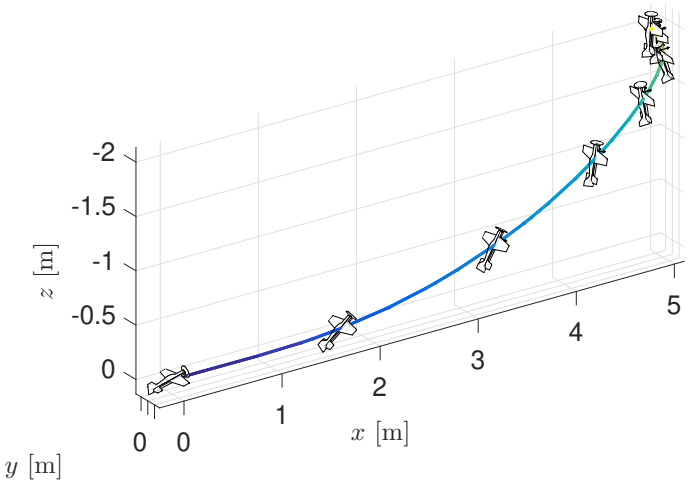


FIGURE 5.8: Cruise-to-Hover [12]

5.3 Trajectory Selection

While the aircraft is cruising, we use the trim primitives in our library to avoid obstacles and steer the aircraft towards the goal. Using the point cloud from a stereo camera we come up with a motion plan within the aircraft's field-of-view (FOV).

First we compute a set of trim primitives from the current aircraft location to various locations on the edge of the FOV. The target locations on the edge of the FOV are represented by the red dots in RVIZ in Fig. 5.9, and the trim trajectories which end at those locations are represented by the purple arrows, where the direction of the arrow is the yaw along the trajectory. As shown in the figure, not all final target positions are reachable using dynamically feasible trim primitives. The final target positions are fixed to the edge of the FOV, and their location is therefore a function of the aircraft attitude (and position). The selection of the final target locations and computation of the trim primitive is discussed in detail in Sec. 5.3.1.

Next, the minimum distance between every point in the point cloud and each potential trajectory is computed and used to determine whether the potential trajectory will result in a collision. This computation is discussed in detail in Sec. 5.3.2. A cost is assigned to each collision-free trajectory, where this cost increases by: being near obstacles, steering the aircraft away from the goal, or changing the current heading rate. The trim trajectory with the lowest cost is selected for the aircraft to track. Referring to Fig. 5.9, the point cloud is represented by the white dots, the goal location is represented by the blue sphere,

and the blue arrows represent the minimum cost trajectory. The trajectory costs are discussed in detail in Sec. 5.3.3.

If there are no collision-free trim trajectories which lead to the edge of the FOV, the aircraft will execute a cruise-to-hover maneuver. The final target positions at the edge of the FOV are chosen such that the aircraft always has enough space to execute a collision-free cruise to hover, which theoretically guarantees collision-free flight. This notion of safety is discussed further in Sec. 5.3.4.

In any scenario, the obstacle avoidance will send the controller a reference trajectory. The reference states are extracted from the reference trajectory (discussed in Sec. 5.3.5) and the controller generates the actuator commands to track the reference trajectory as discussed in Chapter 3. The reference state is represented by the blue aircraft in Fig. 5.9.

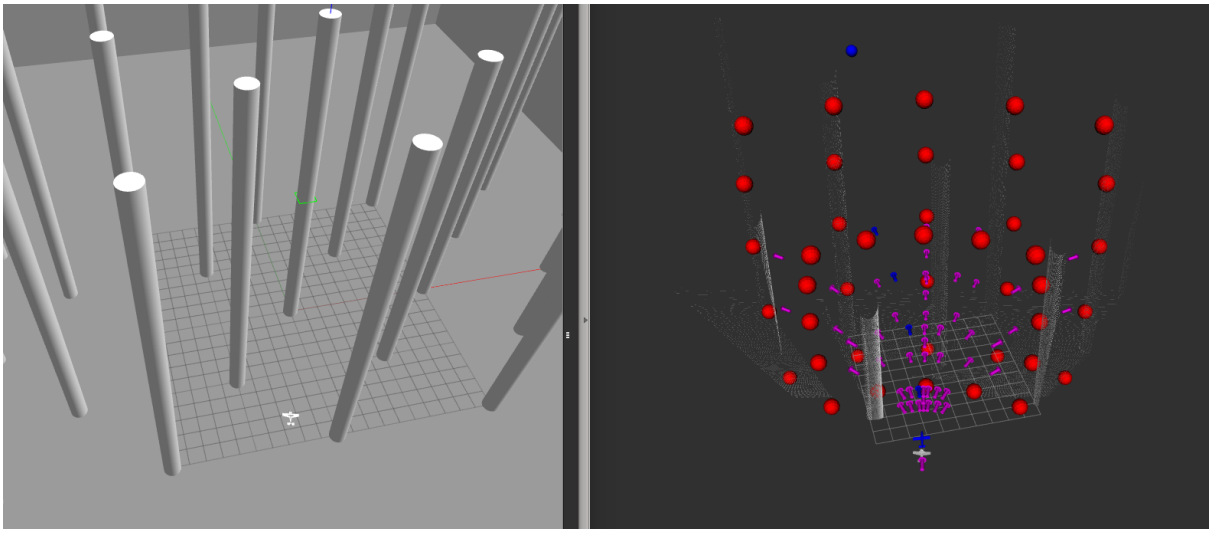


FIGURE 5.9: Example of one motion planning time-step in Gazebo (left) and RVIZ (right)

5.3.1 Obtaining Trajectories to Evaluate

Evaluating too many trajectories increases the computation time and ultimately slows down the speed at which the obstacle avoidance algorithm can run. By contrast, evaluating too few trajectories doesn't utilize all of the maneuvering capability of the aircraft, and could potentially cause the collision-avoidance algorithm to believe there are no collision-free trajectories, when in reality there actually are. We select 41 final target positions along the edge of the FOV, and compute the trim primitive which leads to each target position. These final positions at the edge of the FOV are chosen considering the safety

analysis presented later in Sec. 5.3.4, which ensures any trajectory chosen will always have enough space to perform an emergency hover and prevent a collision.

The 41 final target positions are made up of (a) 25 final positions in an equally spaced 5×5 grid at the end of the range of the depth sensor (20 m), and (b) 16 final positions on the horizontal and vertical edges of the FOV at an intermediate exit distance (10 m). Larger exit distances allow the aircraft more time to react to obstacles, but limit the aggressiveness of the maneuvers. On the contrary, smaller exit distances allow for more aggressive maneuvering which is necessary to fly through certain environments. We compute these final positions using Eqs. (5.1 & 5.2), which are derived from the geometry shown for an arbitrary final target position in Fig. 5.10.

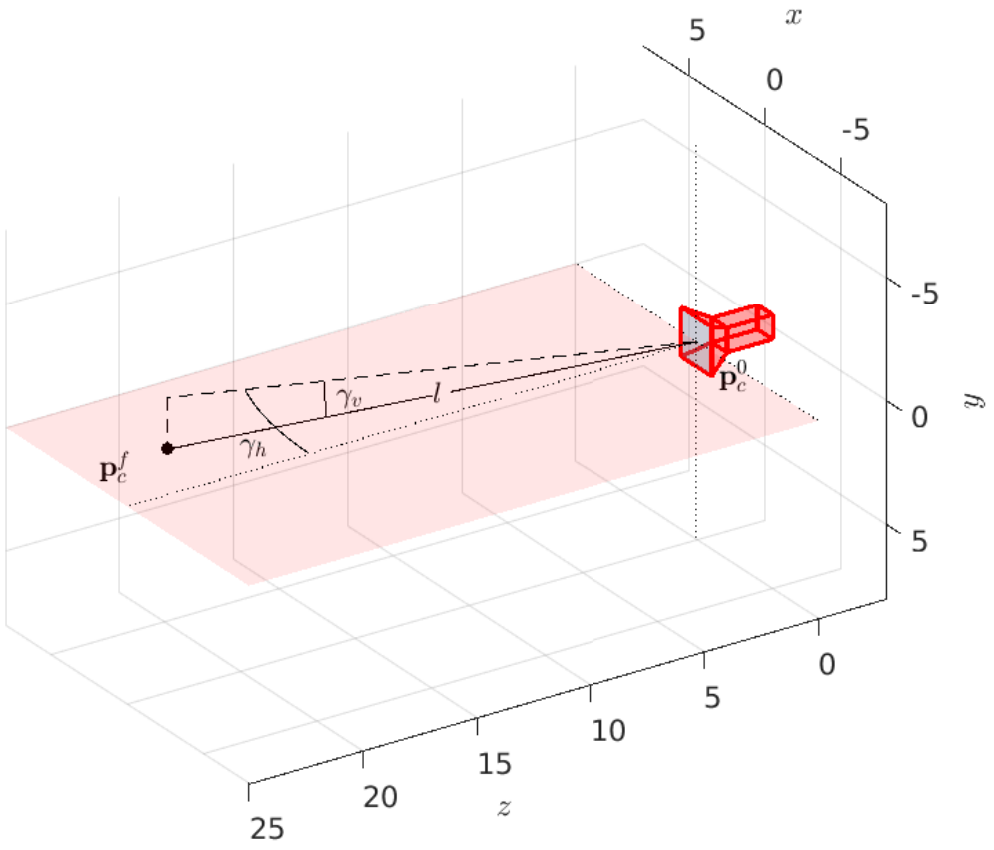


FIGURE 5.10: Computing final positions in the camera coordinate frame

We use the camera coordinate system, $\mathcal{F}_c$, in Fig. 5.10, where the z-axis points out of the camera shown in red. The axes are represented by black dotted lines, and the $x-z$ plane at $y = 0$ is shown in red. The straight line vector from the aircraft's current position to the final position ($\mathbf{p}_c^0$ to $\mathbf{p}_c^f$), which has a magnitude of l, is shown using the solid black line. The projection of this vector onto the $x - z$ plane is shown with the dotted black

line, and the angle between the vector and its projection is represented by γ_v. The angle between the projection and z-axis is denoted by γ_h.

In order to obtain the 25 final positions in an equally spaced 5×5 grid at the end of the range of the depth sensor, l is set to the camera range, while every combination of γ_v and γ_h are chosen from the sets of $\gamma_v = \{-\frac{VFOV}{2}, -\frac{VFOV}{4}, 0, \frac{VFOV}{4}, \frac{VFOV}{2}\}$ and $\gamma_h = \{-\frac{HFOV}{2}, -\frac{HFOV}{4}, 0, \frac{HFOV}{4}, \frac{HFOV}{2}\}$, where the $VFOV$ is the vertical field-of-view, and $HFOV$ is the horizontal field-of-view. Obtaining the 16 final positions at an intermediate exit distance can be obtained by setting l to a smaller value than the range of the depth sensor, but large enough to ensure a safe stopping maneuver is possible (see Sec. 5.3.4). At this intermediate exit distance the final positions must remain on the edge of the FOV, and thus only combinations of γ_v and γ_h are used when at least one angle is at its maximum or minimum value. In practice, we found it advantageous to set the value of the $HFOV$ in the algorithm to slightly less than the manufacturer's spec of the camera, to ensure the trajectories remain within the FOV even with some control or camera errors.

For arbitrary values of l,γ_h, and γ_v, we can compute the position vector from the aircraft's current position to the final position, resolved in the camera frame as

$$\mathbf{p}_c^f - \mathbf{p}_c^0 = \begin{bmatrix} l\cos\gamma_v \sin\gamma_h \\ l\sin\gamma_v \\ l\cos\gamma_v \cos\gamma_h \end{bmatrix}, \tag{5.1}$$

which ultimately is used to compute the final position resolved in the inertial frame as

$$\mathbf{p}_i^f = \mathbf{C}_{bi}^T \mathbf{C}_{cb}^T \begin{bmatrix} l\cos\gamma_v \sin\gamma_h \\ l\sin\gamma_v \\ l\cos\gamma_v \cos\gamma_h \end{bmatrix} + \mathbf{p}_i^0. \tag{5.2}$$

The direction cosine matrix from the body-fixed frame to the camera frame is denoted by $\mathbf{C}_{cb}$, where

$$\mathbf{C}_{cb} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}. \tag{5.3}$$

As we can see in Eq. (5.2), the final target positions are a function of both the aircraft's current position and orientation, since the camera is rigidly mounted to the aircraft.

These final positions are shown in red in Fig. 5.11. Evaluating up to 41 trim trajectories allows the algorithm to run at $5Hz$ using our on-board computer, the Odroid-XU4.

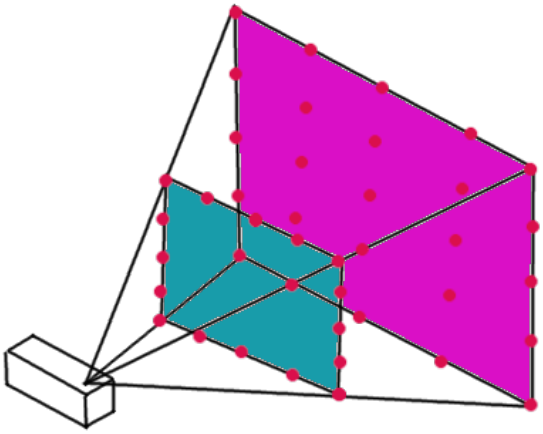


FIGURE 5.11: Final positions of one motion planning time-step

Using the geometry in Figs. 5.12 & 5.13, we can easily compute the trim trajectory that travels from the aircraft's current location, $\mathbf{p}_i^0(x^0, y^0, z^0)$, to the final position at the edge of the depth camera's FOV, $\mathbf{p}_i^f(x^f, y^f, z^f)$, which are the red dots in Fig. 5.11.

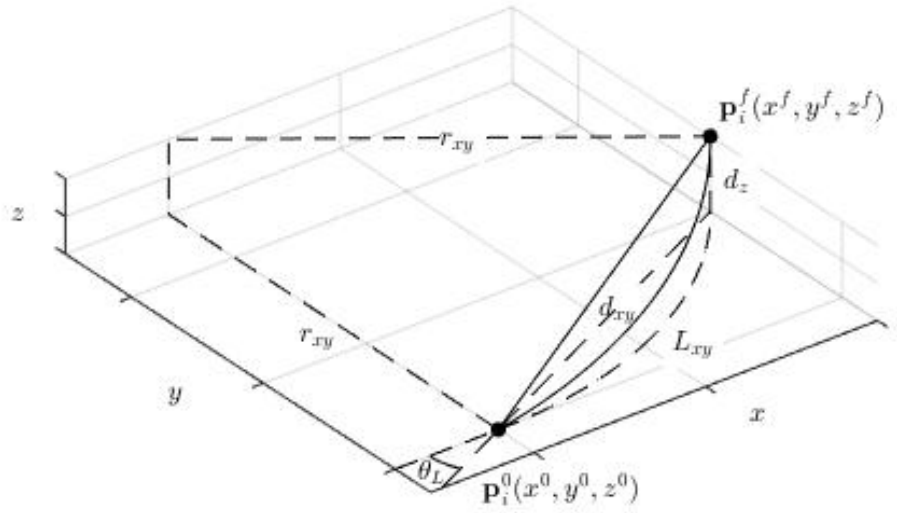


FIGURE 5.12: 3D circular arc defining trim primitive geometry

Since the trim primitives have no sideslip, by assuming the wind is small we can assume that the aircraft's yaw, ψ, is the same as the aircraft's heading. With this assumption we can solve for the trim primitive defined by its yaw rate, $\dot{\psi}^{ref}$, climb/descent rate, $v_{i,z}^{ref}$, and coasting time, Δt.

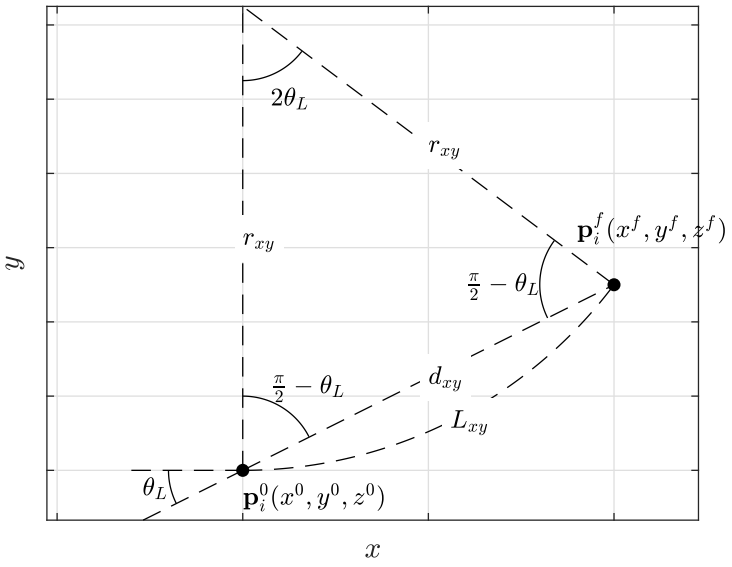


FIGURE 5.13: Top-View of 3D circular arc defining trim primitive geometry

The magnitude of the vector going from the aircraft to the final position, projected on the horizontal plane, is

$$d_{xy} = \sqrt{(x^f - x^0)^2 + (y^f - y^0)^2}, \tag{5.4}$$

and this vector projected on to the vertical plane is

$$d_z = z^f - z^0. \tag{5.5}$$

The angle between aircraft's current yaw, ψ^0, and vector from $\mathbf{p}_i^0$ to $\mathbf{p}_i^f$ projected onto the horizontal plane, is

$$\theta_L = \text{wrap2pi}(\arctan(\frac{y^f - y^0}{x^f - x^0}) - \psi^0). \tag{5.6}$$

Referring to Fig. 5.13, we can obtain the turn radius in the horizontal plane, r_{xy}, from

$$r_{xy} \sin(\frac{1}{2} 2\theta_L) = \frac{1}{2} d_{xy} \tag{5.7}$$

which reduces to

$$r_{xy} = \frac{d_{xy}}{2 \sin \theta_L}. \tag{5.8}$$

We can then compute the arclength projected onto the horizontal plane from

$$L_{xy} = \int_0^{2\theta_L} r_{xy} \, d\Theta = r_{xy}(2\theta_L) = \frac{\theta_L d_{xy}}{\sin \theta_L}. \tag{5.9}$$

The coasting time is computed as

$$\Delta t = \frac{L_{xy}}{v_{i,xy}^{ref}} = \frac{d_z}{v_{i,z}^{ref}}, \tag{5.10}$$

where aircraft reference speed in the horizontal plane is denoted by $v_{i,xy}^{ref}$ and the reference climb/descent rate as $v_{i,z}^{ref}$. By definition,

$$||\mathbf{v}_i^{ref}||^2 = {v_{i,xy}^{ref}}^2 + {v_{i,z}^{ref}}^2, \tag{5.11}$$

and since $||\mathbf{v}_i^{ref}||$ is a constant known quantity, we can obtain the speed in the horizontal plane as

$$v_{i,xy}^{ref} = \frac{||\mathbf{v}_i^{ref}||}{\sqrt{1 + \frac{d_z^2}{L_{xy}^2}}}. \tag{5.12}$$

Now that $v_{i,xy}^{ref}$ is solved for, we can obtain the coasting time, Δt, from Eq. 5.10. We can then compute the climb/descent rate from

$$v_{i,z}^{ref} = \frac{d_z}{\Delta t}. \tag{5.13}$$

The yaw rate can then be computed as

$$\dot{\psi}^{ref} = \frac{v_{i,xy}^{ref}}{r_{xy}}. \tag{5.14}$$

The computed yaw rate and climb/descent rate are then rounded to the nearest 10 $^\circ/s$ and 1 m/s to find the closest trim primitive in the library. For each trim primitive the associated roll, pitch, angular and linear body-frame velocity are all constant, and stored in memory. The reference yaw and position can be easily computed at a time t since the start of the maneuver. The aircraft's reference yaw at time t, $\psi^{ref}(t)$, is calculated by:

$$\psi^{ref}(t) = \psi^0 + \dot{\psi}^{ref} t. \tag{5.15}$$

The reference position at time t, $\mathbf{p}_i^{ref}(t) = (x^{ref}(t), y^{ref}(t), z^{ref}(t))$, is calculated by:

$$
\begin{aligned}
x^{ref}(t) &= \begin{cases} x^0 + v_{i,xy}^{ref}\ t\cos\psi^0, & \text{if } \dot{\psi}^{ref} = 0 \\ x^0 + \frac{v_{i,xy}^{ref}}{\dot{\psi}^{ref}}(\sin(\psi^0 + \dot{\psi}^{ref}t) - \sin\psi^0), & \text{otherwise} \end{cases} \\
y^{ref}(t) &= \begin{cases} y^0 + v_{i,xy}^{ref}\ t\sin\psi^0, & \text{if } \dot{\psi}^{ref} = 0 \\ y^0 + \frac{-v_{i,xy}^{ref}}{\dot{\psi}^{ref}}(\cos(\psi^0 + \dot{\psi}^{ref}t) - \cos\psi^0), & \text{otherwise} \end{cases} \\
z^{ref}(t) &= z^0 + v_{i,z}^{ref}\ t
\end{aligned}
\tag{5.16}
$$

The ability to quickly compute the trim primitive is a major advantage, since it allows us to only use computational resources on evaluating trajectories that (a) remain within the FOV, and (b) represent the entirety of the FOV. While we only evaluate up to 41 trajectories in at given time-step, they are (up to) 41 trajectories selected from a much larger trajectory library, and increasing the size of the trajectory library does not increase the computational burden on the algorithm. If we weren't using trim primitives, and therefore couldn't compute the trajectory which leads to a final position, we would have to evaluate trajectories regardless of where they end up. This would lead to evaluating trajectories that potentially: don't span the entirety of the FOV, don't end at the edge of the FOV, or end outside the FOV.

Another benefit of using trim primitives is their constant roll, pitch, and turn rate is advantageous when switching maneuvers frequently. In our preliminary work in [6], we only used finite-time primitives. In order to keep the size of the library tractable, each maneuver started from level flight (zero roll), similar to [79]. In order to effectively turn, these reference trajectories must be executed for a significant amount of time to allow the reference trajectory to go from level to banking, and then turn while banking. If the aircraft was initially banked, it would first roll towards level flight, and then back towards the desired bank angle, ultimately slowing down the turn. By using constant trim primitives, the need to start from level flight is avoided, and the aircraft is always directly commanded to the specified bank angle. This avoids the problem of initially turning the wrong way, and removes the need to execute the trajectory for a significant amount of time, which puts no limitation on the frequency at which reference trajectories are updated. This higher frequency enables the aircraft to fly through more cluttered environments at higher speeds.

5.3.2 Distance to Obstacles

The minimum distance from each primitive being evaluated to the point cloud representation of obstacles is used to determine if the primitive is collision-free, and to determine the cost associated with each primitive. The point cloud is made up of position vectors from the camera to the points in the point cloud, and are resolved in a coordinate frame fixed to the depth camera, $\mathcal{F}_c$. On the other hand, the positions along the trajectory are with respect to the NED origin, and are resolved in the inertial frame. In order to compute the distance from a point on a trajectory to the point cloud, we need to translate and rotate the positions along the trajectory. We can perform this transformation at a time t along the trajectory as follows:

$$\mathbf{p}_c^{ref/0}(t) = \mathbf{C}_{cb}\mathbf{C}_{bi}(\mathbf{p}_i^{ref}(t) - \mathbf{p}_i^0) \tag{5.17}$$

where $\mathbf{p}_c^{ref/0}(t)$ is the reference position with respect to aircraft, resolved in the camera frame, t seconds along of the trajectory. The position t seconds along the trajectory, with respect to the origin and resolved in the inertial frame, $\mathbf{p}_i^{ref}(t)$, can be computed using Eq. (5.16). We assume that the camera is positioned at the aircraft's center of mass, and thus $\mathbf{p}_c^{ref/camera}(t) = \mathbf{p}_c^{ref/0}(t)$.

Every 25 cm along the trajectory we compute the distance from that position to the closest point in the point cloud, which is stored in an octree for efficient distance computation. The minimum distance obtained is the distance to obstacle, d^{obs}. If this distance is less than half the aircraft's wingspan, the trajectory will result in a collision and is discarded. While the true wingspan of our vehicle is 0.86 m, we set the wingspan during collision checking to 2 m, to give a buffer for potential camera misalignment, state estimation errors, control errors, or modeling errors. Increasing the buffer size reduces the risk of collision in case of these aforementioned errors, but comes at the cost of potentially executing an emergency hover when the aircraft could have actually maneuvered around an obstacle.

5.3.3 Trajectory Cost

A cost function is constructed for each trajectory that specifies how competing objectives are combined. The cost of a potential trajectory (c) grows when being near obstacles (c^{obs}), the aircraft is steered away from the goal in both the horizontal (c^h) and vertical (c^v) plane, and when the trajectory selected significantly differs than the trajectory it's

currently on (c^{diff}). The total cost associated with the potential trajectory is the sum of all these sub-costs:

$$c = c^{obs} + c^{h} + c^{v} + c^{diff}. \tag{5.18}$$

The cost for being near obstacles, c^{obs} is computed as follows:

$$c^{obs} = \begin{cases} -d^{obs} w^{obs}, & \text{if } d^{obs} < 10 \\ -10 w^{obs}, & \text{otherwise} \end{cases} \tag{5.19}$$

where w^{obs} is a positive weight and d^{obs} is the minimum distance from the potential trajectory to the point cloud. The rationale behind this cost function is the cost gets reduced linearly the further the trajectory is from an obstacle, until a critical distance of 10 m at which the trajectory is so far from an obstacle that it becomes irrelevant how far away it is. In our preliminary work in [6], we simply discarded trajectories that result in a collision, but didn't add the distance to the obstacles within the cost function. Adding this distance into the cost function significantly improves performance because trajectories that are predicted to be collision-free but close to obstacles could actually result in a collision due to modeling inaccuracies and control errors. With trajectories being penalized for being near obstacles, the trajectories that end up getting selected are further away from obstacles.

The cost due to being steered away from the goal in the horizontal plane, c^{h}, is computed as follows:

$$c^{h} = w^{h} || \text{wrap2pi}(\tan^{-1}(\frac{y^{g} - y^{0}}{x^{g} - x^{0}}) - \psi^{f}) || \tag{5.20}$$

where the goal position is denoted $\mathbf{p}_i^g(x^g, y^g, z^g)$ and the wrap2pi function keeps the angle in parenthesis between $-\pi$ and π. The cost is proportional to the angle between the straight line from the aircraft to the goal, and the final yaw angle of the trajectory, which is depicted in Fig. 5.14.

The cost due to being steered away from the goal in the vertical plane, c^{v}, is proportional to the distance from the altitude of the aircraft at the end of the trajectory and the altitude of the goal, and is computed as follows:

$$c^{v} = w^{v} || z^{g} - z^{f} ||. \tag{5.21}$$

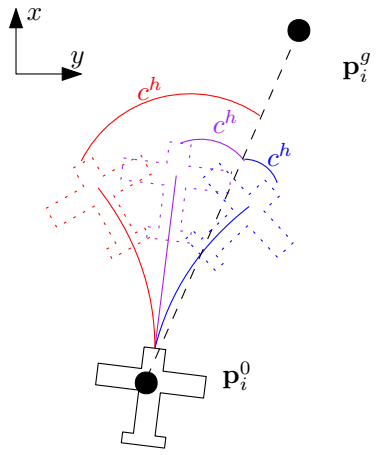


FIGURE 5.14: Depiction of horizontal cost

The last portion of the cost comes from switching trajectories with large differences in yaw rate, because frequent switching reduces the efficiency of flight, the stability of the aircraft, and the ability for the aircraft to track the desired trajectory. In order to reduce the number of switches, as well the extremity of each switch, we add a penalty when switching to a trajectory with a different yaw rate. The cost is linearly proportional to the difference between the yaw rate of the trajectory being evaluated and the yaw rate of the primitive the aircraft is currently tracking:

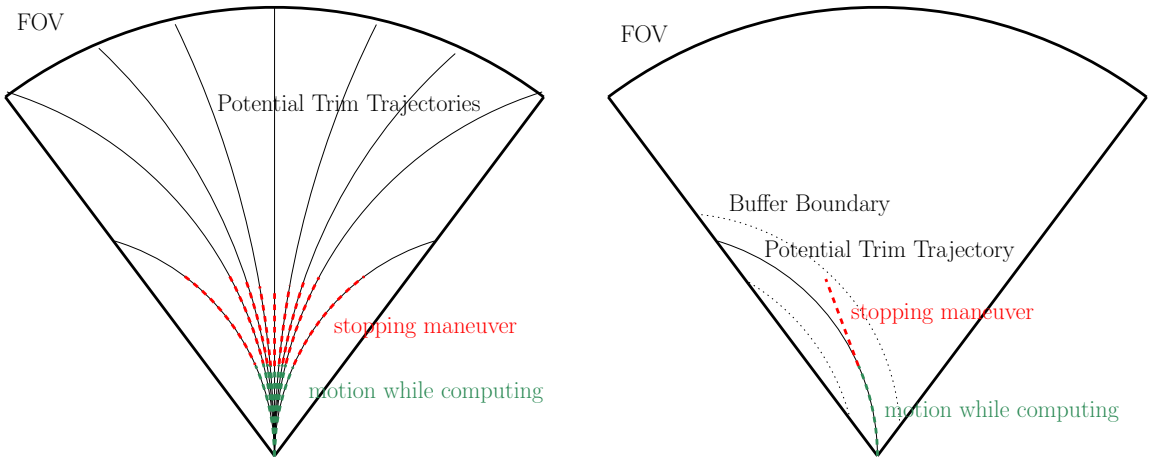
$$c^{diff} = w^{diff}||(\dot{\psi}^{ref} - \dot{\psi}^{ref,current})||. \tag{5.22}$$

Ultimately, the trajectory with lowest cost, c, is sent to the controller to track.

5.3.4 Safety Gaurantees

The obstacle avoidance algorithm must ensure the aircraft never enters an inevitable collision state. In [70, 71], safety is guaranteed with a quadrotor by restricting motion primitives to remain within the depth sensor's FOV in addition to ensuring a stop maneuver is possible if no collision-free path is found. In [75], safety is guaranteed by ensuring the flapping-wing MAV can always fly in a circle within the FOV.

A major advantage to developing collision avoidance strategies with agile fixed-wing aircraft, as opposed to traditional aircraft, is the ability to come to a complete stop while hovering. This allows us to guarantee safety using methods developed for rotorcraft. Since only collision-free motion primitives are selected, if we can ensure that at the next motion planning time-step, the aircraft can execute a stopping maneuver that remains within the



(a) Multiple stopping maneuvers along trim trajectories

(b) Single stopping maneuver: the Cruise-to-Hover

FIGURE 5.15: Top-down view of motion primitives within FOV

current time-step's FOV, and travels along the current time-step's collision-free path, one can guarantee the aircraft can always stop to avoid a collision. This is demonstrated in Fig. 5.15a, where the green dash represents the aircraft motion while the motion plan is being computed, and the red dash represents the motion if the aircraft was to execute a stopping maneuver on the following time-step, and that stopping maneuver followed the path of the previous time-step's trim primitive.

While it is potentially possible to generate a stopping maneuver associated with each trim primitive, it is much simpler to only have one stopping maneuver, such as the Cruise-to-Hover maneuver described in Sec. 5.2.2. During the collision checking step a buffer distance is added, so a trajectory will only be selected if the space occupied by the buffer is also collision-free. If the single cruise-to-hover maneuver remains close enough to any potential trim primitive, such that it remains in the buffer, the same logic as before still applies. This is the case for even fairly small buffer distances, since the duration of the cruise-to-hover is short, and the aircraft won't move too much laterally in a very short period of time. This is demonstrated in Fig. 5.15b, where the cruise-to-hover maneuver does not perfectly follow the previous trim trajectory, but remains within the buffer distance, and within the FOV, guaranteeing collision-free flight. While we only show this for one particular primitive for simplicity, this must hold for all potential trim primitives.

The safety guarantees are only valid under the following assumptions: perfect sensing, perfect control, the initial conditions are such that a cruise-to-hover is collision-free, the obstacles are static.

5.3.5 Controller Integration

The obstacle avoidance algorithm sends the controller a reference motion primitive, as well as the pose of the aircraft at the time the reference primitive was selected at a rate of 5 Hz. The controller runs at 200 Hz and extracts the reference states from the reference trajectory sent from the obstacle avoidance. Once a new reference trajectory is received, the controller instantaneously switches to tracking the new reference trajectory.

For the trim primitives, the reference roll and pitch angles, as well as reference translational and angular velocities, can be directly looked up in memory. The reference yaw and position can be computed using Eqs. (5.15 - 5.16), where $\mathbf{p}_i^0$ and ψ^0, come from the pose sent along with the trajectory, and t is the difference between the current time, and time stamp associated with $\mathbf{p}_i^0$ and ψ^0. Since the position and yaw reference is continuously being reset to its current value, there is no position or yaw error at the beginning of a new trajectory. We found it advantageous to add a tenth of a second to t, which pushes the reference state further downstream the reference trajectory, resulting in position and yaw errors that cause a control action at the beginning of the maneuver. For agile primitives, all of the reference states are stored in memory, but the position reference is rotated by ψ^0 and translated by $\mathbf{p}_i^0$, and yaw reference is offset by ψ^0.

Chapter 6

Obstacle Avoidance Validation

We validate the obstacle avoidance methodology presented in Chapter 5 using both simulated and experimental flight tests. We describe our experimental test platform in Sec. 6.1, which is simulated in Sec. 6.2, and is used in the flight tests presented in Sec. 6.3.

6.1 Platform Description

The same airframe used in the controller validation experiments is used to validate the obstacle avoidance methodology. In addition to the existing hardware, we add a depth sensor, the Intel RealSense D435 camera, and a companion computer, the ODROID-XU4. To compensate for the added mass due to the additional hardware, we upgraded to an 1150 KV motor, the T-Motor AT2312 Long Shaft 1150KV Brushless Motor, and a 30 amp ESC, the Turnigy Plush-32 30A Speed Controller. The propeller used in the controller validation experiments is no longer in production, and so a similar propeller, the Master Airscrew MR Series - 10 x 4.5, is used instead. We also upgraded the Pixhawk flight controller to a Pixracer. With these modifications the mass of the aircraft is 660 g, and shown in Fig. 6.1 with labeled hardware components.

6.1.1 Intel RealSense D435

While searching for an off-the-shelf obstacle detection sensor for our test platform, we were mainly interested in the sensor's weight, power consumption, and range. Given the limited payload on our test platform, LiDAR or RADAR are not feasible options due to their weight and power consumption. On the other hand, stereo cameras are lightweight

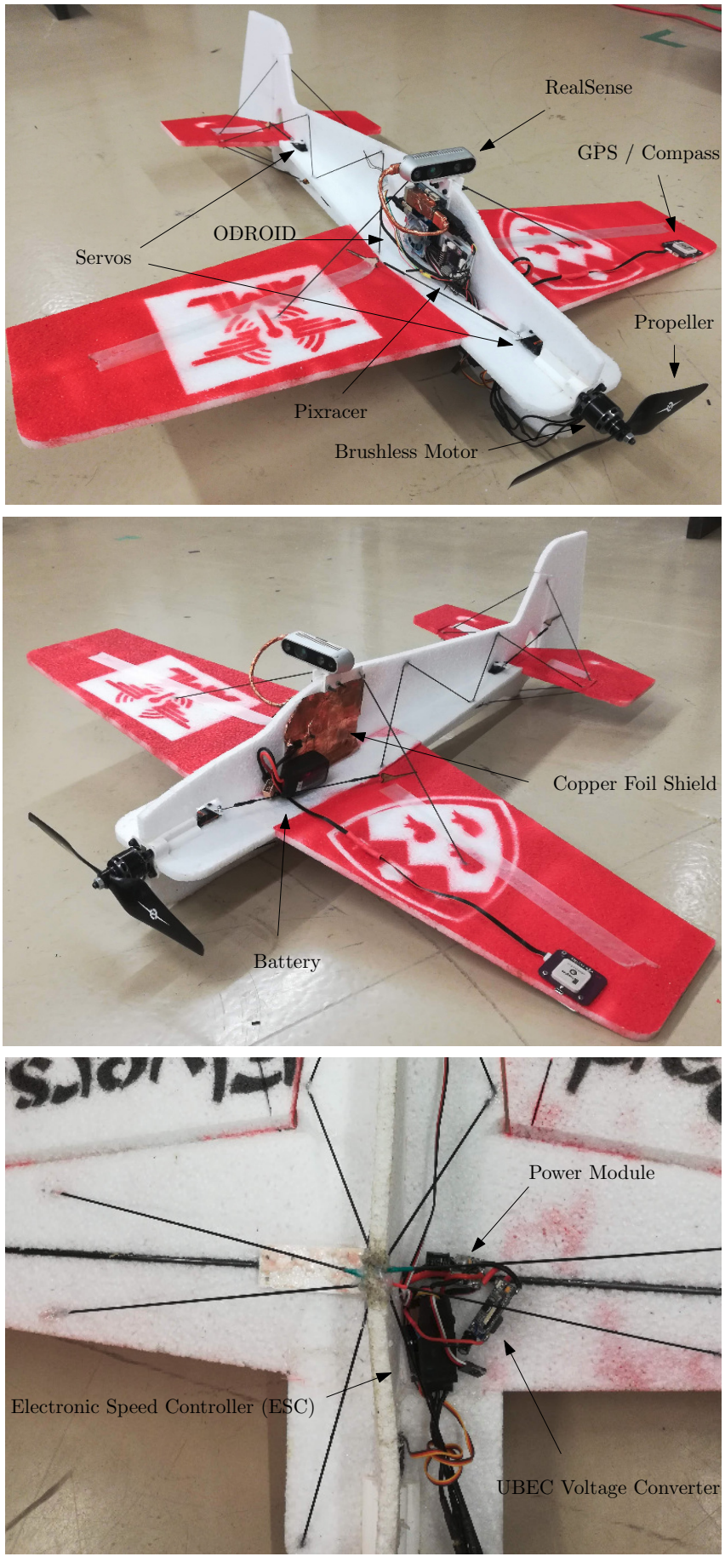


FIGURE 6.1: Mcfoamy Hardware

and consume little power. While a few commercially available stereo cameras exist on the market, we found the Intel Realsense D435 to be the best for our application, due to its weight, power consumption, range, field-of-view, and ease of use. The RealSense contains two global shutter imagers, and RGB camera, and an IR projector. The system is 72 grams, has a field-of-view of $86° \times 57°$ (horizontal $\times$ vertical), and a maximum range of over 10 m that varies with lighting conditions, according to [105]. We found the system provides reliable depth measurements outdoors at much larger ranges, and thus we rely on the system having a 20 m range. We use the camera at 30 frames per second, although it can work as fast at 90 fps. The depth camera is placed such that part of the image is obstructed by the propeller. When the propeller is spinning, the depth image is unaffected with a disabled IR projector, but affected when the IR projector is enabled. Thus we disable the IR projector for this reason, as well as the fact that it is not needed for outdoor operation and consumes power. For the rest of the camera settings we refer to [106], which provides RealSense settings for drone collision avoidance applications.

6.1.2 ODROID-XU4

A companion computer is needed for the obstacle avoidance computation. We use the ODROID-XU4, which has a 2GHz quad-core processor, 2 GB RAM, and a mass of 58 grams. We use the Ubuntu 16.04 operating system along with Ros-Kinetic. The ODROID is powered by a 5V/3A UBEC voltage converter from the Pixracer's power module, and communicates to the Pixracer from a USB port on the ODROID to the telemetry (serial) port of the Pixracer through an FTDI cable. The ODROID also provides power and communication with the RealSense using a USB3 cable.

6.1.3 System Communication

The communication between various hardware and software components is shown in Fig. 6.2. As shown in the diagram, all control and obstacle avoidance computation is done on the ODROID, while the Pixracer, which runs in 'off-board' mode, is only used for state estimation and to send PWM signals. Note this is different from the controller validation experiments, where all computation was done on the Pixhawk. The MAVLink protocol is used to communicate between the Pixracer and ODROID, and all internal communication on the ODROID, as well as communication between the ODROID and the RealSense, is done using ROS. We also note that MAVROS, a package used to convert between MAVLink and ROS, changes the inertial coordinate frame to East-North-Up

(ENU) and the body frame to Front-Left-Up (FLU), so we create a bridge node to convert back to North-East-Down (NED) and Front-Right-Down (FRD), which is sent to the obstacle avoidance and controller nodes.

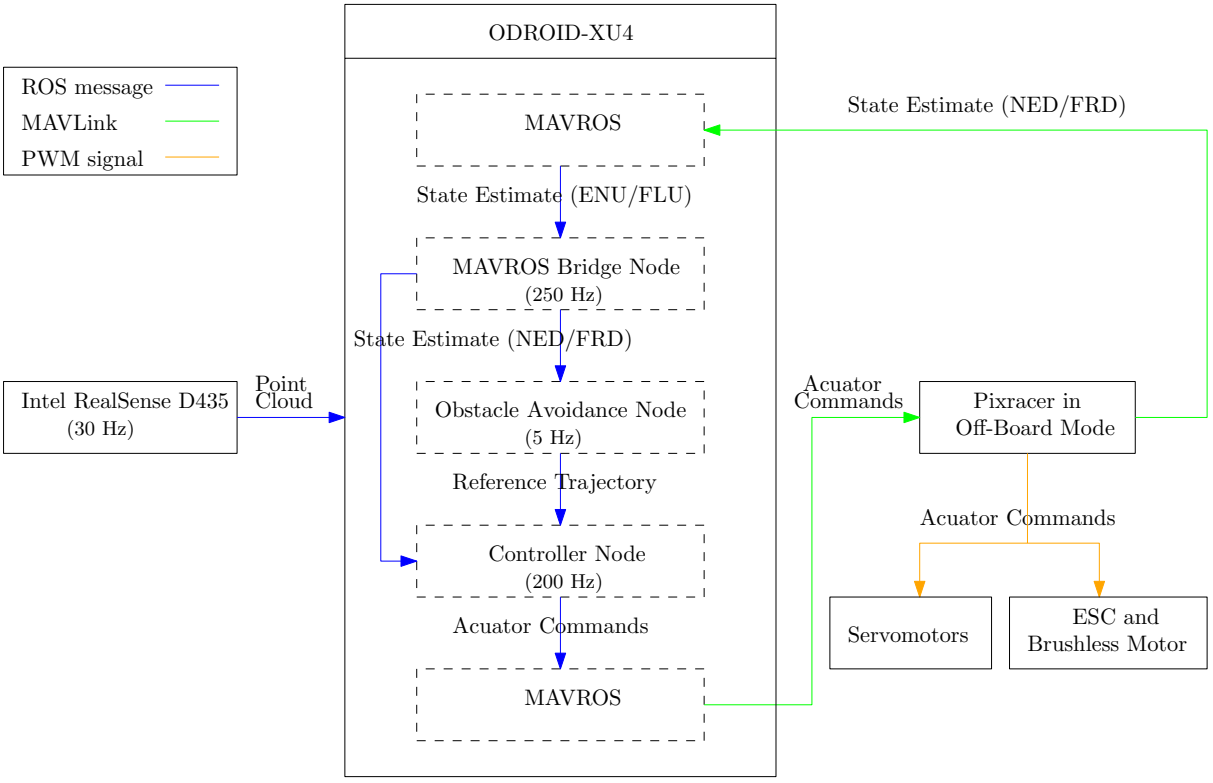


FIGURE 6.2: Hardware and Software Communication Diagram

6.1.4 USB3 Interference

The RealSense D435 requires a USB3 connection, which interferes with the GPS signal. Without adequate shielding this inference can make the GPS so noisy that it is not usable. We were able to significantly reduce the interference by moving the GPS unit to the end of the wing (further from the RealSense and ODROID), lining the side of the fuselage (between the ODROID and GPS unit) with copper foil, and wrapping the USB3 cable in copper foil. These modifications are apparent in Fig. 6.1.

6.1.5 Parameters

Both the simulated and actual aircraft are equipped with the algorithms developed in this thesis. In Table 6.1, we present the parameters used in both the obstacle avoidance and control algorithm during both the simulated and experimental flights. For the true

	Symbol	Simulation Value	Experiment Value	Unit
Obstacle Avoidance Parameters				
Camera Range in Algorithm		20	20	m
True Camera Range		20	10+	m
Field-of-View in Algorithm		65×58	65×58	$^\circ \times ^\circ$
True Field-of-View		86×58	86×58	$^\circ \times ^\circ$
Aircraft Wingspan in Algorithm		2	4	m
True Aircraft Wingspan		0.86	0.86	m
Intermediate Exit Distance		12	10	m
Weight Used to Compute c^{diff}	w^{diff}	0.03	0.03	$\frac{s}{^\circ}$
Weight Used to Compute c^{h}	w^{h}	15	15	$\frac{1}{rad}$
Weight Used to Compute c^{obs}	w^{obs}	2	2	$\frac{1}{m}$
Weight Used to Compute c^{v}	w^{v}	2	2	$\frac{1}{m}$
Controller Parameters				
Position Proportional Control Gain	K_{p_p}	.06	.06	rad/m
Position Derivative Control Gain	K_{p_d}	.15	.15	$rad/\frac{m}{s}$
Attitude Proportional Control Gain	$\mathbf{K_{a_p}}$	120 $\mathbf{1}_{3x3}$	120 $\mathbf{1}_{3x3}$	$\frac{rad}{s^2}/rad$
Attitude Derivative Control Gain	$\mathbf{K_{a_d}}$	8 $\mathbf{1}_{3x3}$	8 $\mathbf{1}_{3x3}$	$\frac{rad}{s^2}/\frac{rad}{s}$
Speed Proportional Control Gain (During Trim Primitives)	K_v	6	6	$\frac{m}{s^2}/\frac{m}{s}$
Height Proportional Control Gain (During Trim Primitives)	K_{h_p}	0	0	$\frac{m}{s^2}/m$
Height Integral Control Gain (During Trim Primitives)	K_{h_i}	0	0	$\frac{m}{s^2}/ms$
Speed Proportional Control Gain (During Agile Primitives)	K_v	3	3	$\frac{m}{s^2}/\frac{m}{s}$
Height Proportional Control Gain (During Agile Primitives)	K_{h_p}	5	5	$\frac{m}{s^2}/m$
Height Integral Control Gain (During Agile Primitives)	K_{h_i}	0.5	0.5	$\frac{m}{s^2}/ms$
Aerodynamic Force Scaling Parameter	K_{aero}	2	2.5	

TABLE 6.1: Parameters used during obstacle avoidance validation

camera parameters, we used the manufacturer's specifications for the experiment column, and the values specified in the Gazebo plugin for the simulation column.

6.2 Simulation

We first use a simulation to validate the obstacle avoidance algorithm. A simulation environment is a useful tool to test various algorithms and parameter values, prior to testing

on an actual aircraft. Furthermore, certain environmental conditions, such as wind, sunlight, or arrangement of trees are uncontrolled in an experimental implementation, but can be easily controlled in simulation. In Chapter 4, we utilize a conventional MATLAB simulation, and an HIL simulation, prior to validating the controller in experiment. These simulations are only capable of simulating the aircraft dynamics and controller. To realistically simulate the obstacle avoidance algorithm, the simulation environment must also include the ability to simulate obstacles and a depth camera. We found it easiest to simulate all of these components using Gazebo [85], where we can accurately simulate the aircraft described in Sec. 6.1. The aircraft dynamics are simulated using a custom plug-in that corresponds to the aerodynamics, thruster, and slipstream modeling presented in Sec. 2.1. The RealSense D435 camera described in Sec. 6.1.1 is simulated in Gazebo by modifying the parameters in the existing Kinect depth camera plug-in.

We evaluate the obstacle avoidance methodology in four environments, each with different types of obstacles and of varying difficulty. Each test environment contains very tall obstacles such that aircraft must go around them, and not over them. While the methodology presented is in three dimensions and the aircraft can climb and descend to avoid obstacles, we found it clearer to display results using the horizontal plane, and therefore use very tall obstacles in each simulation environment, which forces the aircraft to go around, and not over them. In each test environment, we run 10 simulations that use a 9 $\frac{m}{s}$ trajectory library, and 10 simulations that use a 13 $\frac{m}{s}$ trajectory library. For each test environment, we select 10 positions to initialize the aircraft from zero velocity, zero angular velocity, 0° roll, 0° pitch, and 90° yaw. We allow one second to pass to allow the aircraft to reach cruise conditions, and show the simulation results from then. For each test environment, we use the same 10 initial positions for both trajectory library speeds. Each simulation is executed until either the aircraft reaches the goal and hovers, the aircraft executes an emergency hover, or the aircraft collides with an obstacle. For every simulation run, the goal is located at $(x, y, z) = (30, 0, -10)$ m in the NED frame, and if the aircraft enters within 5 m of this position it is commanded to hover.

We must emphasize the primary objective of the obstacle avoidance algorithm is to avoid obstacles, and reaching the goal is secondary, due to the lack of a SLAM implementation and global motion planner, as mentioned in Chapter 5. In these simulation runs, we end the simulation once the aircraft performs an emergency hover, since the aircraft cannot perceive the environment with its camera pointing upward, and has no memory of the environment it was just flying through. If a SLAM algorithm was implemented, the aircraft would build a map of the environment while flying, and if an emergency hover

is needed, the aircraft could compute a motion plan using that map while hovering, and continue flying to the goal.

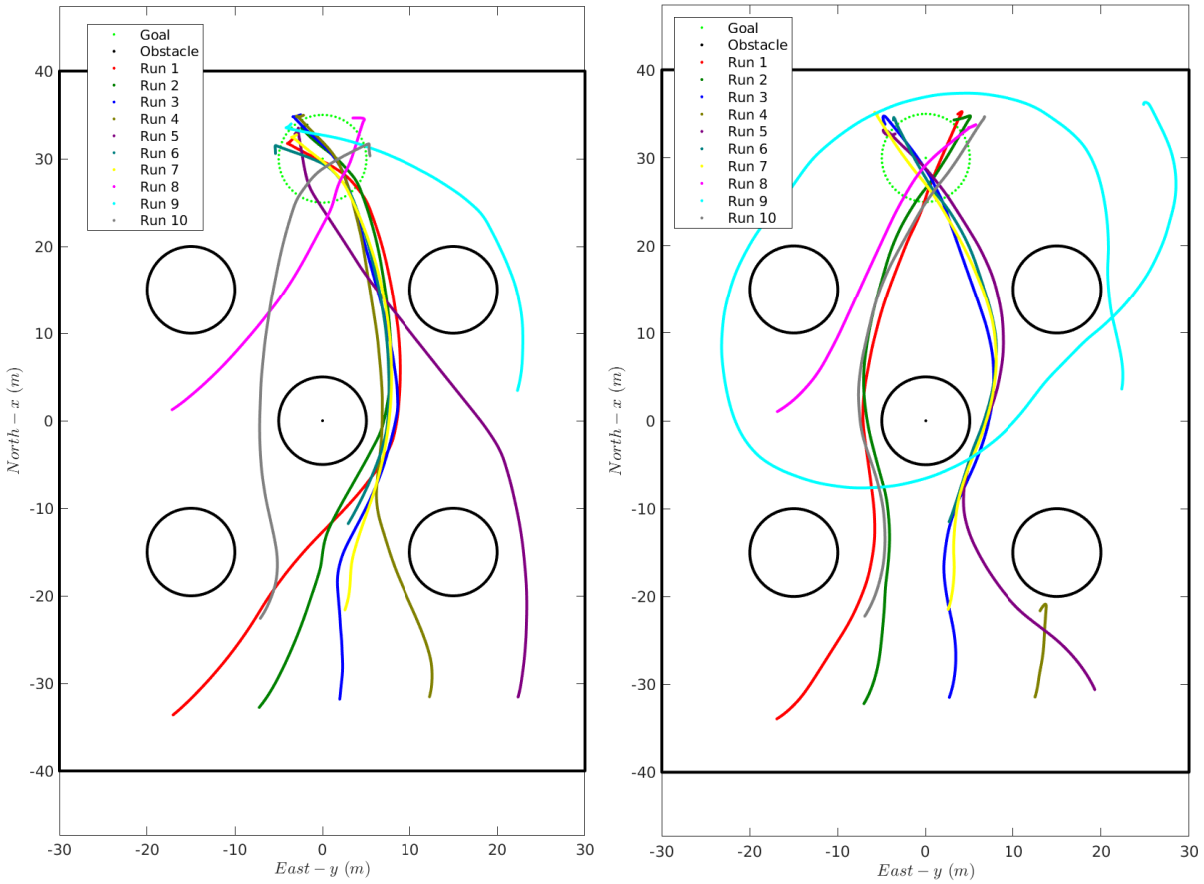
6.2.1 Environment 1

We first test the obstacle avoidance methodology in a relatively easy environment with only a few obstacles and large gaps between them. The first environment consists of 5 tall poles with a 5 m radius which are enclosed in a 60 $m \times 80$ m rectangle. A top-down view of the environment and flight trajectories is shown for the trajectories using the 9 $\frac{m}{s}$ trajectory library in Fig. 6.3a and for the trajectories using the 13 $\frac{m}{s}$ trajectory library in Fig. 6.3b. As shown in Fig. 6.3a, the aircraft reaches the goal for all 10 runs using the 9 $\frac{m}{s}$ trajectory library.

Fig. 6.3b shows a top-down view of the environment using the 13 $\frac{m}{s}$ library. The aircraft reaches the goal 8 out 10 times. In the two runs where the aircraft does not reach the goal, collisions are avoided with the emergency hover. In run 4, the aircraft simply does not have enough space to maneuver around the pole at 13 $\frac{m}{s}$. In run 9, the aircraft is not able to turn sharply enough to enter the goal region, and ends up looping back through poles, and ultimately corners itself and performs an emergency hover. The aircraft reaches the goal more often at lower speed because it can turn with a smaller turning radius. Not only is the turning radius larger when flying faster with the same heading rate, the maximum heading rate is decreased at higher speed, due to throttle saturation, as mentioned in Sec. 5.2.

We examine run 3 of the 9 $\frac{m}{s}$ trajectory library in more detail using Figs. 6.4 & 6.5. At selected times throughout the flight, Fig. 6.4, shows from left to right: a 3rd-person aerial image; the location and yaw (red dot and red line) with respect to the top-down view of the trajectory and environment; the on-board color image; and the on-board depth image. In the depth image, black corresponds to nothing being detected, and obstacles are represented in gray-scale where the darker pixels correspond to closer obstacles. In Fig. 6.5 we plot the aircraft states, references, and control inputs (throttle (u_1^s), aileron (u_2^s), elevator (u_3^s), rudder (u_4^s)) for the duration of the flight. For clarity, we only show the roll and yaw angles from $-50°$ to $50°$ and $-60°$ to $60°$ respectively, as the motion in cruising flight remains within these limits. They exceed these limits when hovering, but roll and yaw become unintuitive variables when the pitch is near $90°$.

As we can see in Fig. 6.4, at the start of the trajectory the aircraft sees the pole at $(x, y) = (-15, 15)$ and turns slightly left. The aircraft then continues straight, and at



(a) 9 $\frac{m}{s}$

(b) 13 $\frac{m}{s}$

FIGURE 6.3: Environment 1

$t = 1.2\ s$ the aircraft sees the pole at $(x, y) = (0, 0)$. The aircraft subsequently turns right. By $t = 3.7\ s$ the pole at $(x, y) = (0, 0)$ is to the left and mostly out of sight of the aircraft. However, the aircraft now sees the pole at $(x, y) = (15, 15)$ directly in front. To avoid this pole, and head towards the goal, the aircraft turns left. At $t = 4.2\ s$ the aircraft is in the middle of the left bank turn. The aircraft continues to fly straight, and by $t = 7\ s$ the aircraft has flown past all of the obstacles. The aircraft enters the goal region at $t = 8\ s$ and executes the cruise-to-hover maneuver, and holds the hover for the remainder of the flight.

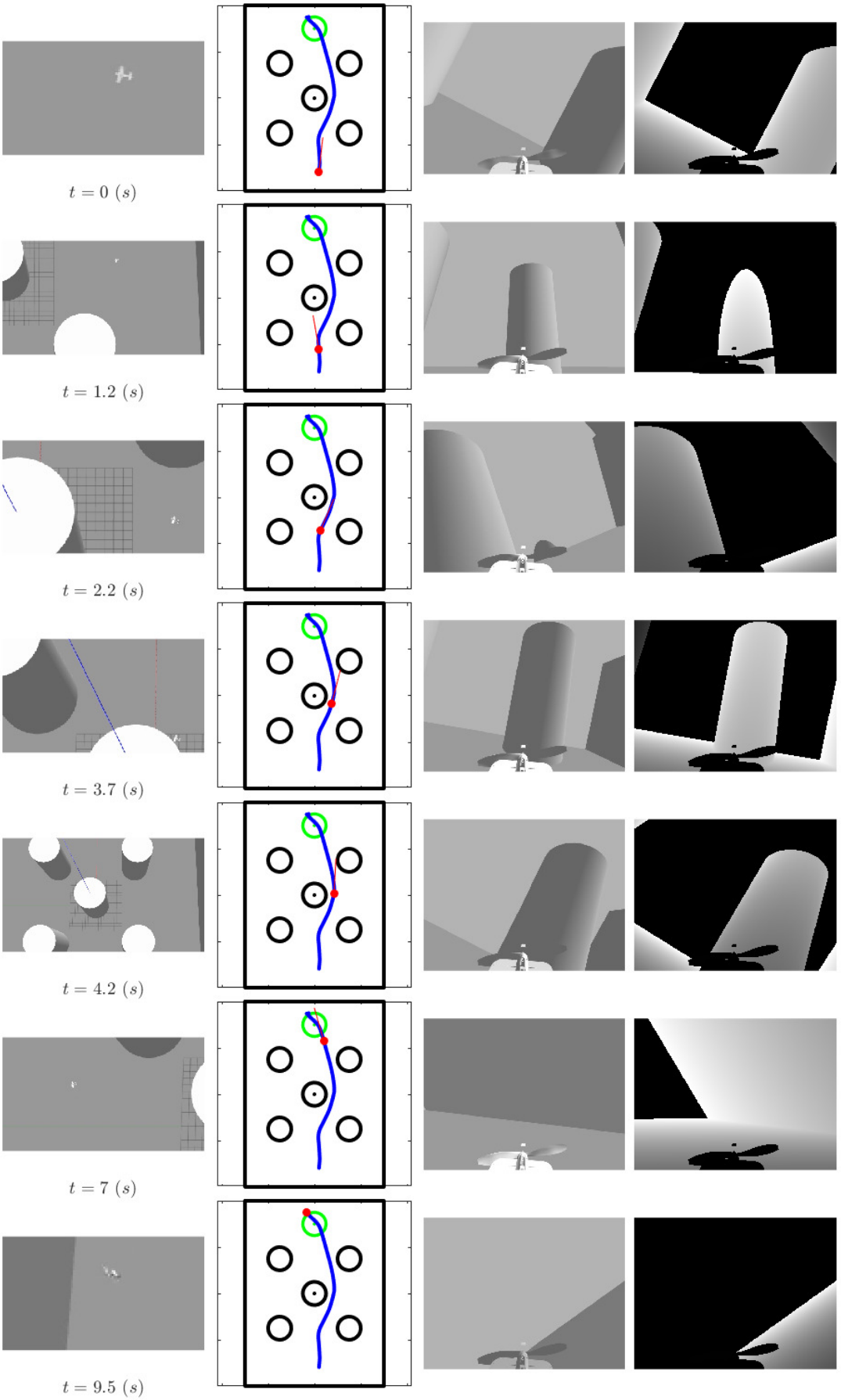


FIGURE 6.4: Environment 1 at 9 $\frac{m}{s}$, Run 3: aerial image, flight trajectory, on-board color image, and on-board depth image

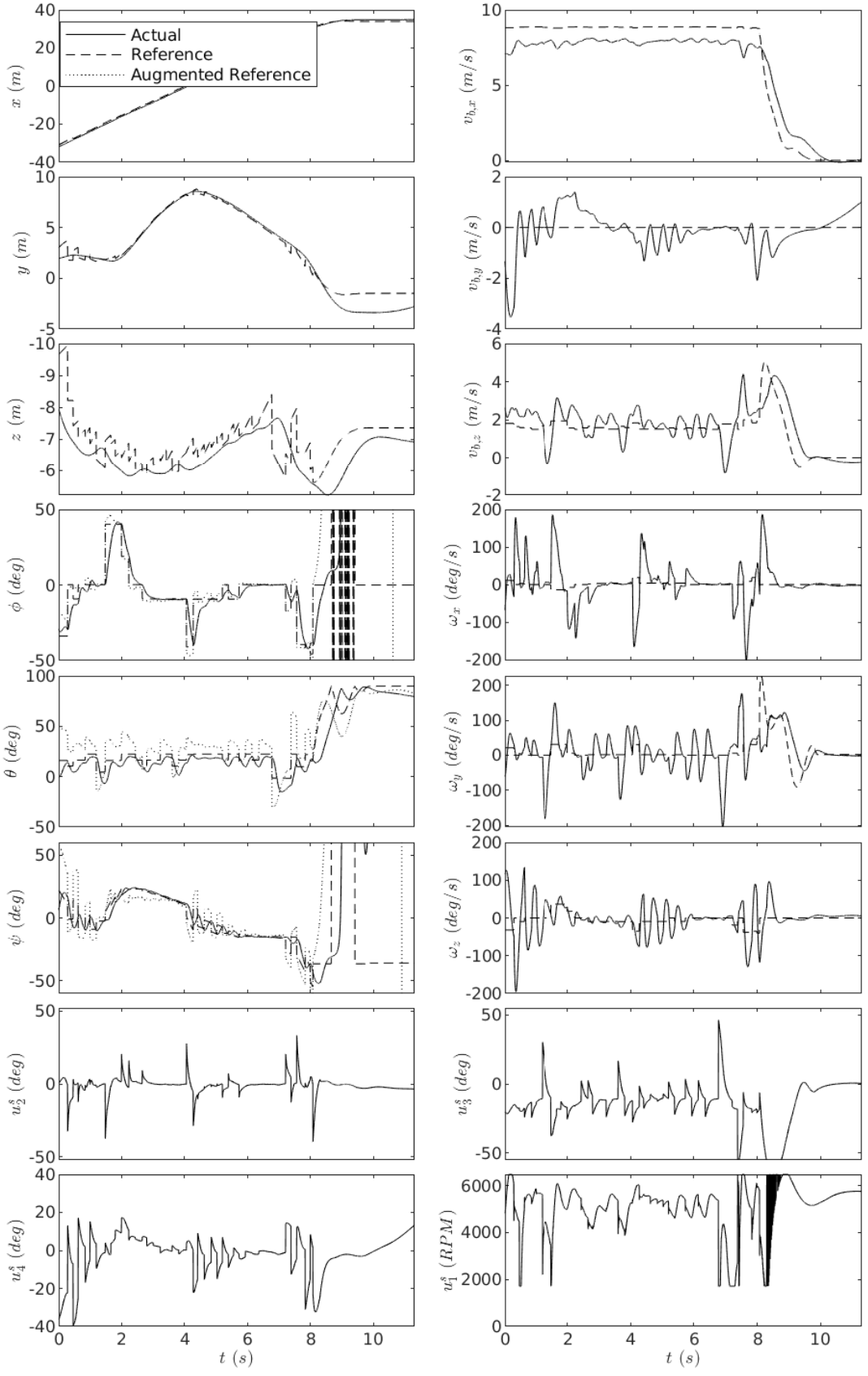
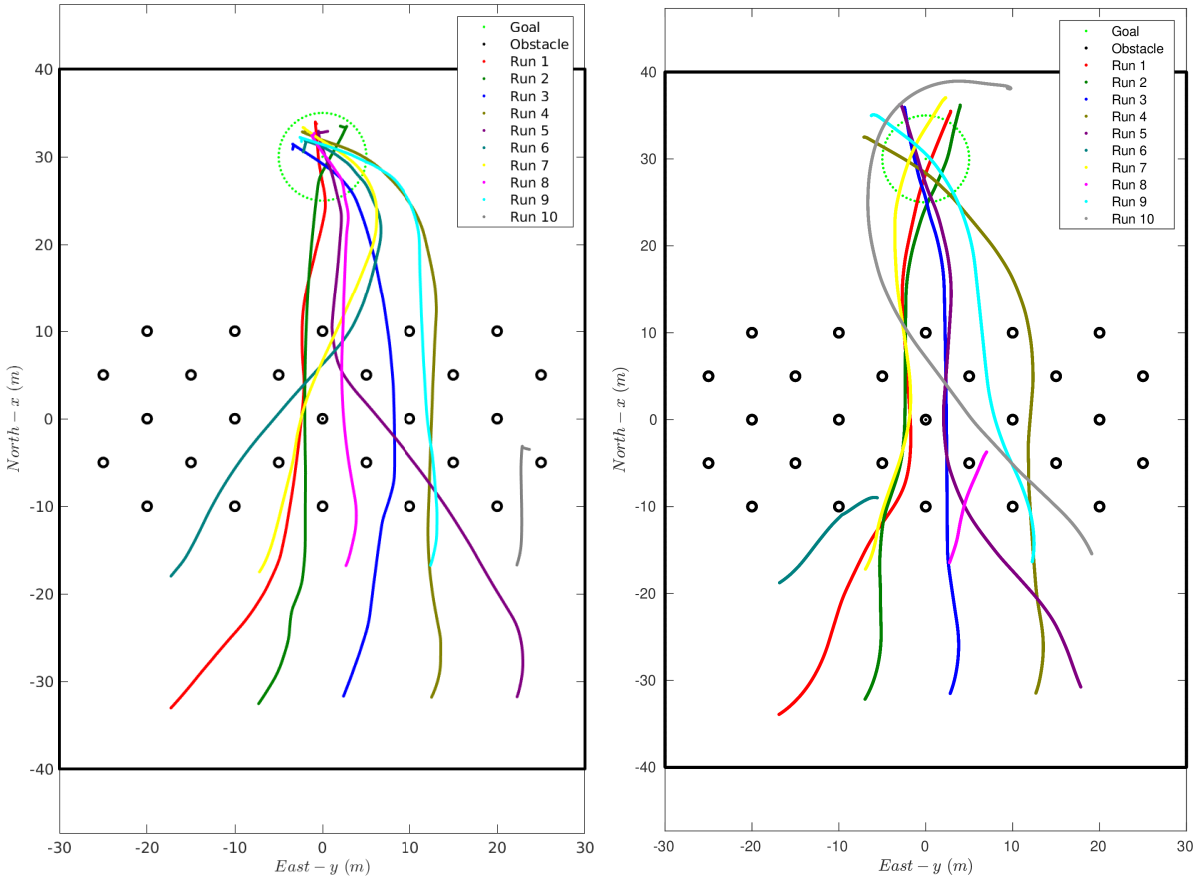


FIGURE 6.5: Environment 1 at 9 $\frac{m}{s}$, Run 3: state estimates, reference states, and control inputs

6.2.2 Environment 2

Our second test environment is more challenging, containing an area with densely spaced tall thin poles, resembling a forest. The environment consists 5 rows of tall poles spaced 10 m apart in the East direction. Alternating rows are shifted by 5 m in the East/West direction and spaced 5 m apart in the North/South direction. Cumulatively there are 27 poles, each with a 0.5 m radius, and enclosed in a 60 $m \times 80$ m rectangle. A top-down view of the environment and flight trajectories is shown for the trajectories using the 9 $\frac{m}{s}$ trajectory library in Fig. 6.6a, and for the trajectories using the 13 $\frac{m}{s}$ trajectory library in Fig. 6.6b. Collisions were successfully avoided in all 10 runs at both speeds. Using the 9 $\frac{m}{s}$ trajectory library, 9 runs reach the goal while 1 run ends with an emergency hover within the rows of poles. Using the 13 $\frac{m}{s}$ trajectory library, 7 runs reach the goal while 3 runs ends with an emergency hover. Out of the emergency hovers, 2 runs hover within the rows of poles, while 1 run makes it through the rows of poles, but cannot turn sharply enough to reach the goal, and ends up hovering near the wall.



(a) 9 $\frac{m}{s}$

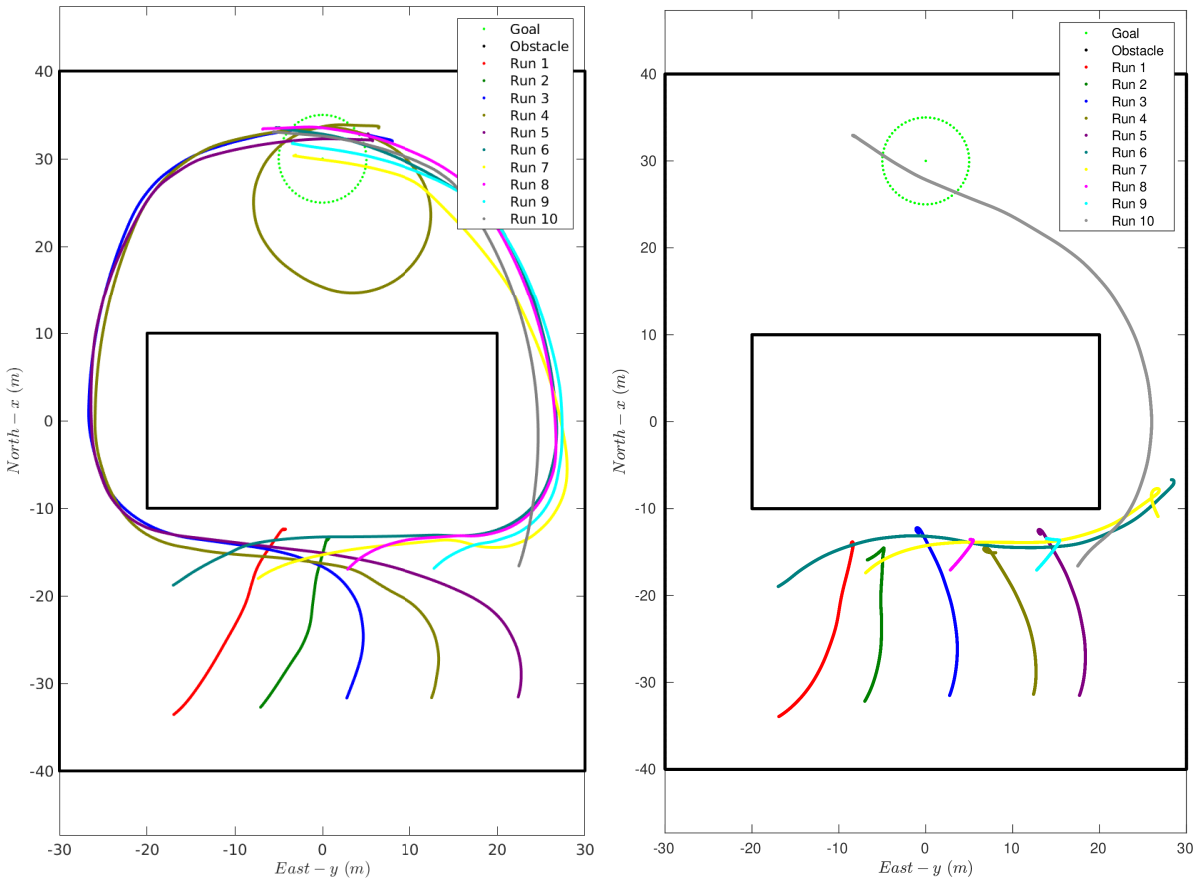
(b) 13 $\frac{m}{s}$

FIGURE 6.6: Environment 2

6.2.3 Environment 3

The third test environment is used to test the aircraft approaching a wall. Environment 3 consists of a 40 $m \times 20$ m rectangular obstacle enclosed by a 60 $m \times 80$ m rectangular wall. This environment would be a relatively simple environment to maneuver through if planning with a known map of the environment, but is more difficult when only planning locally. At 9 $\frac{m}{s}$, shown in Fig. 6.7a, all 10 flights avoid collisions, but only 8 reach the goal. In runs 1-5, the aircraft initially heads toward rectangular obstacle since that is the same direction towards the goal, and the obstacle has yet to be detected. In runs 3-5, the aircraft turns before reaching the rectangle, and eventually reaches the goal. In runs 1 and 2, by the time the aircraft start turning right to avoid the rectangle, there is not enough room to complete the turn, and the aircraft hovers to avoid a collision. In runs 6-10, the aircraft immediately sees the rectangular obstacle, and initially flies parallel to the front side. The aircraft can then turn sharply enough to avoid hitting the outer wall, and eventually reaches the goal. While it is unintuitive that the runs starting closer to the obstacle have better success in maneuvering around the obstacle, the reason is the aircraft is not initially facing the obstacle (in all runs), and because the aircraft is closer to the obstacle, it can detect the obstacle before turning itself directly towards it, which enables the aircraft to maneuver around it. In runs 1-5, the aircraft cannot detect the rectangle before it turns itself toward it.

At 13 $\frac{m}{s}$, shown in Fig. 6.7b, the aircraft has much less success reaching the goal. While it does avoid collisions in all 10 flights, it only reaches the goal once. Due to the larger turning radius at higher speed flight, the aircraft cannot turn sharply enough to avoid hitting the rectangle and outer walls, and thus hovers to avoid collisions. In run 10, where the aircraft does reach the goal, the aircraft is not required to make any sharp turns due to its initial pose.



(a) 9 $\frac{m}{s}$

(b) 13 $\frac{m}{s}$

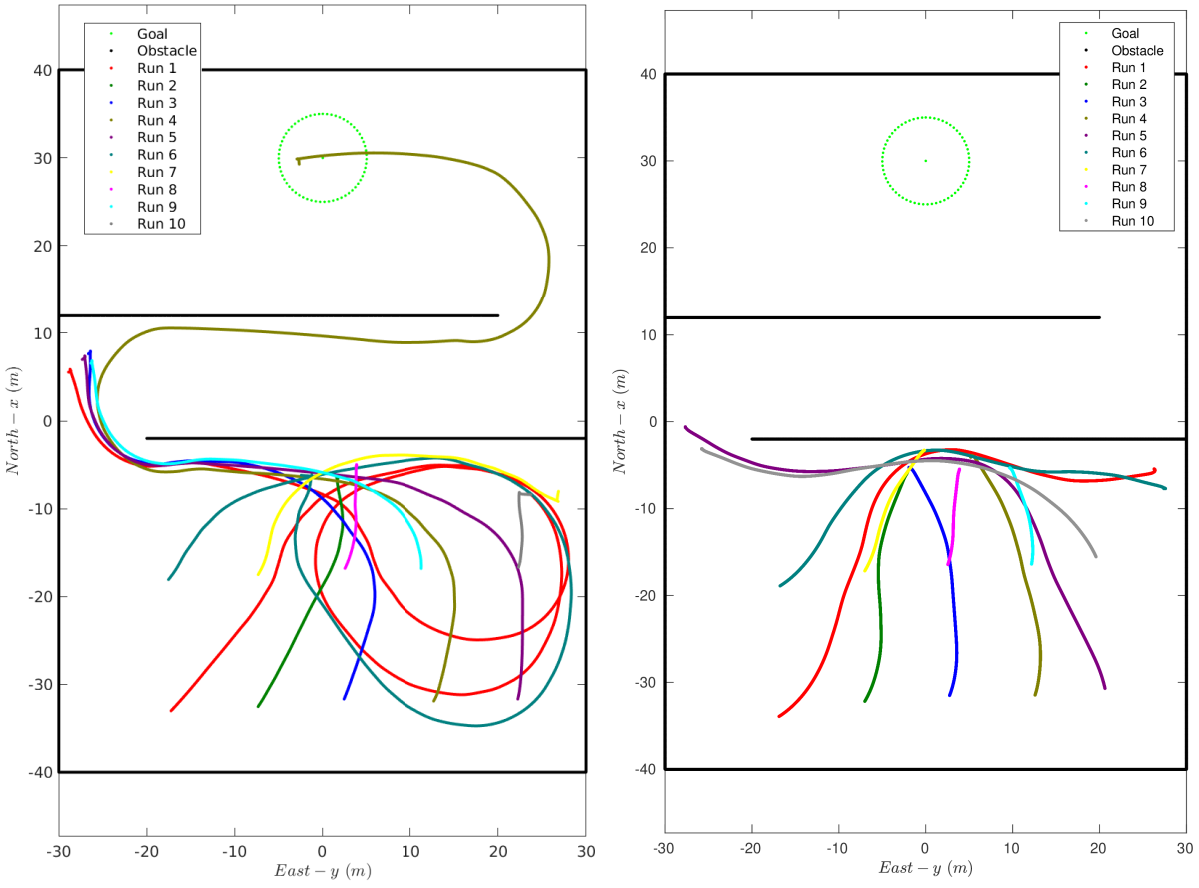
FIGURE 6.7: Environment 3

6.2.4 Environment 4

Environment 4 is a labyrinth type environment. Similarly to the third environment, this environment is much harder to navigate through using local planning than if the entire obstacle-map was known a priori. Looking at the 9 $\frac{m}{s}$ trajectories, shown in Fig. 6.8a, all of the runs avoid collisions, but only one reaches the goal. When trajectories are only planned locally, they don't factor in the distant future motion. As we can see in multiple runs (runs 1,3,5,9), the aircraft has difficulty completing the U-turn to pass through the opening and avoid the walls. The aircraft has difficulty making this turn because it is unaware of the second east-west wall until passes through the opening, giving the aircraft little room to plan around. During the one run which completes the U-turn (run 4), the aircraft initiated the U-turn slightly south in comparison to the runs which ended in an emergency hover, which enabled the aircraft to finish the U-turn.

Turning our attention to the 13 $\frac{m}{s}$ trajectories, shown in Fig. 6.8b, none of the trajectories

reach the goal, but all avoid collisions. In comparison to the 9 $\frac{m}{s}$ flights, the aircraft has a larger turning radius–and can never execute a U-turn in through the opening. An environment like this demonstrates the need for a global planner and SLAM implementation, while still showcasing the obstacle avoidance's capability to prevent collisions in difficult environments.



(a) 9 $\frac{m}{s}$

(b) 13 $\frac{m}{s}$

FIGURE 6.8: Environment 4

6.3 Outdoor Flight Tests

Our obstacle avoidance flight testing campaign began in August 2019 and continued through November 2019, at which time flight testing was suspended for the colder months and the subsequent COVID-19 lockdown. We resumed flight testing in June 2020 through November 2020. Due to the various regulations regarding COVID-19, our access to flight testing locations was highly variable. Flight tests were conducted at the Macdonald Campus of McGill University, the West Island Model Aeronautics Club (WIMAC), the

Montreal Area Thermal Soarers Club (MATS), and the cottage of a generous McGill Professor. All of these locations are either in or near Montreal, Canada, and contain large plots of land with a mix of open space, sparse trees, and dense forest.

A great deal of time was spent at the flight testing sites diagnosing and fixing hardware and software issues unrelated to the actual obstacle avoidance methodology. Initially, the Intel RealSense D435 depth camera output too many false-positive points within the point cloud, which was largely resolved by using the settings presented in [106]. Another issue arose due to USB3 connection to the Intel Realsense, which interfered with the GPS signal. This issue was mostly resolved by shielding the electronics with copper foil and moving the GPS further from the electronics. The last major issue was a bug in the trajectory generation code, which was easily fixed once diagnosed.

We completed 6 days of flight testing with all the aforementioned issues resolved, where we were able to focus on gathering experimental results with the completely autonomous aircraft. We executed 35 runs, where each run consists of hand launching the aircraft; manually flying the aircraft to a region where a tree must be avoided in order to reach the goal location; switching the control authority from manual to autonomous; autonomously avoiding trees; and then concluding by either (a) hovering at the goal, (b) executing an emergency hover, or (c) colliding with a tree. Once the aircraft enters a hover, control is given back to the pilot, who manually lands the aircraft. At any instant during these flights, any error could cause the aircraft to crash into a tree.

The 35 autonomous runs take place in six different test environments. While we are satisfied with experimental validation in six environments, the multiple environments are more a consequence of where permission is obtained to fly the aircraft, rather than an intentional testing strategy. In Figs. 6.9 - 6.13, we show the testing environments using Google Earth, and approximately mark the start and goal locations on the images. For Environments 1 & 6, which are both at WIMAC, three-dimensional renderings are available. For the remaining environments, only top-down two-dimensional images are available.

Thoroughly explaining the aircraft's motion during an autonomous run requires a large volume of data, including state estimates, reference states, external video images, on-board color images, on-board depth images, and trajectories overlaid onto satellite images. For this reason, we summarize the results for all of the runs using various statistics in Sec. 6.3.1, followed by a brief analysis of all the runs individually using only top-down satellite imagery in Sec. 6.3.2, and finally a detailed analysis of four selected runs in Sec. 6.3.3.

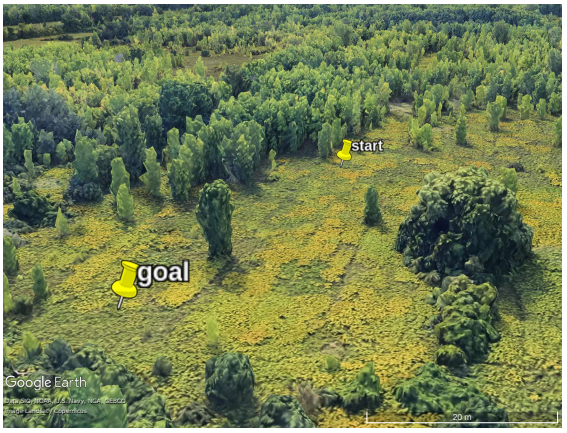


FIGURE 6.9: Environment 1 (at WIMAC)

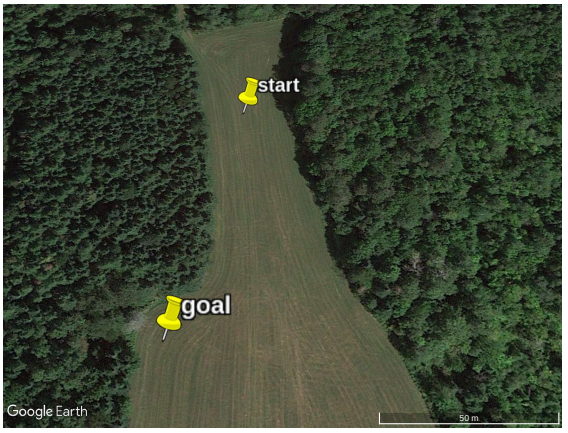


FIGURE 6.10: Environment 2 (at cottage)

FIGURE 6.11: Environment 3 (at MATS)

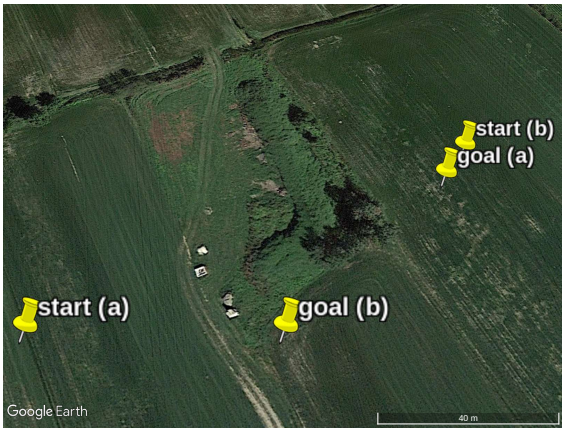


FIGURE 6.12: Environment 4 (start/-goal (a)) & 5 (start/goal (b)) (at MATS)

FIGURE 6.13: Environment 6 (at WIMAC)

6.3.1 Summary

Table 6.2 summarizes the 35 autonomous runs where, for each run, we specify the environment (Env.), date, the trajectory library speed (Lib.), the configuration of the control and obstacle avoidance algorithm (Conf.), the total distance traveled in m (Dist.), the duration of the run in s, the average speed in $\frac{m}{s}$ (Avg.Spd), the maximum speed in $\frac{m}{s}$ (Max.Spd), and whether the run ended because the aircraft reached the goal, executed an emergency hover, or collided with a tree. During the flight testing campaign we made a few modifications to the control and obstacle avoidance algorithm, which are denoted by configuration a, b, c and d, and discussed in Sec. 6.3.2.

In total, the aircraft autonomously flew 4.4 km over 543 s while avoiding trees, and maintained an average speed of 8.1 $\frac{m}{s}$ and a top speed of 14.4 $\frac{m}{s}$. Out of the 35 runs, 16 reached the goal, 15 ended with an emergency hover, and 4 collided with a tree. Given the difficulty of the research problem at hand, we are very satisfied with these results. For comparison, the most relevant existing research that attempts to autonomously fly a fixed-wing aircraft through an unknown and unstructured environment using only onboard computation and sensing is in [78, 79]. In that work, the aircraft crashed into a tree 10 times out of 26 runs. Thus, our 4 collisions out of 34 runs is significantly better than the state-of-the-art. Furthermore, none of the collisions were the fault of the obstacle avoidance algorithm; one collision was caused by a mechanical failure resulting in loss of aileron control, and the other three were a result of a camera failure.

6.3.2 High-Level Analysis

We show top-down views of the flight trajectories of each run overlaid onto a Google satellite images of the test environments in Figs. 6.14 - 6.22. Looking at Fig. 6.14, we can see Runs 1-4 all avoid the tree in the middle of the image, and Run 4 also avoids the trees in the bottom right of the image. While Runs 1, 3 & 4 all reach the goal, Run 2 ends with an emergency hover.

As shown in the image, the emergency hover is executed in free space. The reason for this hover is as follows: the aircraft significantly pitches up (likely because of a wind gust) for a short period of time. During that time, the obstacle avoidance algorithm computes the trim trajectories that lead to the edge of the FOV. Since, the camera is fixed to the aircraft, and the aircraft is significantly pitched up, all the positions along the edge of the FOV require a large climb rate to reach those positions. If all of the these climb rates are not feasible to achieve using trim primitives, the aircraft executes an emergency hover, to

Run	Env.	Date	Lib.	Conf.	Dist.	Time	Avg.Spd	Max.Spd	Outcome
1	1	15/09/20	9	a	78.00	10.20	7.64	7.91	reached goal
2	1	15/09/20	9	a	53.79	7.17	7.51	8.22	emergency hover
3	1	15/09/20	9	a	108.05	14.23	7.59	8.23	reached goal
4	1	15/09/20	9	a	134.90	18.03	7.48	9.02	reached goal
5	1	24/09/20	9	a	219.19	33.83	6.48	8.85	emergency hover
6	1	24/09/20	9	a	66.33	8.48	7.82	11.85	emergency hover
7	1	24/09/20	9	a	135.00	19.45	6.94	9.48	emergency hover
8	1	24/09/20	9	a	123.77	17.07	7.25	9.98	emergency hover
9	1	24/09/20	11	a	221.35	26.02	8.51	11.13	emergency hover
10	1	24/09/20	11	a	34.15	3.95	8.64	9.84	emergency hover
11	1	24/09/20	11	a	55.67	7.43	7.49	9.56	emergency hover
12	1	24/09/20	11	a	121.89	14.59	8.36	10.50	emergency hover
13	1	24/09/20	11	a	101.92	11.20	9.10	12.56	emergency hover
14	1	24/09/20	11	a	283.24	36.58	7.74	11.78	emergency hover
15	2	31/10/20	11	b	129.32	15.99	8.09	11.04	reached goal
16	2	31/10/20	11	b	102.23	12.90	7.92	14.42	reached goal
17	2	31/10/20	11	b	155.95	20.22	7.71	13.11	collision
18	2	31/10/20	11	b	117.20	13.23	8.86	11.00	collision
19	3	9/11/20	9	c	160.25	17.96	8.92	10.25	reached goal
20	3	9/11/20	9	c	58.58	6.94	8.44	9.35	emergency hover
21	3	9/11/20	9	c	39.25	4.96	7.91	9.83	emergency hover
22	4	13/11/20	9	d	153.97	17.98	8.56	10.63	reached goal
23	4	13/11/20	9	d	250.37	29.93	8.37	11.11	reached goal
24	4	13/11/20	9	d	142.09	16.97	8.37	10.94	reached goal
25	4	13/11/20	9	d	175.45	21.47	8.17	13.49	reached goal
26	4	13/11/20	9	d	131.25	15.57	8.43	11.59	reached goal
27	4	13/11/20	9	d	205.21	24.29	8.45	11.01	collision
28	5	13/11/20	9	d	106.25	11.90	8.93	10.39	emergency hover
29	5	13/11/20	9	d	115.59	12.85	8.99	10.75	reached goal
30	5	13/11/20	9	d	113.28	12.16	9.31	9.95	reached goal
31	5	13/11/20	9	d	138.25	15.65	8.84	9.36	reached goal
32	5	13/11/20	9	d	90.43	11.23	8.05	12.96	emergency hover
33	1	19/11/20	9	d	102.76	11.44	8.98	9.93	reached goal
34	6	19/11/20	9	d	119.07	13.00	9.16	9.68	reached goal
35	6	19/11/20	9	d	46.43	7.82	5.93	8.75	collision

TABLE 6.2: Summary of Runs

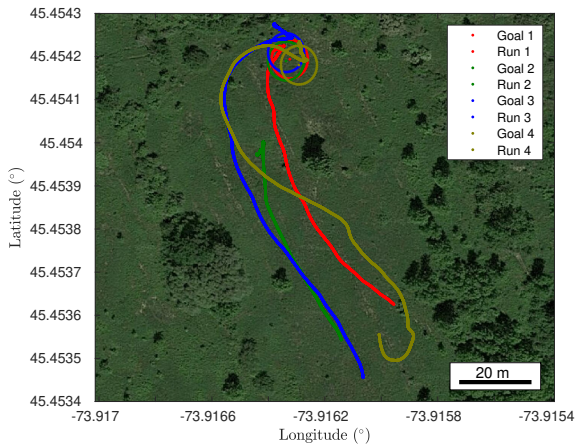


FIGURE 6.14: Top-Down View from Runs 1-4

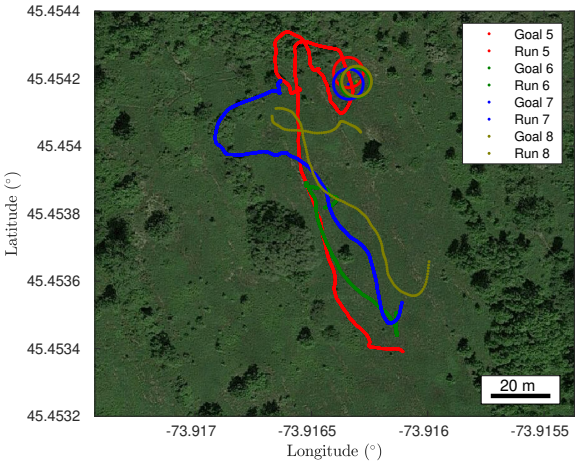


FIGURE 6.15: Top-Down View from Runs 5-8

avoid flying in a space which it cannot see. Emergency hovering for this reason also occurs in Runs 5, 6, & 12. Hovering in these scenarios is a conservative approach. For Runs 1-14, we use this conservative approach, which we label configuration 'a' in Table. 6.2. Another option in this scenario would be to continue on the previously selected reference trajectory until the next motion planning iteration. We use this approach for the remainder of the runs, which is the case for configurations 'b', 'c', and 'd' in Table. 6.2.

Runs 5-8, which use a 9 $\frac{m}{s}$ trajectory library, are shown in Fig. 6.15 and Runs 9-14, which use an 11 $\frac{m}{s}$ trajectory library, are shown in Fig. 6.16. An interesting phenomenon occurs when initializing the aircraft heading away from the goal. In Runs 4, 6, 7 & 8, all which use a 9 $\frac{m}{s}$ trajectory library, are able to turn and avoid the very large tree (right side of Fig. 6.9, middle of Fig. 6.15). Runs 10 & 11, which use an 11 $\frac{m}{s}$ trajectory library, both cannot turn in time to avoid that tree and execute an emergency hover, likely because of the fewer available turning radii in the higher speed trajectory library.

Runs 5-14 occur on the same day, which was windy. Strong winds affect the aircraft's motion beyond pitching up the aircraft and prematurely hovering. In many instances, the aircraft does not take a direct route to the goal. This can be attributed to two reasons. First, say a wind gust yaws the aircraft. Since the aircraft always selects trajectories starting from its current yaw angle, if a wind gust yaws the aircraft, the flight trajectory will completely change, since the aircraft will now select trajectories starting from its new yaw angle. Second, say there is a constant wind pushing the aircraft to the right. If the aircraft wants to turn right, the reference trajectory will originate from the aircraft location while turning right. However, during the time the motion plan is computing, the wind will cause the aircraft to be right of the initial part of the right-turning trajectory, so then the position controller will direct the aircraft to the left. This attempt to stay on

the originally planned trajectory results in longer turns and ultimately an indirect path to the goal.

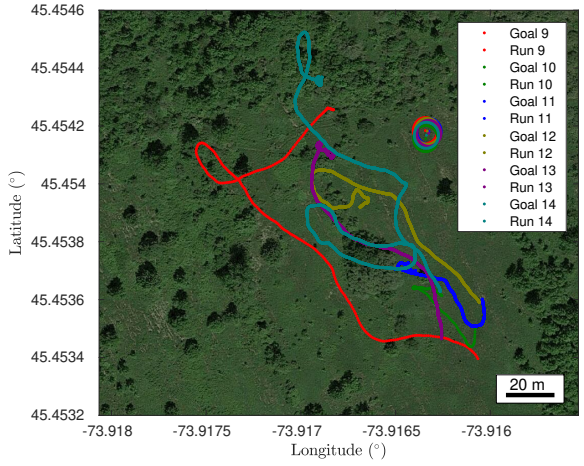


FIGURE 6.16: Top-Down View from Runs 9-14

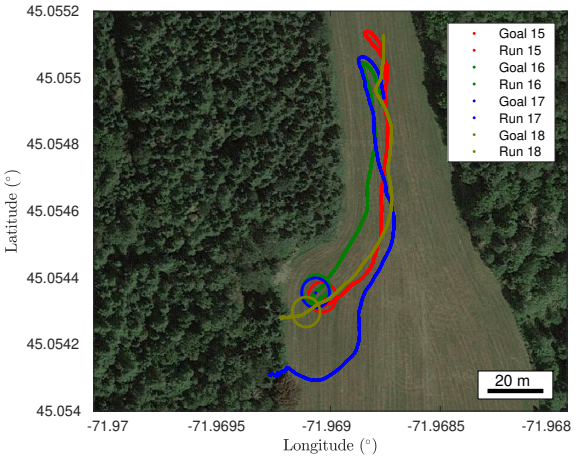


FIGURE 6.17: Top-Down View from Runs 15-18

Looking at Fig. 6.17, the aircraft needs to avoid a wall of dense trees, and then turn after the wall to the reach the goal. The aircraft succeeds at reaching the goal in Runs 15 & 16, but collides with a tree in Runs 17 & 18. During both collisions, the depth camera failed to detect parts of the tree leading up to the collision, and the aircraft therefore flew into what it thought was free space, but was not.

Between Runs 18 & 19, a coding mistake was found in the force controller presented in Sec. 3.2; a sign error was present in $\mathbf{f}_b^{aero}$. In addition to fixing this error, the control gains were also adjusted between these runs. In the first 18 runs, a modification to the force control gains are made while following trim maneuvers in comparison to agile maneuvers. The value of K_v is doubled while K_{h_p} and K_{h_i} are set to zero, as shown in Table 6.1. While this is advantageous in simulation, the presence of wind causes this modification to be less effective in experiments. Thus, this modification is removed, and the same force control gains are used regardless of the maneuver, for the remainder of the experiments. We also added an integrator in the attitude controller, which improved the control performance while hovering. Lastly, the minimum thrust is increased to 3500 RPM, which ensures the aircraft always has airflow over its control surfaces, even in the case of a strong tailwind. All of these changes are denoted by configurations 'c' and 'd' in Table 6.2.

Looking at Fig. 6.18, Run 19 reaches the goal, while Runs 20 & 21 end with an emergency hover. The satellite image is slightly deceiving, as this test occurred in November when the leaves had fallen. So while it appears that Run 19 goes either above or through the tree, it actually goes around. In all of these runs, a creek separating the grass fields (along

the tree line), causes many false positive points to appear in the point cloud, which has a large impact on the aircraft's motion.

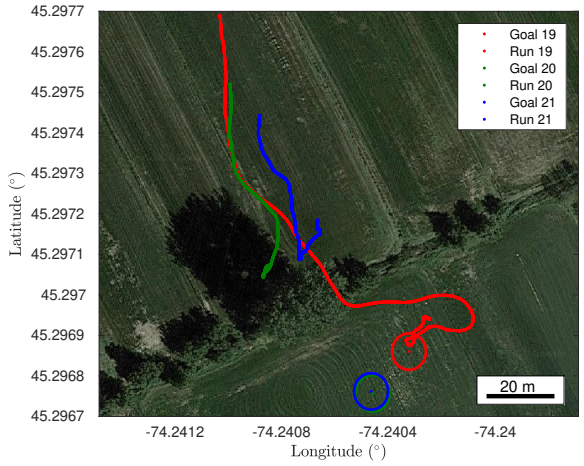


FIGURE 6.18: Top-Down View from Runs 19-21

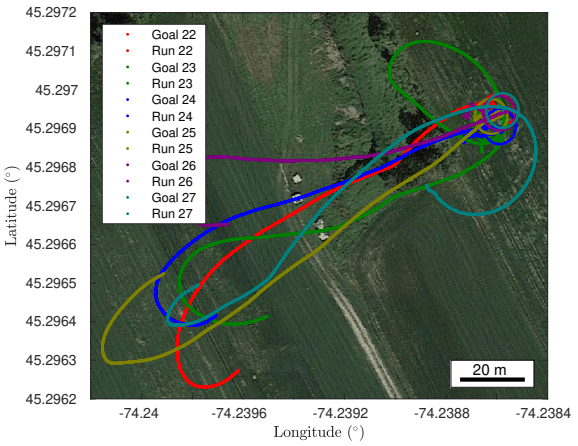


FIGURE 6.19: Top-Down View from Runs 22-27

The obstacle avoidance algorithm will always rely on GPS to reach a desired GPS location. However, testing whether obstacles could still be avoided without GPS is useful, in the case of a temporary loss of GPS signal. The remainder of the runs are conducted with $K_{p_p} = 0$ and $K_{p_d} = 0$, except when hovering. By doing this, the feedback controller no longer relies on position estimates. So while these runs do use GPS (and position estimates) to give priority to trajectories that reach the goal, the position estimate is not used to avoid obstacles. This modification is represented by configuration 'd' in Table 6.2.

The obstacles in Environment 4 & 5 (Fig. 6.12) consist of a mound, which gradually increases altitude on the left side, and has a sharp drop on the right side. On this mound there is a tree on each end. We force the aircraft to approach the mound from each side by varying the start and goal, as shown in Figs. 6.19 & 6.20. When approaching the obstacles from the left side (Fig. 6.19), the aircraft has a tendency to go over the tree, as it starts to climb first because of the mound, and by the time it approaches the tree, it is high enough to either continue straight or slightly climb to fly over the tree. This is shown in Runs 24, 25 & 27. In Run 27, the aircraft is too high above the goal's altitude to enter a hover, and ends up turning back towards the tree. Due to a camera malfunction, the camera fails to detect the tree in time to avoid crashing into it. This run is discussed in detail in Sec. 6.3.3.4. In Runs 22, 23 & 26, the aircraft goes around the tree and reaches the goal, even though it is hard to see due to outdated satellite imagery.

Referring to Fig. 6.20, all of the runs go around the tree. Runs 29-31 reach the goal, while runs 28 & 32 execute an emergency hover. In these runs, the emergency hover

was executed due to having false positive points in the point cloud which resulted in the aircraft thinking it had no collision-free options.

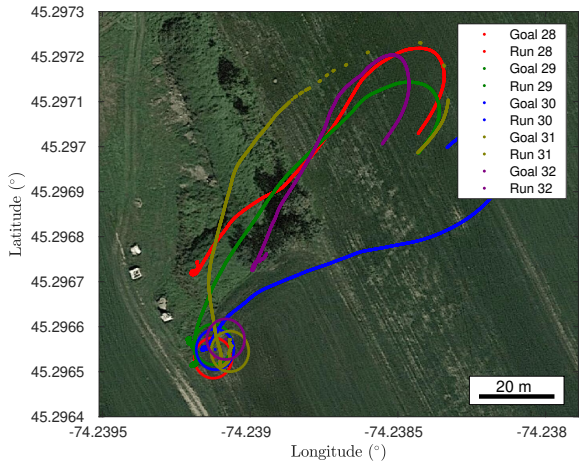


FIGURE 6.20: Top-Down View from Runs 28-32

FIGURE 6.21: Top-Down View from Runs 33

Looking at Fig. 6.21, the aircraft reaches the goal by flying around the tree. Referring to Fig. 6.22, in Run 34 the aircraft avoids several trees and then reaches the goal (although hard to discern due to outdated satellite imagery). In Run 35, a mechanical failure causes a loss of control of the aileron. The aircraft loses control in roll and subsequently flies straight into a tree.

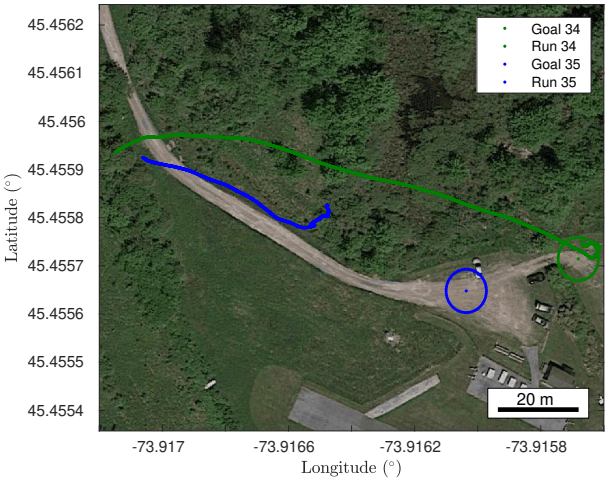


FIGURE 6.22: Top-Down View from Runs 34-35

6.3.3 Detailed Analysis

While the flight trajectories overlaid on to satellite imagery briefly describe the runs, we now proceed to analyze four runs in more detail. We choose to analyze one successful

run that includes position control (Run 4); one successful run that does not use position control to avoid obstacles (Run 22); one run that ends with an emergency hover (Run 20); and one run that ends with a collision (Run 27). At selected times, we show the image from a ground video, the on-board color image, and the on-board depth image. The depth image shows black pixels where there are no obstacles in the point cloud, and a grey-scale for obstacles, where the closer obstacles are represented by darker pixels. While the color and depth image are of roughly the same scene, they do have different fields-of-view. In addition to images, we overlay the flight trajectory onto a satellite view of the flight area. The trajectory line starts blue, and becomes lighter with time, and goal region is represented by the green circle. At the same instants where the images are shown, we display the aircraft's location with a red dot, and its yaw with a red line. In addition to these images, we show plots of the reference states, state estimates, and control inputs. The goal location is considered to be $(x, y, z) = (0, 0, -10)$ m, and 'reaching' the goal occurs when the aircraft is within 5 m of the goal location.

6.3.3.1 Successful Run using Position Control (Run 4)

We present a detailed analysis of Run 4, which successfully avoids trees and reaches the goal. A video from the flight can be seen at `https://youtu.be/Nd54c5_dc64`. From left to right we show the image from the ground video, the satellite imagery, the on-board color image, and the on-board depth image in Fig. 6.23, for the first seven seconds of the run, and in Fig. 6.24 for the remainder of the run. The reference states, state estimates, and control inputs (throttle (u_1^s), aileron (u_2^s), elevator (u_3^s), rudder (u_4^s)) are shown in Fig. 6.25 for the first seven seconds, in Fig. 6.26 between seven and fifteen seconds, and in Fig. 6.27 for the remainder of the flight.

As shown in both Figs. 6.23 & 6.25, the aircraft is initialized heading away from the goal. The aircraft banks left to turn around, and by $t = 2.45$ s, the aircraft has completely turned around. At this instant, flying straight would be the most direct path to the goal, but the aircraft continues banking left since it sees the tree in front of it, which is shown in the second row of Fig. 6.23. By $t = 2.7$ s, the aircraft no longer sees the tree in front of it, and subsequently banks right in order to fly towards the goal. By $t = 3.33$ s, the aircraft sees tree again, and banks left to avoid it.

At 5.6 s, the aircraft has avoided the first tree, and sees a new tree located in the middle of the satellite image, which is shown in the fifth row of Fig. 6.23. The aircraft initiates a mild right turn, which steers the aircraft not too far from the goal, and would allow the aircraft to pass the right side of the tree. However, while the trajectory was being

selected, the aircraft deviated far from the initial position and yaw in which the trajectory is being originated from since the aircraft had turned quickly in its previous time step. So although the trajectory was meant to turn right, which can be shown by the increasing yaw reference, the start of that increasing yaw reference is less than the current yaw. The position controller augments the reference yaw to be even more negative, which induces a large rudder deflection (u_4^s) and yaws the aircraft to the left (negatively).

By 6 s, the aircraft no longer sees the tree, and subsequently banks right to go towards the goal. At 6.5 s the aircraft detects the tree, and subsequently banks even more aggressively to the right, as going right of the tree is the shortest collision-free path to the goal. At 7 s, the aircraft still sees the tree, and continues to select right bank trajectory in order to fly to the right of the tree.

The same issue that occurs at 5.6 s occurs again. As we can see in Fig. 6.26 the first trajectory switch after 7 s has an increasing yaw, but is initialized at the yaw of around 7 s which is much less than the yaw at the time of the trajectory switch. In addition, since the position references are generated based on that 'out-dated' yaw, a large position error causes the augmented reference yaw to be even more negative. A large rudder deflection (u_4^s) negatively yaws the aircraft, and by 8 s the aircraft is directed left of the tree (in Fig. 6.24, the tree being avoided is on the right side of the ground video image in the first two rows). The aircraft continues to select straight and right turning trajectories as it passes the left side of the tree. By 13.7 s, the aircraft has passed the tree, and turns right until the aircraft reaches the goal at 17.8 s. At this instant, the aircraft enters a hover and holds the hover the remainder of the run.

Throughout the run, many of the state variables and control inputs have oscillatory behavior (see Figs. 6.25 - 6.27), which can be attributed to the frequent switching of the reference trajectory.

The issue of having an 'out-dated' initial position and yaw reference was apparent during this run. While it did lead to a longer path than necessary, the issue did not cause the aircraft to crash. The way to properly address this issue would be to predict the initial position and yaw based on the computation time of the obstacle avoidance algorithm, and select the reference trajectories from that predicted position and yaw. However, making this prediction would require accurate wind sensing and an accurate dynamics model which can be run on-board in real-time, and both of these are difficult to achieve within our payload limitations.

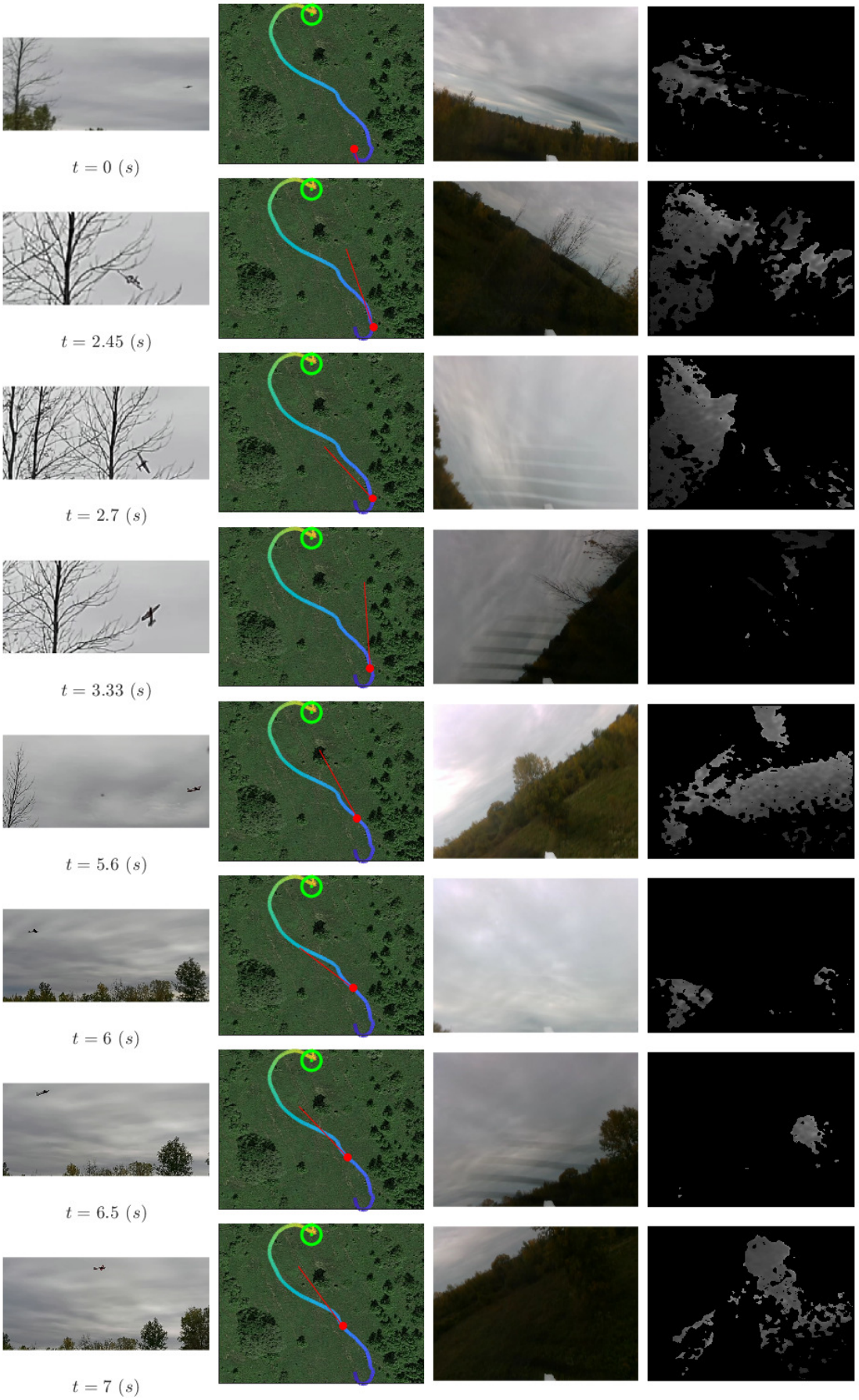


FIGURE 6.23: Run 4: ground image, flight trajectory, on-board color image, and on-board depth image ($t = 0 - 7s$)

FIGURE 6.24: Run 4: ground image, flight trajectory, on-board color image, and on-board depth image ($t = 7 - 20s$)

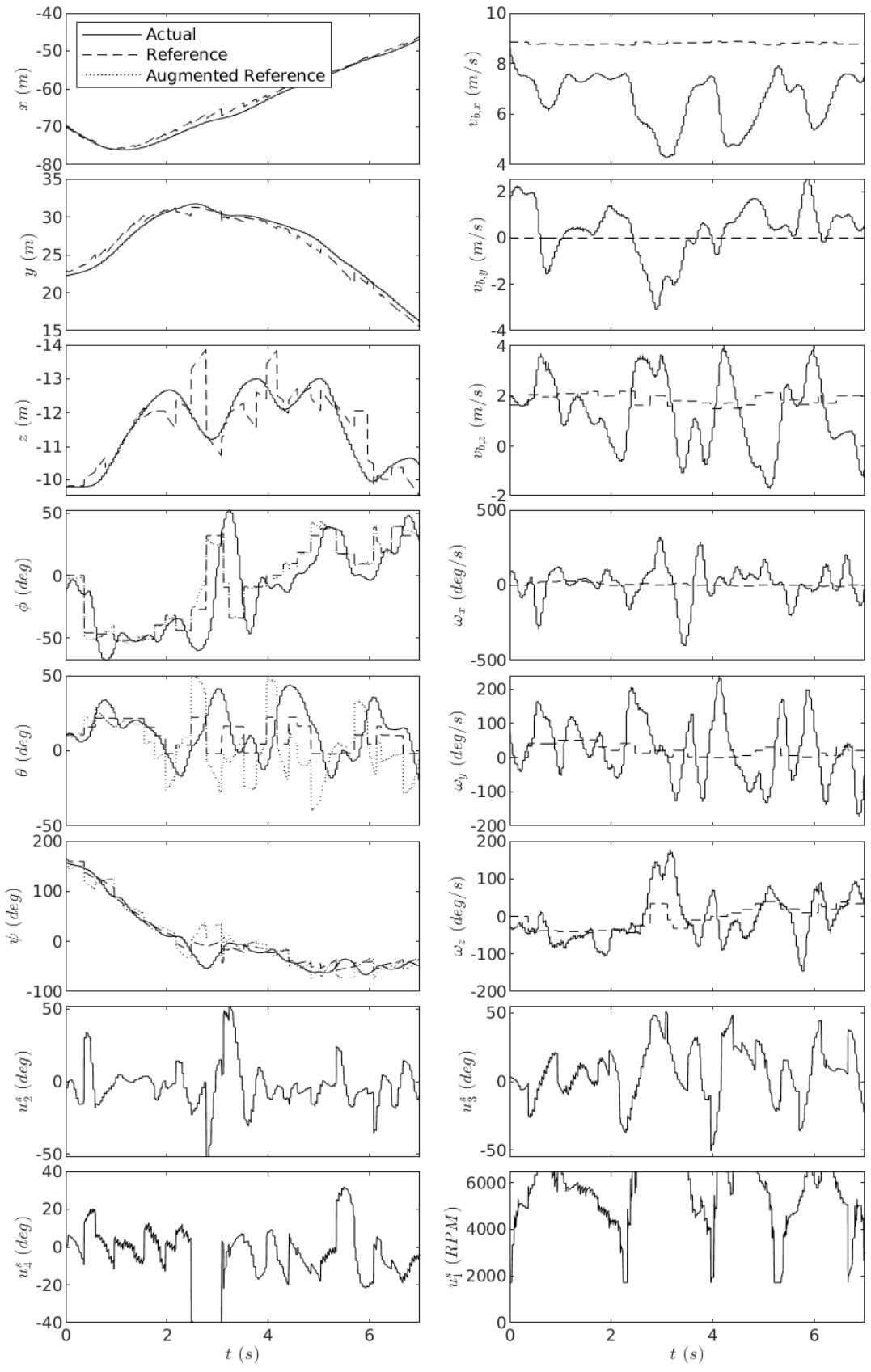


FIGURE 6.25: Run 4: state estimates, reference states, and control inputs ($t = 0 - 7s$)

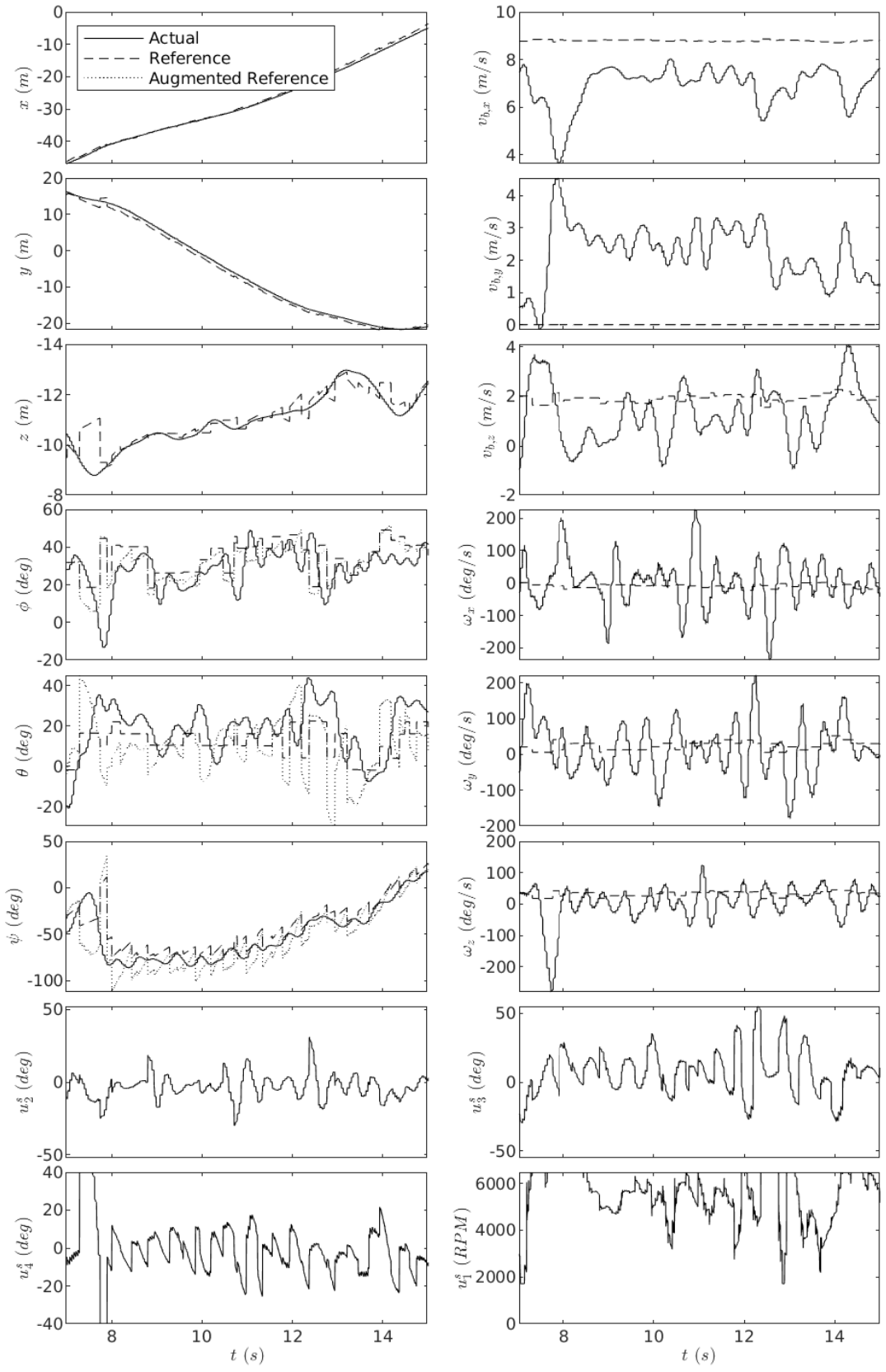


FIGURE 6.26: Run 4: state estimates, reference states, and control inputs ($t = 7 - 15s$)

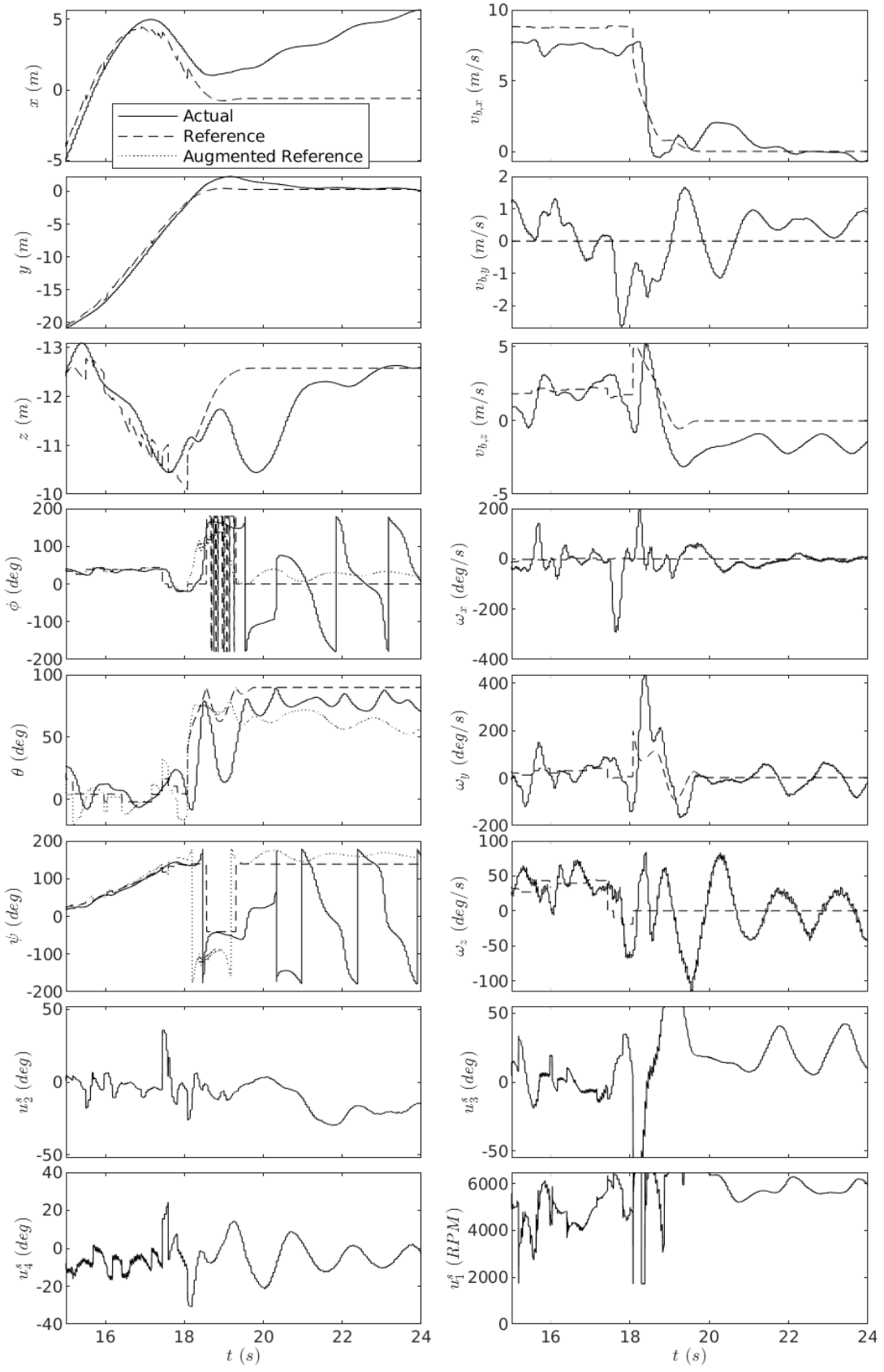


FIGURE 6.27: Run 4: state estimates, reference states, and control inputs ($t = 15-24s$)

6.3.3.2 Successful Run without using Position Control (Run 22)

We now present Run 22 in greater detail, which does not rely on position estimates for avoidance. Images from the run can be found in Fig. 6.28, while plots of the reference states, state estimates, and control inputs (throttle (u_1^s), aileron (u_2^s), elevator (u_3^s), rudder (u_4^s)) can be found in Fig. 6.29, and a video can be found at `https://youtu.be/AhiVG7kXNY8`. Looking at Fig. 6.28, the aircraft is initalized heading a away from the goal. By $t = 5$ s, the aircraft has turned around, and heads toward the goal. The aircraft continues to execute level flight and mild banked turns as it flies toward the goal. Although at $t = 10$ s the aircraft sees part of the mound and at $t = 12$ s part of the tree, neither appear to cause a collision with the straight and mild bank turn trajectories, which the aircraft continues to execute. At $t = 13$ s, the aircraft clearly sees the tree in front of it, and sharply banks left, which can be seen at $t = 13.6$ s in Fig. 6.28, and in the roll plot in Fig. 6.29. Once the aircraft passes the tree, mild right bank turns are executed until reaching the goal around $t = 18$ s.

In comparison to Run 4, which uses position control to avoid obstacles, the trajectory in this run is smoother and more direct to the goal. A possible explanation for this relates to an issue previously mentioned, when the wind causes a mismatch with the initial position of the reference trajectory and the aircraft, and then the position controller corrects for this mismatch by turning in the opposite direction of where the reference trajectory eventually turns. A benefit of not using position control while avoiding obstacles is that this can never happen, since the position control gains are zero. Ultimately, the aircraft flies in a more direct route towards the goal.

A drawback to not using position control is that there is nothing to correct for deviations from the initially planned trajectory in position and velocity, which can cause the aircraft to fly along a slightly different position trajectory than the one that was deemed collision-free.

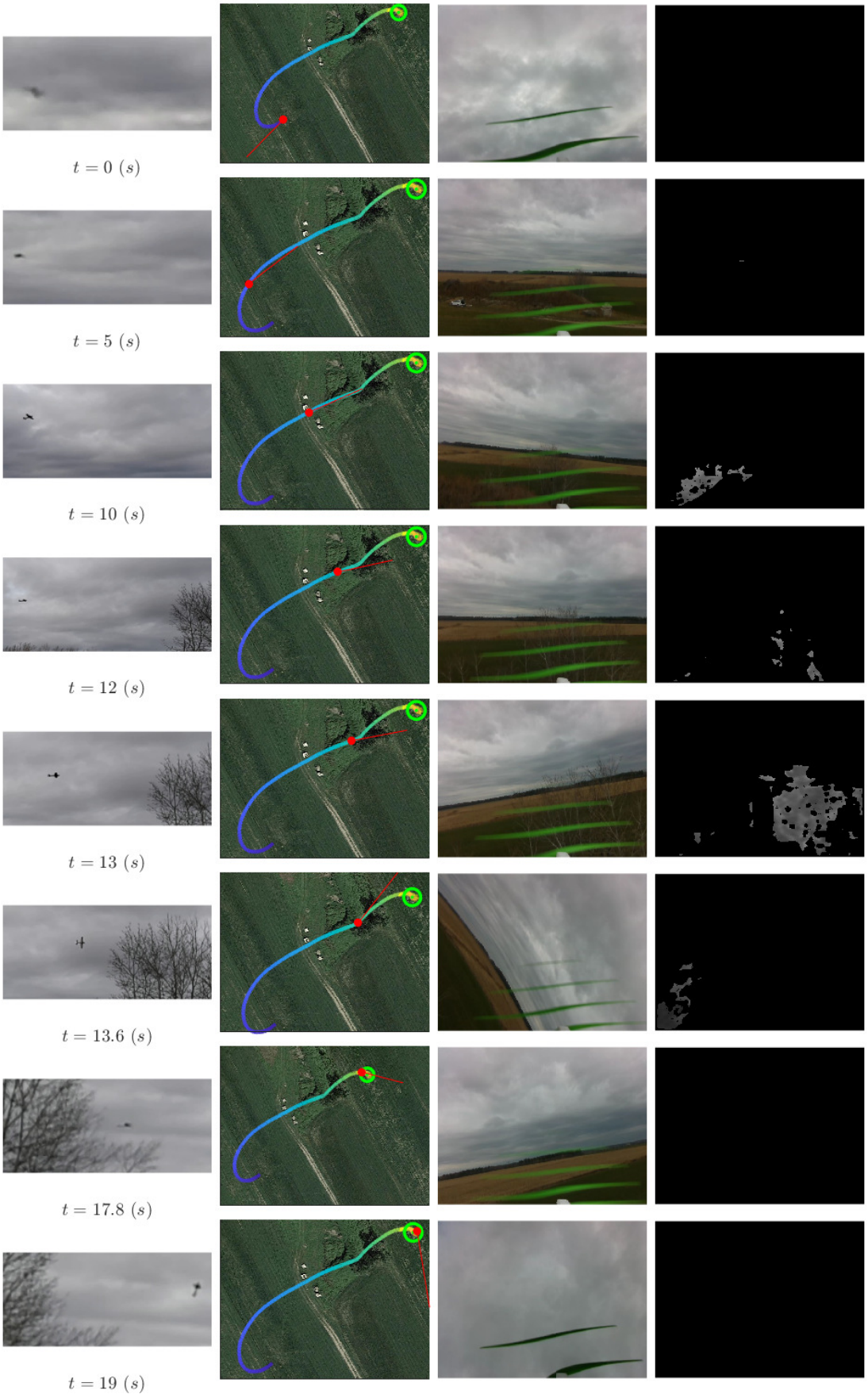


FIGURE 6.28: Run 22: ground image, flight trajectory, on-board color image, and on-board depth image

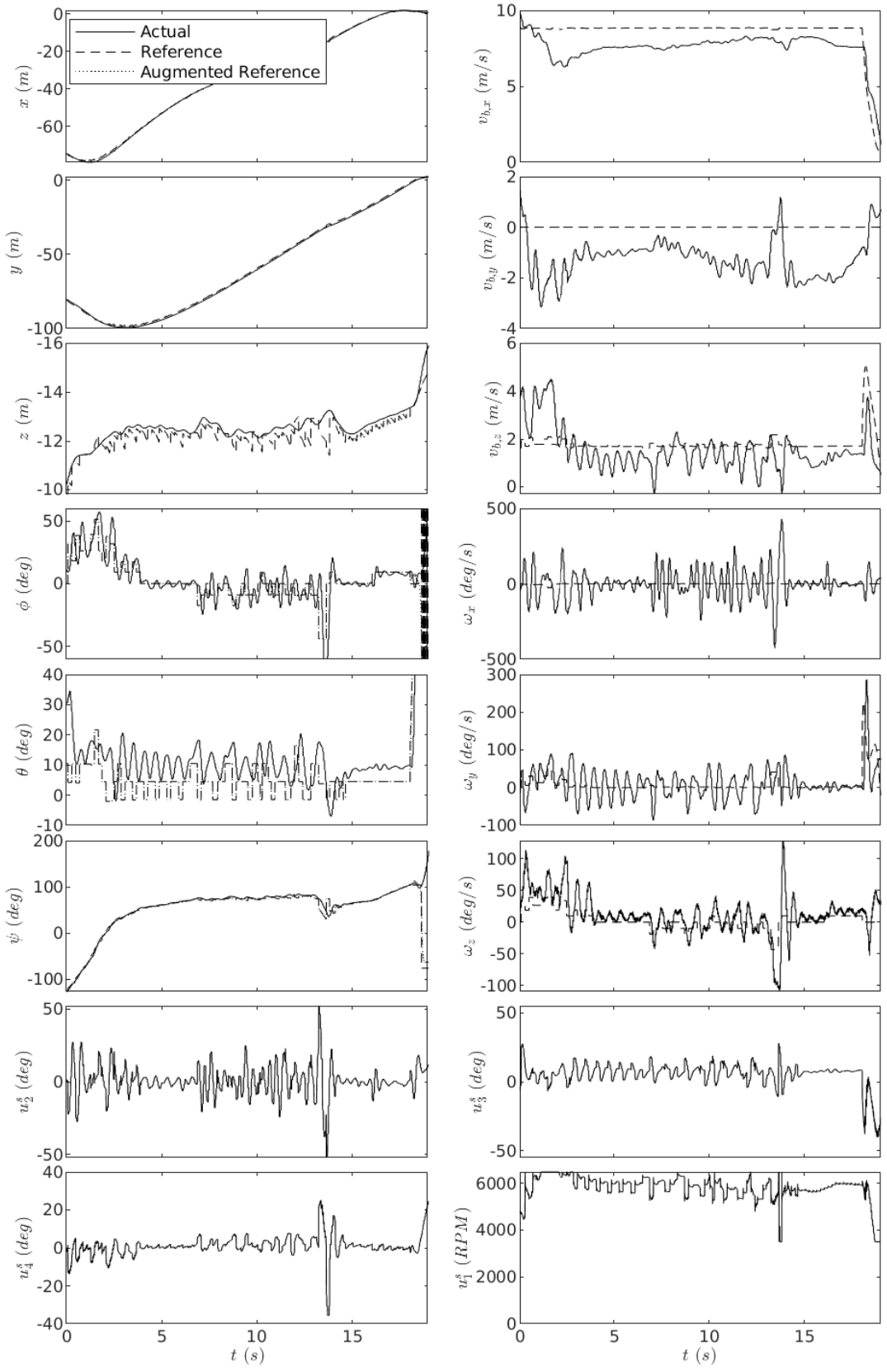


FIGURE 6.29: Run 22: state estimates, reference states, and control inputs

6.3.3.3 Emergency Hover (Run 20)

We now present a detailed description of a run that terminates in an emergency hover (Run 20), which can additionally be seen at https://youtu.be/lwPFORUAYEg. Images from the run are shown in Fig. 6.30, and plots are shown in Fig. 6.31. Looking at Fig. 6.30, the aircraft initially detects the tree, and subsequently banks right, shown in the roll in Fig. 6.31. Looking at Fig. 6.30 when $t = 1.6$ s, the creek separating the two grass fields causes false positive points in the point cloud. These false positive points cause the aircraft to bank left in order to avoid them, when looking at Fig. 6.31. A large wind gust occurs around $t = 3$ s, causing the roll angle to drop to $-100°$, which can be seen by the roll and sideslip speed ($v_{b,y}$) in Fig. 6.31. By $t = 3.5$ s, the aircraft recovers from the wind gust and banks right because the tree is not detected at that instant. Looking at Fig. 6.30 at $t = 4.6$ s, the aircraft stops banking right because part of the tree goes undetected, and the aircraft thinks flying straight is collision-free. At 5.1 s enough of the tree is detected and the aircraft decides to bank right for a short period of time until part of the tree is undetected again (seen at $t = 5.8$ s) where the aircraft executes a slight left turn. At about $t = 7$ s, the aircraft realizes it can no longer maneuver around the tree using the trim trajectory library, and executes an emergency hover. The aircraft is seen hovering at $t = 9.9$ s.

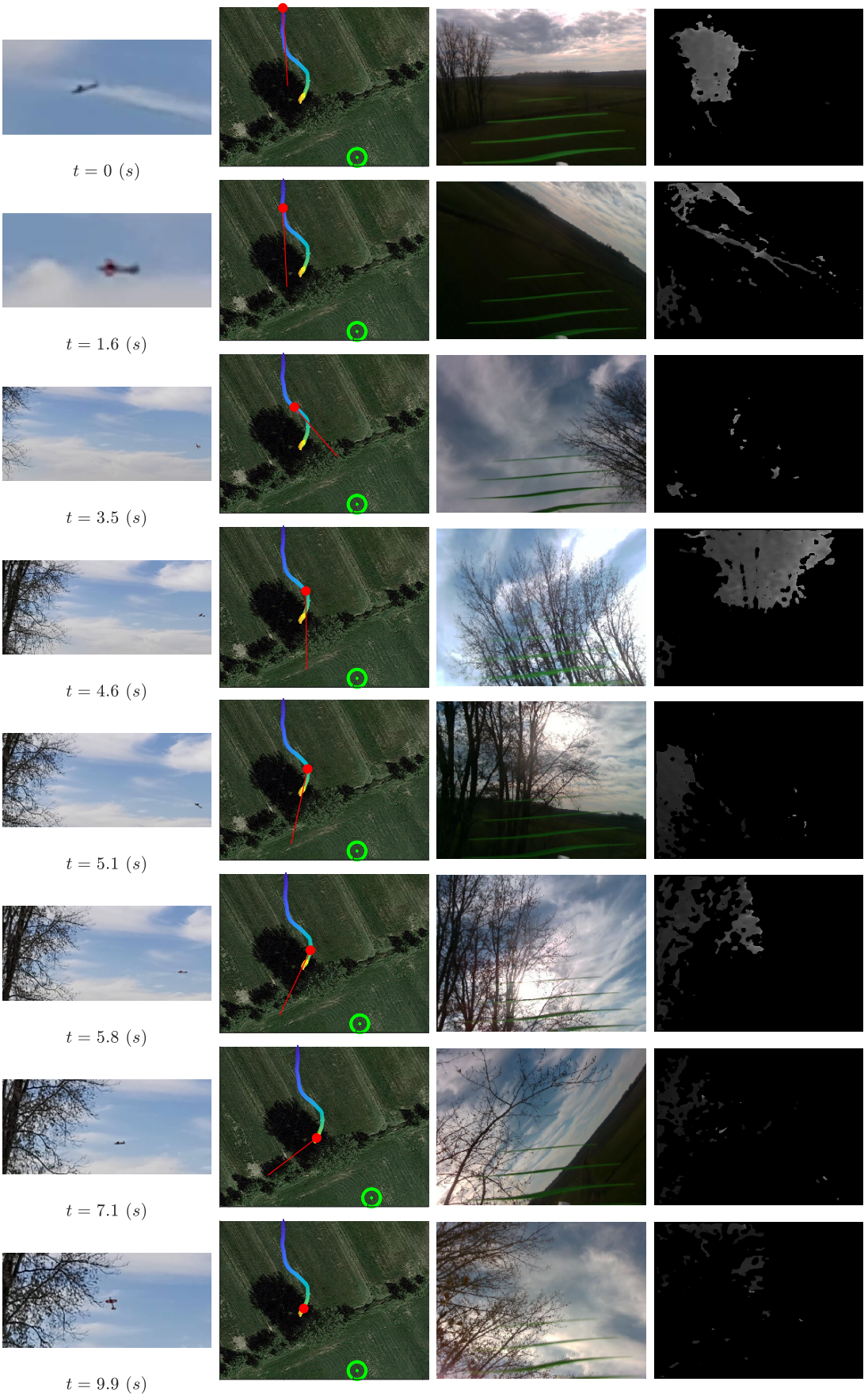


FIGURE 6.30: Run 20: ground image, flight trajectory, on-board color image, and on-board depth image

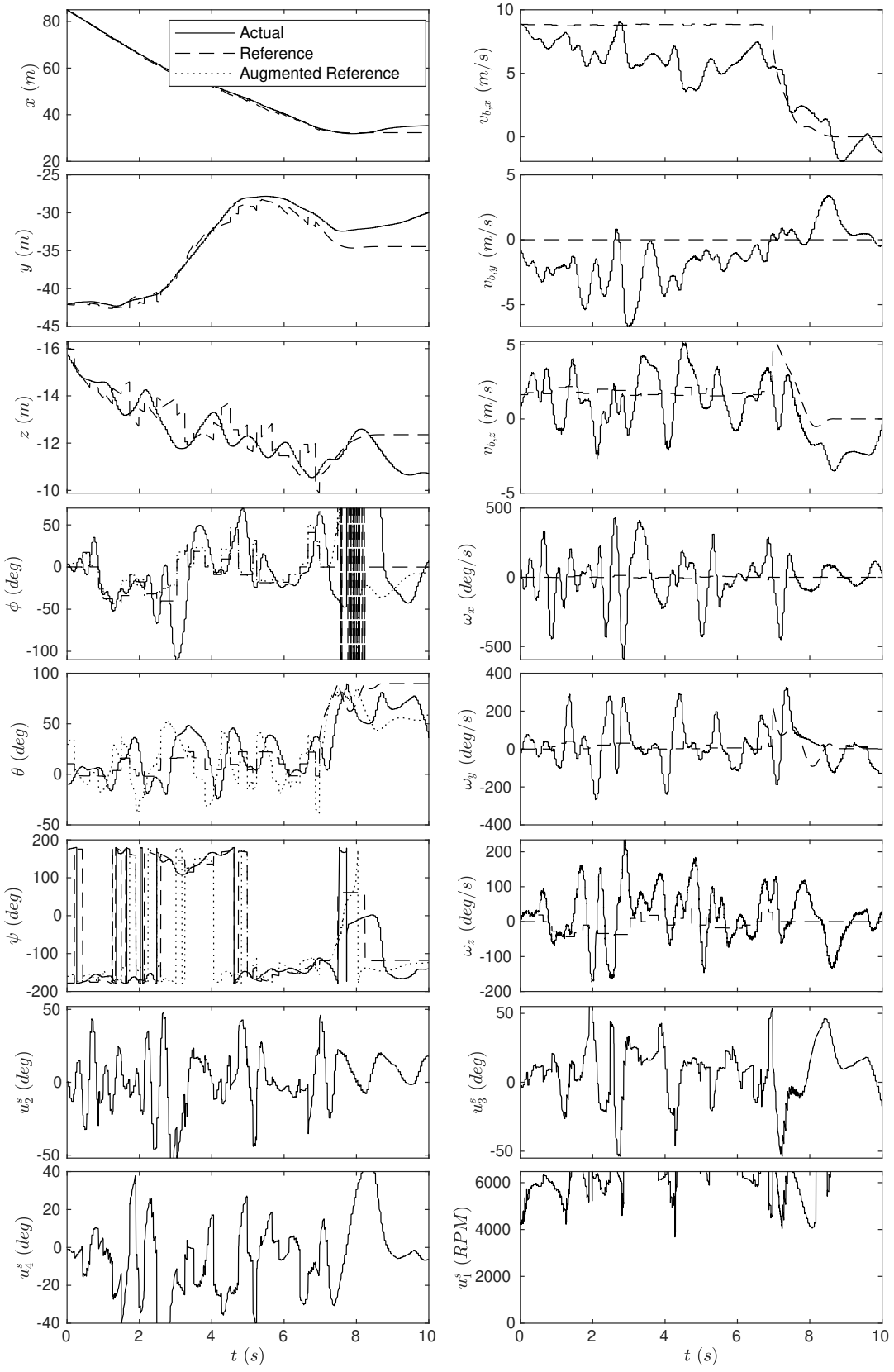


FIGURE 6.31: Run 20: state estimates, reference states, and control inputs

6.3.3.4 Collision (Run 27)

We now analyze in detail a run that ends in a collision caused by a camera failure (Run 27), and show the video of the collision in https://youtu.be/mPjxH4MMlzc. In Fig. 6.32, we show the image from a ground video, a top-down flight trajectory, the on-board color image, and on-board depth image, for the seconds leading up to the collision. For that same duration of time, we show the state estimate, reference states, and control inputs (throttle (u_1^s), aileron (u_2^s), elevator (u_3^s), rudder (u_4^s)) in Fig. 6.33.

During this run, the aircraft flew over the goal but did not enter a hover because it was too high above the goal altitude. After flying over the goal, the aircraft executes a descending right turn to loop back to the goal. We would have expected the aircraft to turn right until it detects the tree, and at that point would then turn left once realizing the right turn is not collision-free. However, as shown in the color image at 22.33 s, the tree is starting to appear on the right side, middle height. As the aircraft progresses, more of that tree is shown in the color images at 22.66 s and 23 s. However, during all this time, the depth image still does not detect the tree. Our algorithm relies on the depth camera detecting the tree at these times, as it is within the depth range and field-of-view (the field-of-view of the depth is larger than that of color). It is likely the camera does not detect the tree during these times because the branches and ground are a similar color. At 23.33 s and 23.66 s, parts of the tree have now been detected, but there are still large chunks that are undetected, shown by the black spots in the right side of the depth image. Looking at Fig. 6.33, the aircraft continues selecting banked right turns during that time, which means the undetected parts of the tree were large enough for the aircraft to deem those trajectories collision-free. The aircraft does not decide to enter an emergency hover until after 24 s, and at that point the collision is inevitable.

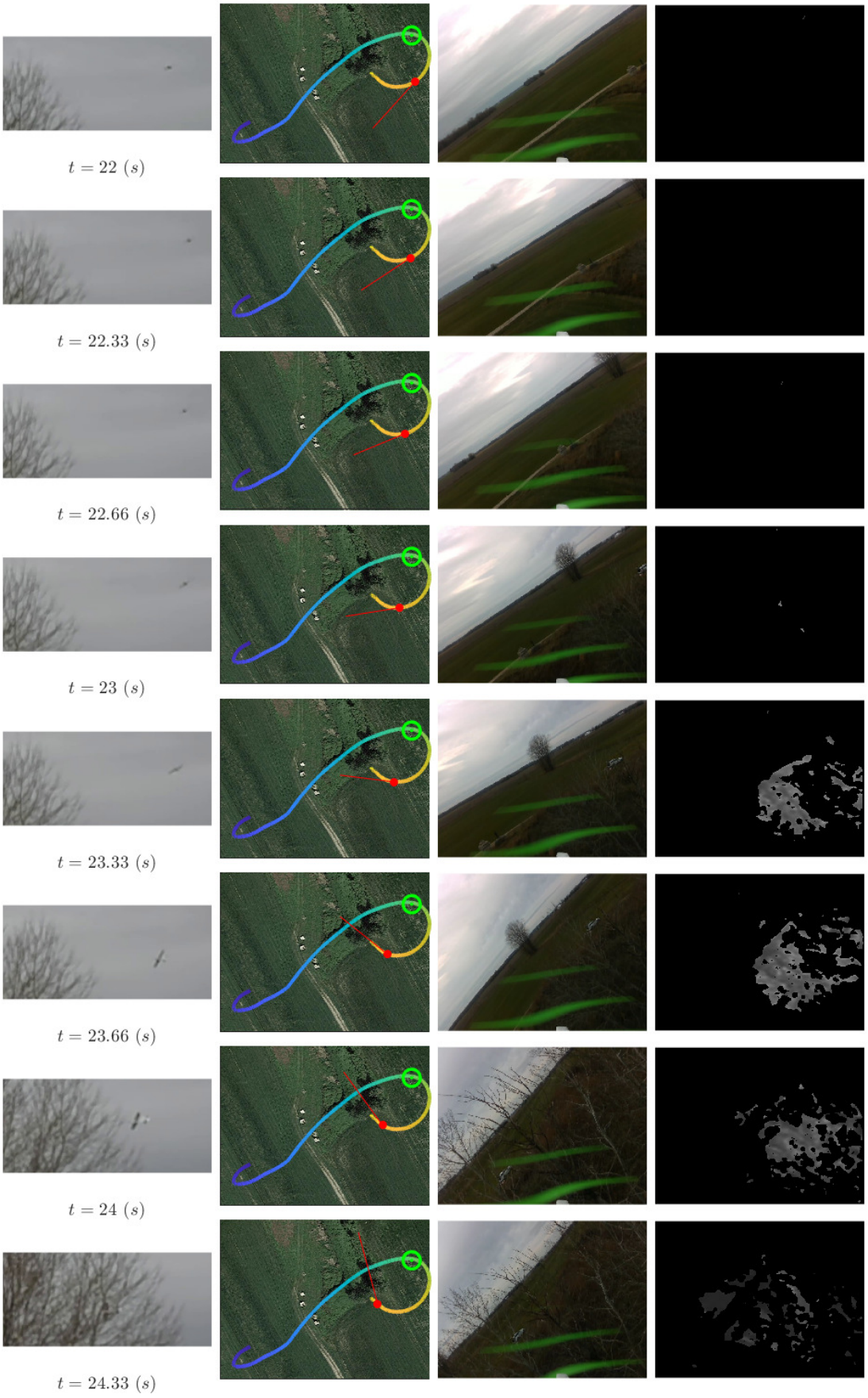


FIGURE 6.32: Collision: ground image, flight trajectory, on-board color image, and on-board depth image

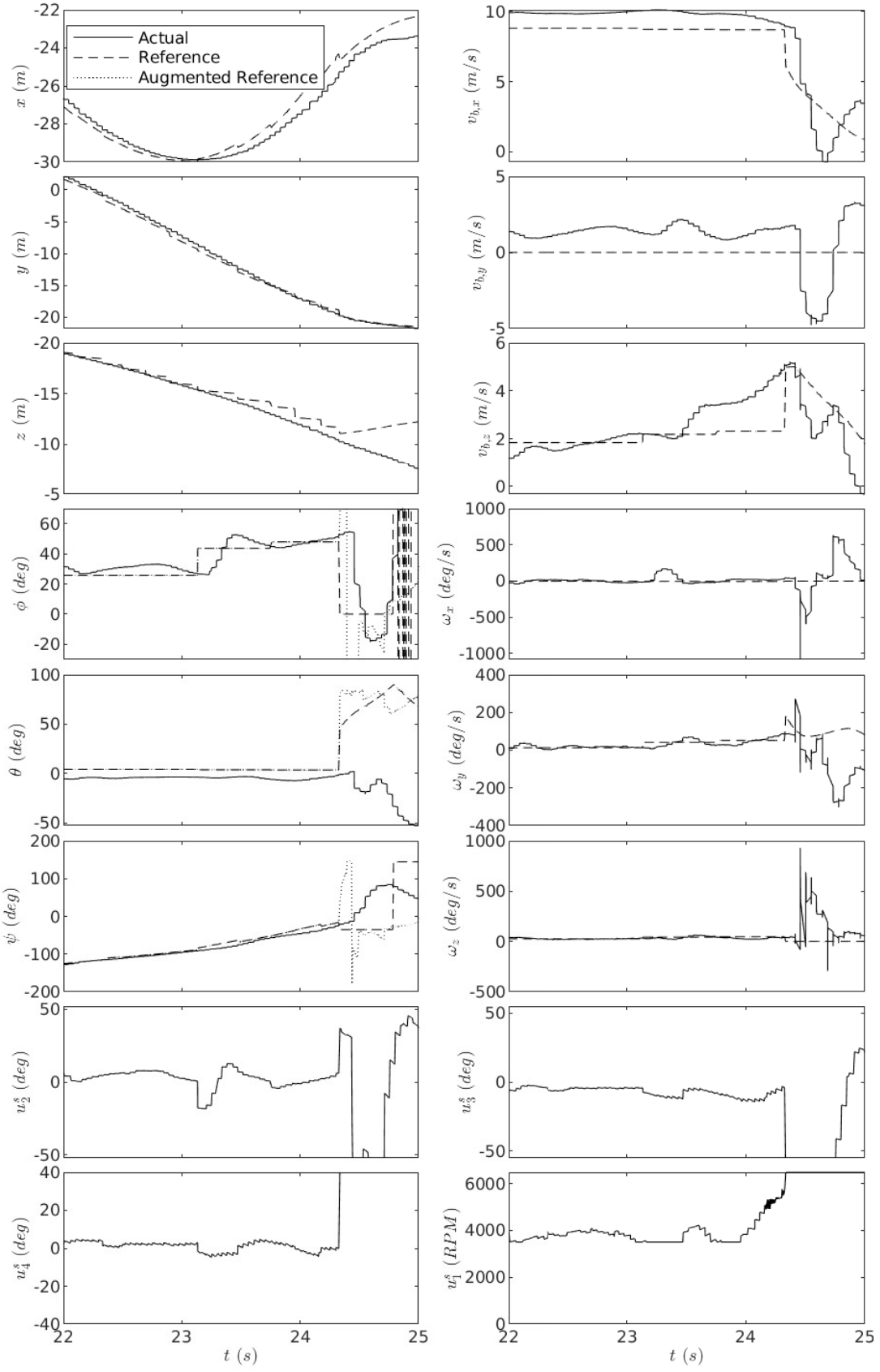


FIGURE 6.33: Collision: state estimates, reference states, and control inputs

6.3.4 Dynamic Obstacles

Outside of the 35 runs presented in the preceding subsections, we briefly considered avoiding moving obstacles. The obstacle avoidance methodology developed in this thesis is designed for static obstacles. If the main purpose of the algorithm was to avoid dynamic obstacles, the intelligent approach would be to factor in the motion of the obstacles into the collision checking process, and not simply treat a dynamic obstacle as static points in a point cloud. However, since our static obstacle avoidance algorithm can be executed quickly, we have succeeded in avoiding dynamic obstacles in certain cases. First off, the dynamic obstacle must be approaching the aircraft from within the FOV. A dynamic obstacle approaching the aircraft from outside of its FOV will never be able to be avoided with any algorithm. Second, there are limitations on the speed of the dynamic obstacle. This speed limitation will be a function of where it enters the aircraft's FOV, and its trajectory relative to the aircraft. Finding this limitation is outside the scope of this thesis, but such a limitation does exist.

We demonstrate dynamic obstacle avoidance with two examples, the first where we throw a football at the aircraft, and the second where we fly another UAV through the aircraft's intended flight path. The avoidance of a football is shown in Fig. 6.34. The football is thrown from the left while the aircraft is initially flying towards it on the right. The trajectories of both the football and aircraft are overlaid on to the image, where the same color circle around the football and aircraft represents the same instant in time. Between the green and yellow times the aircraft has seen the football and decides to turn right to avoid colliding with the football.

The avoidance of an intruder aircraft is shown in Fig. 6.35. The intruder aircraft is initially on the right, and the autonomous aircraft begins on the left. The trajectories of both aircraft are overlaid on to the image, where the same color circle around each aircraft represents the same instant in time. At an instant between the blue and cyan times the autonomous aircraft detects the intruder aircraft, and banks right to avoid it.

FIGURE 6.34: Avoidance of a Football

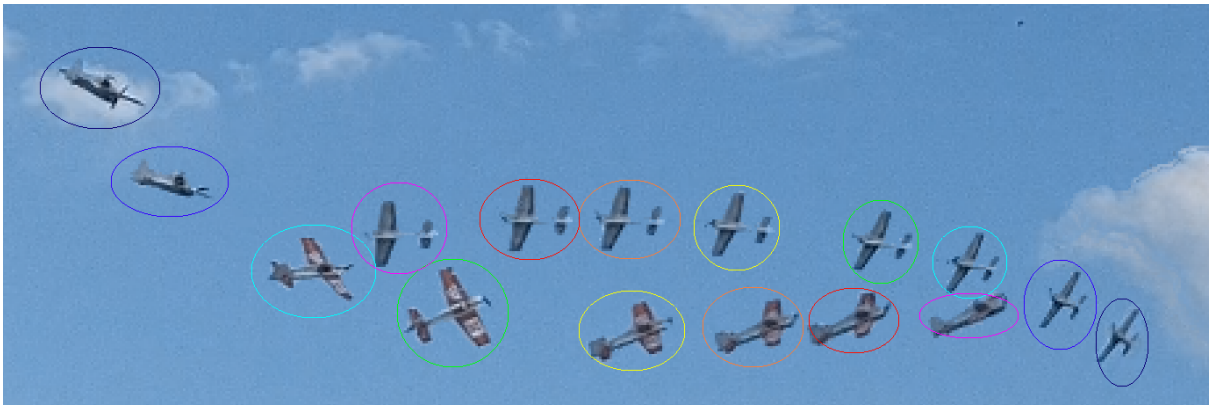


FIGURE 6.35: Avoidance of a fixed-wing aircraft

6.4 Selected Trajectory Distribution

It is useful to analyze the frequency with which particular trim trajectories are being used during the flights discussed in Sec. 6.2 & 6.3. This can provide additional insight into whether the full flight envelope of the aircraft is being used. We display the frequency of selected reference trajectories using bar graphs in Figs. 6.36 - 6.40. The horizontal axes correspond to the reference yaw rate and climb/descent rate, while the vertical axis corresponds to the time in which that reference trajectory was being tracked. We display these plots for the runs where we show the detailed flight data (Fig. 6.5, Figs. 6.25 - 6.27 & Fig. 6.29), which correspond to Figs. 6.36 - 6.38, respectively. We display the selected

reference trajectory distribution for all of the simulations combined in Fig. 6.39, and all of the experiments combined in Fig. 6.40.

Referring to Figs. 6.36 - 6.38, we can see that for an individual run, the reference yaw rates are far from evenly distributed, and tend to concentrate around a few yaw rates. This is likely due to our cost function, which penalizes large changes in yaw rate. Looking at all of the simulations and experiments in Figs. 6.39 & 6.40, we can see the yaw rate selection tends to favor near zero yaw rates, and is distributed more evenly in comparison to a single run. In both simulation and experiment, the aircraft never utilizes the extreme turn rates ($> 80°/s$), as these extreme turn rates would turn the aircraft out of the field-of-view, without enough distance to come to an emergency stop if needed. Using a sensor with a wider horizontal field-of-view would allow using more of the aircraft's flight envelope. We also notice that the selected trajectories in simulation tend to climb ($v_{i,z}^{ref} < 0$), where as in experiment the trajectories are more evenly distributed with respect to climb rate, but the descending trajectories are frequented slightly more. We also see that the positive yaw rate trajectories are frequented more than the negative yaw rate trajectories in experiment, which is likely only caused by the specific environments and initial conditions used during experiment.

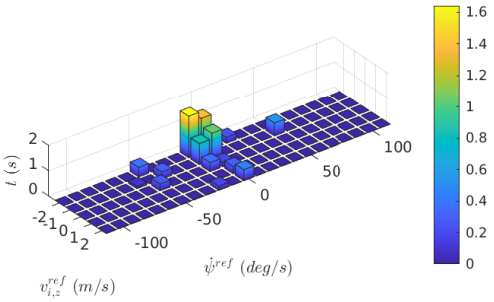


FIGURE 6.36: Trajectory Distribution of Simulation (Environment 1 at $9\frac{m}{s}$, Run 3)

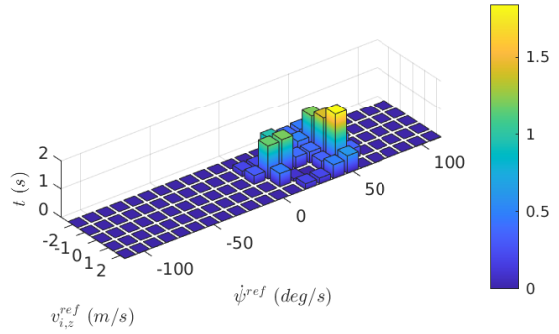


FIGURE 6.37: Trajectory Distribution of Experiment (Run 4)

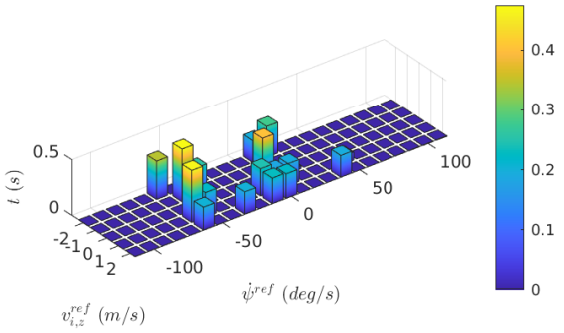


FIGURE 6.38: Trajectory Distribution of Experiment (Run 22)

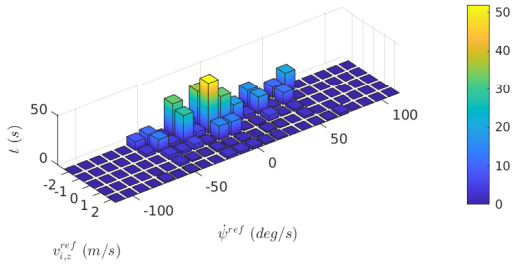


FIGURE 6.39: Trajectory Distribution of All Simulations

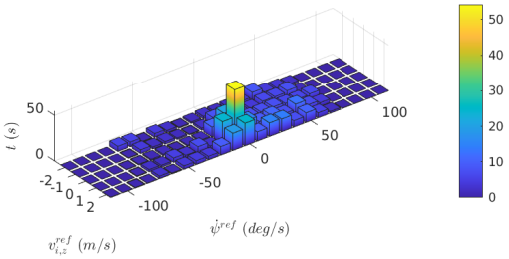


FIGURE 6.40: Trajectory Distribution of All Experiments

6.5 Concluding Remarks

We successfully demonstrate autonomous high-speed flight in cluttered environments in simulation and experiment. When comparing between simulation and experiment, we are able to demonstrate flight in more cluttered environments and with a higher success rate in simulation in comparison to experiment. In our simulation, there is no wind, the aircraft has perfect sensing, and the obstacle avoidance utilizes a trajectory library built with a perfect model. In experiments, there is wind, sensing errors, and modelling discrepancies which all degrade the capabilities of the autonomous aircraft. However, while we don't obtain experimental results on par with our simulation, we do obtain leading-edge experimental results in comparison to other experimental work.

In both simulation and experiment, the hover maneuver is utilized in many situations to avoid a collision. We are not surprised that the hover maneuver was useful during both simulated and actual flight, as the ability to come to a complete stop enables us to theoretically guarantee collision-free flight. We believe utilizing the hover is a significant reason we obtained such successful simulations and experiments. In the work in [78, 79], which lacks a stopping maneuver (i.e. hover), it is quite possible that some of the collisions occurred in their experiments would have been avoided with a stopping maneuver.

Out of the 80 simulations presented in Sec. 6.2, which never had a vision or mechanical failure, the aircraft never collided with an obstacle. Obtaining a perfect record in experiment is much more difficult due to the many uncontrolled factors that can cause a collision. It is noteworthy that out of the four autonomous collisions that did occur, none of the collisions were directly related to the obstacle avoidance or control algorithm. The three collisions related to the depth camera show that while the depth camera does enable autonomous flight, a more expensive autonomous aircraft should have redundant obstacle detection sensors to minimize the risk of a collision. The collision caused by the

mechanical failure is more specific to our test platform, and was more specifically a result of not properly fixing the damage caused by one of the previous three collisions.

Chapter 7

Conclusion

We conclude the thesis by summarizing the key conclusions from our results and making recommendations for future work.

7.1 Conclusions

Model validation for an agile fixed-wing aircraft should not be done in open-loop, but rather with an automatic closed-loop controller or a human pilot, through a comparison of accelerations of the simulated and real platforms. The validation of the model used in this thesis shows a good correspondence at the level of translational accelerations, but significant discrepancies in the angular accelerations.

We have shown it is possible to formulate a single closed-loop controller that can execute a wide range of aerobatic maneuvers with an agile fixed-wing aircraft. Using this controller, we provide a Lyapunov stability analysis for the orientation and translational error dynamics. We have shown it is possible to extend the applicability of this control architecture to a large variety of aerial platforms, including quadrotors, tailsitters, flapping-wing, and tilt-wing aircraft. The extension is applicable to aircraft capable of generating a moment in any direction and a force in a single body-fixed direction. The controllers for all these platforms only differ at the level of control allocation which is vehicle-specific.

We utilize conventional and hardware-in-the-loop simulations to initially validate the agile fixed-wing aircraft controller. The control gains found in the conventional simulation environment needed to be significantly reduced when testing in the HIL, but no further

adjustments were needed in flight tests. Both simulation tools were shown to be valuable for ensuring a smooth transition into flight testing. In all of the testing scenarios, which include conventional simulation, HIL simulation, indoor flights, outdoor flights at 5 $\frac{m}{s}$, and outdoor flights at 9 $\frac{m}{s}$, we demonstrated autonomous extreme aerobatic maneuvers including knife-edge, rolling Harrier, aggressive turnaround, and hovering and its transitions to and from level flight. In every scenario the controller is able to effectively execute each maneuver, and for the most part, the motion is similar in every scenario. The main differences in the aircraft's motion between simulation and flight testing occurred while executing the knife-edge maneuver; and the main differences between the motion in indoor and outdoor flight occurred while hovering. The extended applicability of the control logic is validated through quadrotor simulations and outdoor flight tests. In these flights, autonomous flips are demonstrated using the controller.

We have shown it is possible to formulate an obstacle avoidance algorithm that enables autonomous high-speed flight of an agile fixed-wing UAV in unknown, unstructured environments; all while only relying on on-board computation and sensing; and can theoretically guarantee collision-free flight.

Using the obstacle avoidance algorithm in conjunction with the controller, we first demonstrate autonomous flight through various unknown cluttered environments in simulation. Next, the algorithms are implemented on a test platform, which run in real-time and only utilize on-board sensing and computation. Using the test platform, we demonstrate autonomous flight in unknown, tree-filled environments. During flight testing, the aircraft autonomously flew 4.4 km over 543 s while avoiding trees, and maintained an average speed of 8.1 $\frac{m}{s}$ and a top speed of 14.4 $\frac{m}{s}$. During some of these tests, the position controller was not used to avoid obstacles, demonstrating that the aircraft could still avoid collisions with a loss of GPS signal. During both simulation and flight testing, the ability for the aircraft to hover and come to a stop was shown to be very useful for avoiding collisions in complex scenarios. In simulation, the aircraft never collided with an obstacle, while in experiment, the aircraft incurred four collisions out of 35 trials. Obstacle sensing failures were the main cause for collisions, showing that while stereo cameras do provide a obstacle detection solution in the majority of situations, a robust autonomous system should have redundant obstacle detection sensors.

7.2 Recommendations for Future Work

The research carried out during this thesis raises some potential future research avenues:

- The model validation technique presented in Chapter 2 is used to get a general understanding of the accuracy of the dynamics model. The validation technique could be used to identify weaknesses of the dynamics model. These could then be addressed, and the model validation technique could be used again to assess the modifications. Furthermore, this model validation technique could be used for vehicles other than agile fixed-wing aircraft.
- The control allocation for the agile fixed-wing aircraft presented in Sec. 3.5 uses simple modelling techniques to map the control force and moment to thruster rotational speed and control surface deflection angle. The control surface deflection angle is obtained using a simple slipstream model (momentum theory), *and* assumes this slipstream speed is the same, and uniform over each control surface. The control performance could potentially be improved by using either more sophisticated modelling techniques (such as those in Chapter 2) or an adaptive approach to learn these mappings in-flight.
- The controller presented in Chapter 3 assumes the airspeed is the ground speed (i.e no wind). While the controller is shown to be robust enough to obtain successful results in outdoor conditions with moderate winds, the control system was not tested in extreme winds, when the force generated by the wind approaches the maximum thruster force. An interesting extension to the control logic presented would be to consider these extreme wind scenarios. This would likely require estimating the wind from either a wind sensor or other aircraft state variables.
- The experimental implementation of the controller in Chapter 4 uses an open-loop mapping of the thruster rotational speed and control surface deflections to PWM signals. Sensing the thruster rotational speed and/or the control surface deflections would be useful for model validation, and could be used to incorporate local closed-loop control over the thruster rotational speed and control surface deflection angle.
- All of the maneuvers demonstrated in Chapter 4 are airborne. Investigating takeoff and landing maneuvers would be worthwhile.
- The extension of the control architecture to other platforms in Sec. 3.6 could be validated with more test platforms; both in simulation and experimentally.
- The reactive obstacle avoidance algorithm presented in Chapter 5 should be combined with a SLAM implementation and a global motion planner. This may not be feasible with state-of-the-art processors and algorithms within our current vehicle's payload capacity, but this could be either evaluated in simulation or using a larger platform with a bigger capacity.
- While the (reference) trajectory selection process in Chapter 5 is thoroughly developed, the effects switching reference trajectories could be studied in more detail.

Differences in yaw and position from before the motion plan is computed until the time the reference trajectory is selected can have significant effects on the aircraft motion, as discussed in Sec. 6.3.3.1. This could be addressed by predicting the aircraft motion for the duration of the motion plan computation time. Another potential area for improvement pertains to the discontinuous switching of maneuvers. Currently, the reference attitude, velocity, and angular velocity from one reference trajectory are discontinuously switched to the those of next reference trajectory. Smoothing this transition could improve the performance.

- With the exception of the hover, the obstacle avoidance methodology presented in Chapter 5 uses only trim primitives at a specified speed. When operating at higher speeds, the aircraft's trim primitives have a large minimum turning radius due to thruster saturation. Incorporating additional agile primitives could potentially create turns with smaller turning radii. Another way to address this issue would be to enhance the obstacle avoidance methodology to vary the aircraft speed, and when necessary the aircraft could lower its speed to achieve sharper turns. Furthermore, other agile maneuvers could be incorporated, such as the knife-edge to travel in gaps narrower than the aircraft's wingspan.
- In Chapter 5, 41 final target positions are used somewhat arbitrarily. Increasing the number of final target positions increases the number of potential trajectories for the aircraft, but decreases the computational time. There are potential advantages to increasing or decreasing the number of final target positions. It would be interesting to investigate varying the amount and placement of the final target positions.
- The obstacle avoidance algorithm is developed for static obstacles. Explicitly accounting for the motion of dynamic obstacles would be interesting, specifically in the case of avoiding other unmanned aerial vehicles.
- The simulations in Chapter 6 use a depth camera with a fixed range and field-of-view. Investigating the effect of varying the range and field-of-view of the camera would be worthwhile.

References

[1] Eitan Bulka and Meyer Nahon. Autonomous control of agile fixed-wing UAVs performing aerobatic maneuvers. In *2017 international conference on unmanned aircraft systems (ICUAS)*, pages 104–113. IEEE, 2017.

[2] 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.

[3] Eitan Bulka and Meyer Nahon. A universal controller for unmanned aerial vehicles. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4171–4176. IEEE, 2018.

[4] Eitan Bulka and Meyer Nahon. Automatic control for aerobatic maneuvering of agile fixed-wing UAVs. *Journal of Intelligent & Robotic Systems*, 93(1-2):85–100, 2019.

[5] Walter Jothiraj, Corey Miles, Eitan Bulka, Inna Sharf, and Meyer Nahon. Enabling bidirectional thrust for aggressive and inverted quadrotor flight. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 534–541. IEEE, 2019.

[6] Eitan Bulka and Meyer Nahon. High-speed obstacle-avoidance with agile fixed-wing aircraft. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 971–980. IEEE, 2019.

[7] Eitan Bulka and Meyer Nahon. A unified control strategy for autonomous aerial vehicles. In *Autonomous Robots (Under Review)*, 2020.

[8] Eitan Bulka and Meyer Nahon. Reactive obstacle-avoidance for agile fixed-wing unmanned aerial vehicles. In *Field Robotics (Under Review)*, 2021.

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

[10] JL Verboom, Sjoerd Tijmons, C De Wagter, B Remes, Robert Babuska, and Guido CHE de Croon. Attitude and altitude estimation and control on board a flapping wing micro air vehicle. In *International Conference on Robotics and Automation (ICRA)*, pages 5846–5851. IEEE, 2015.

[11] Airbus. Vahana. https://vahana.aero, 01 2017. (accessed 16/10/2018).

[12] Joshua Levin. *Maneuver Design and Motion Planning for Agile Fixed-Wing UAVs*. PhD thesis, McGill University, Montreal, April 2019.

[13] Zipline. Light, fast, electric, 2020. URL https://flyzipline.com/. [Online; accessed November 25, 2020].

[14] DJI. Mavic 2, 2020. URL https://www.dji.com/ca/mavic-2?site=brandsite&from=nav. [Online; accessed November 25, 2020].

[15] Robert E Roberson. Two decades of spacecraft attitude control. *Journal of Guidance and Control*, 2(1):3–8, 1979.

[16] Bong Wie, H Weiss, and Aristotle Arapostathis. Quaternion feedback regulator for spacecraft eigenaxis rotations. *Journal of Guidance, Control, and Dynamics*, 12(3): 375–380, 1989.

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

[18] V Gavrilets, Bernard Mettler, and E Feron. Human-inspired control logic for automated maneuvering of miniature helicopter. *Journal of Guidance Control and Dynamics*, 27(5):752–759, 2004.

[19] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. An Application of Reinforcement Learning to Aerobatic Helicopter Flight. In P. B. Schlkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1–8. MIT Press, 2007. URL http://papers.nips.cc/paper/3151-an-application-of-reinforcement-learning-to-aerobatic-helicopter-flight.pdf.

[20] 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, April 2012. ISSN 0278-3649. doi: 10.1177/0278364911434236. URL http://dx.doi.org/10.1177/0278364911434236.

[21] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Nonlinear robust tracking control of a quadrotor UAV on SE (3). *Asian Journal of Control*, 15(2):391–408, 2013.

[22] William E Green and Paul Y Oh. Autonomous hovering of a fixed-wing micro air vehicle. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2164–2169. IEEE, 2006.

[23] W. E. Green and P. Y. Oh. A Hybrid MAV for Ingress and Egress of Urban Environments. *IEEE Transactions on Robotics*, 25(2):253–263, April 2009. ISSN 1552-3098. doi: 10.1109/TRO.2009.2014501.

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

[25] Frantisek Michal Sobolic. *Agile flight control techniques for a fixed-wing aircraft.* Thesis, Massachusetts Institute of Technology, 2009. URL http://dspace.mit.edu/handle/1721.1/51640.

[26] Rick Cory and Russ Tedrake. Experiments in fixed-wing UAV perching. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 7256, 2008.

[27] Joseph Moore, Rick Cory, and Russ Tedrake. Robust post-stall perching with a simple fixed-wing glider using LQR-Trees. *Bioinspiration & Biomimetics*, 9(2): 025013, June 2014. ISSN 1748-3190. doi: 10.1088/1748-3182/9/2/025013.

[28] Alexis Lussier Desbiens and Mark R Cutkosky. Landing and perching on vertical surfaces with microspines for small unmanned air vehicles. *Journal of Intelligent and Robotic Systems*, 57(1-4):313, 2010.

[29] Alan T Asbeck, Sangbae Kim, Mark R Cutkosky, William R Provancher, and Michele Lanzetta. Scaling hard vertical surfaces with compliant microspine arrays. *The International Journal of Robotics Research*, 25(12):1165–1179, 2006.

[30] 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, 2011.

[31] Dino Mehanovic, David Rancourt, and Alexis Lussier Desbiens. Fast and efficient aerial climbing of vertical surfaces using fixed-wing UAVs. *IEEE Robotics and Automation Letters*, 4(1):97–104, 2018.

[32] Eric N Johnson, Allen 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.

[33] Robin Ritz and Raffaello D'Andrea. A global controller for flying wing tailsitter vehicles. In *International Conference on Robotics and Automation (ICRA)*, pages 2731–2738. IEEE, 2017.

[34] N.K. Ure and G. Inalhan. Autonomous control of unmanned combat air vehicles: Design of a multimodal control and flight planning framework for agile maneuvering. *IEEE Control Systems*, 32(5):74–95, 2012. ISSN 1066033X. doi: 10.1109/MCS.2012.2205532.

[35] James K. Hall and Timothy W. McLain. Aerobatic maneuvering of miniature air vehicles using attitude trajectories. In *AIAA Guidance, Navigation and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics Inc., August 2008.

[36] Eivind Bøhn, Erlend M Coates, Signe Moe, and Tor Ame Johansen. Deep reinforcement learning attitude control of fixed-wing UAVs using proximal policy optimization. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 523–533. IEEE, 2019.

[37] Shanelle G Clarke and Inseok Hwang. Deep reinforcement learning control for aerobatic maneuvering of agile fixed-wing aircraft. In *AIAA Scitech 2020 Forum*, page 0136, 2020.

[38] Sanghyuk Park. Autonomous aerobatics on commanded path. *Aerospace Science and Technology*, 22(1):64–74, 2012. ISSN 12709638. doi: 10.1016/j.ast.2011.06.007.

[39] A. J. Barry, T. Jenks, A. Majumdar, H. T. Lin, I. G. Ros, A. A. Biewener, and R. Tedrake. Flying between obstacles with an autonomous knife-edge maneuver. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2559–2559, May 2014. doi: 10.1109/ICRA.2014.6907217.

[40] Andrew Barry. Flying between obstacles with an autonomous knife-edge maneuver. Master's thesis, Massachusetts Institute of Technology. Department of Electrical Engineering and Computer Science, 2012.

[41] Minh-Duc Hua, Tarek Hamel, Pascal Morin, and Claude Samson. Control of a class of thrust-propelled underactuated vehicles and application to a VTOL drone.

In *International Conference on Robotics and Automation (ICRA)*, pages 972–978. IEEE, 2009.

[42] Daniele Pucci, Tarek Hamel, Pascal Morin, and Claude Samson. Nonlinear feedback control of axisymmetric aerial vehicles. *Automatica*, 53:72–78, 2015.

[43] Minh-Duc Hua, Daniele Pucci, Tarek Hamel, Pascal Morin, and Claude Samson. A novel approach to the automatic control of scale model airplanes. In *53rd Annual Conference on Decision and Control (CDC)*, pages 805–812. IEEE, 2014.

[44] 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, 2008.

[45] Markus Ryll, John Ware, John Carter, and Nick Roy. Efficient trajectory planning for high speed flight in unknown environments. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 732–738. IEEE, 2019.

[46] 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.

[47] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[48] Steven M. LaValle and James J. Kuffner. Randomized Kinodynamic Planning. *The International Journal of Robotics Research*, 20(5):378–400, May 2001. ISSN 0278-3649. doi: 10.1177/02783640122067453. URL http://dx.doi.org/10.1177/02783640122067453.

[49] Emre Koyuncu, N. Kemal Ure, and Gokhan Inalhan. A Probabilistic Algorithm for Mode Based Motion Planning of Agile Unmanned Air Vehicles in Complex Environments. *IFAC Proceedings Volumes*, 41(2):2661–2668, January 2008. ISSN 1474-6670. doi: 10.3182/20080706-5-KR-1001.00448. URL http://www.sciencedirect.com/science/article/pii/S1474667016393533.

[50] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, August 1996. ISSN 1042-296X. doi: 10.1109/70.508439.

[51] Emre Koyuncu, N. Kemal Ure, and Gokhan Inalhan. Integration of Path/Maneuver Planning in Complex Environments for Agile Maneuvering UCAVs. *Journal of Intelligent and Robotic Systems*, 57(1-4):143, January 2010. ISSN 0921-0296, 1573-0409. doi: 10.1007/s10846-009-9367-1. URL https://link.springer.com/article/10.1007/s10846-009-9367-1.

[52] M. Hwangbo, J. Kuffner, and T. Kanade. Efficient Two-phase 3d Motion Planning for Small Fixed-wing UAVs. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1035–1041, April 2007. doi: 10.1109/ROBOT.2007.363121.

[53] Myung Hwangbo and Takeo Kanade. Maneuver-based autonomous navigation of a small fixed-wing UAV. In *2013 IEEE International Conference on Robotics and Automation, ICRA 2013, May 6, 2013 - May 10, 2013*, Proceedings - IEEE International Conference on Robotics and Automation, pages 3961–3968. Institute of Electrical and Electronics Engineers Inc., 2013. doi: 10.1109/ICRA.2013.6631135.

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

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

[56] Joshua M. Levin, Meyer Nahon, and Aditya A. Paranjape. Real-time motion planning with a fixed-wing uav using an agile maneuver space. *Autonomous Robots*, May 2019. ISSN 1573-7527. doi: 10.1007/s10514-019-09863-2. URL https://doi.org/10.1007/s10514-019-09863-2.

[57] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 723–730. IEEE, 2011.

[58] 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, June 2015. ISSN 0278-3649. doi: 10.1177/0278364914558129. URL http://dx.doi.org/10.1177/0278364914558129.

[59] A. Bry, A. Bachrach, and N. Roy. State estimation for aggressive flight in GPS-denied environments using onboard sensing. In *2012 IEEE International Conference on Robotics and Automation*, pages 1–8, May 2012. doi: 10.1109/ICRA.2012.6225295.

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

[61] Tom Schouwenaars, Jonathan How, and Eric Feron. Receding horizon path planning with implicit safety guarantees. In *Proceedings of the 2004 American control conference*, volume 6, pages 5576–5581. IEEE, 2004.

[62] Max Basescu and Joseph Moore. Direct NMPC for post-stall motion planning with fixed-wing UAVs. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9592–9598. IEEE, 2020.

[63] 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, Part B (Cybernetics)*, 33(6):898–912, 2003.

[64] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, June 1991. ISSN 1042-296X. doi: 10.1109/70.88137.

[65] Lin Zhao. 3D Obstacle Avoidance for Unmanned Autonomous System (UAS). Master's thesis, University of Las Vegas, Las Vegas, USA, 2015.

[66] A. J. Barry, A. Majumdar, and R. Tedrake. Safety verification of reactive controllers for UAV flight in cluttered environments using barrier certificates. In *2012 IEEE International Conference on Robotics and Automation*, pages 484–490, May 2012. doi: 10.1109/ICRA.2012.6225351.

[67] Yucong Lin and Srikanth Saripalli. Sense and avoid for unmanned aerial vehicles using ads-b. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6402–6407. IEEE, 2015.

[68] Yucong Lin and Srikanth Saripalli. Sampling-based path planning for UAV collision avoidance. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):3179–3192, 2017.

[69] Ruan Van Breda. Vector field histogram star obstacle avoidance system for multicopters. Master's thesis, Stellenbosch University, Stellenbosch, South Africa, 2016.

[70] Sikang Liu, Michael Watterson, Sarah Tang, and Vijay Kumar. High speed navigation for quadrotors with limited onboard sensing. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1484–1491. IEEE, 2016.

[71] Brett Thomas Lopez and Jonathan P How. Aggressive 3-d collision avoidance for high-speed navigation. In *ICRA*, pages 5759–5765, 2017.

[72] Brett T Lopez and Jonathan P How. Aggressive collision avoidance with limited field-of-view sensing. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1358–1365. IEEE, 2017.

[73] Jesus Tordesillas, Brett T Lopez, John Carter, John Ware, and Jonathan P How. Real-time planning with multi-fidelity models for agile flights in unknown environments. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 725–731. IEEE, 2019.

[74] Jesus Tordesillas, Brett T Lopez, Michael Everett, and Jonathan P How. Faster: Fast and safe trajectory planner for flights in unknown environments. *arXiv preprint arXiv:2001.04420*, 2020.

[75] Sjoerd Tijmons, Guido CHE de Croon, Bart DW Remes, Christophe De Wagter, and Max Mulder. Obstacle avoidance strategy using onboard stereo vision on a flapping wing mav. *IEEE Transactions on Robotics*, 33(4):858–874, 2017.

[76] Antoine Beyeler, Jean-Christophe Zufferey, and Dario Floreano. Vision-based control of near-obstacle flight. *Autonomous robots*, 27(3):201–219, 2009.

[77] William E Green and Paul Y Oh. Optic-flow-based collision avoidance. *IEEE Robotics & Automation Magazine*, 15(1):96–103, 2008.

[78] 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, 2018.

[79] Andrew Barry. *High-Speed Autonomous Obstacle Avoidance with Pushbroom Stereo*. PhD thesis, Massachusetts Institute of Technology. Department of Electrical Engineering and Computer Science, 2016.

[80] W. Khan and M. Nahon. Toward an Accurate Physics-Based UAV Thruster Model. *IEEE/ASME Transactions on Mechatronics*, 18(4):1269–1279, August 2013. ISSN 1083-4435.

[81] Waqas Khan and Meyer Nahon. Development and Validation of a Propeller Slipstream Model for Unmanned Aerial Vehicles. *Journal of Aircraft*, 52(6):1985–1994, 2015. ISSN 0021-8669.

[82] W. Khan and M. Nahon. Modeling dynamics of agile fixed-wing UAVs for real-time applications. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1303–1312, June 2016.

[83] W. Khan and M. Nahon. Real-time modeling of agile fixed-wing UAV aerodynamics. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1188–1195, June 2015.

[84] X-Plane. How x-plane works. http://www.x-plane.com/desktop/how-x-plane-works/, 2018. (Accessed: 1/15/2018).

[85] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[86] Dongwon Jung and Panagiotis Tsiotras. Modeling and hardware-in-the-loop simulation for a small unmanned aerial vehicle. In *AIAA Infotech@ Aerospace 2007 Conference and Exhibit*, page 2768, 2007.

[87] Hou In Leong, Rylan Jager, Shahriar Keshmiri, and Richard Colgren. Development of a pilot training platform for UAVs using a 6dof nonlinear model with flight test validation. In *AIAA Modeling and Simulation Technologies Conference and Exhibit*, page 6368, 2008.

[88] Matthew Rose, Hovig Yaralian, Joseph Wagster, and Subodh Bhandari. Development and validation of flight dynamics model of a UAV airplane. In *Infotech@ Aerospace 2012*, page 2592. 2012.

[89] Miguel Angel García Terán, Ernesto Olguín-Díaz, Angel Flores-Abad, and Manuel Nandayapa. Experimental validation of an aerodynamic sectional modeling approach in fixed-wing unmanned aerial vehicles. *IEEE Access*, 6:74190–74203, 2018.

[90] DR Wong, Q Ou, M Sinclair, YJ Li, XQ Chen, and A Marburg. Unmanned aerial vehicle flight model validation using on-board sensing and instrumentation. In *2008 15th International Conference on Mechatronics and Machine Vision in Practice*, pages 109–114. IEEE, 2008.

[91] 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, 1992.

[92] Waqas Khan and Meyer Nahon. A propeller model for general forward flight conditions. *International Journal of Intelligent Unmanned Systems*, 3(2/3):72–92, 2015.

[93] B. Etkin. *Dynamics of flight: stability and control*. John Wiley & Sons Australia, Limited, 1982. ISBN 978-0-471-08936-0. URL https://books.google.ca/books?id=4n5TAAAAMAAJ.

[94] Barnes McCormick. *Aerodynamics, Aeronautics, and Flight Mechanics*. Wiley, 2 edition, 1995. ISBN ISBN: 978-0-471-57506-1.

[95] Bambang Riyanto Trilaksono, Syahron Hasbi Nasution, Eko Budi Purwanto, et al. Design and implementation of hardware-in-the-loop-simulation for UAV using pid control method. In *2013 3rd International Conference on Instrumentation, Communications, Information Technology and Biomedical Engineering (ICICI-BME)*, pages 124–130. IEEE, 2013.

[96] Guowei Cai, Ben M Chen, Tong H Lee, and Miaobo Dong. Design and implementation of a hardware-in-the-loop simulation system for small-scale UAV helicopters. In *2008 IEEE International Conference on Automation and Logistics*, pages 29–34. IEEE, 2008.

[97] *Pixhawk Mini Quick Start Guide*. 3D Robotics.

[98] Environment and Climate Change Canada. Hourly data report. URL http://climate.weather.gc.ca.

[99] Romain Chiappinelli and Meyer Nahon. Modeling and control of a tailsitter UAV. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 400–409. IEEE, 2018.

[100] Christian Patience and Meyer Nahon. Control of a passively-coupled hybrid aircraft. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020.

[101] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles. *IEEE Robotics and Automation magazine*, 20(32), 2012.

[102] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham, 2016. ISBN 978-3-319-26054-9. doi: 10.1007/978-3-319-26054-9_23.

[103] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[104] 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 Trans. Math. Softw.*, 41(1), October 2014. ISSN 0098-3500. doi: 10.1145/2558904. URL https://doi.org/10.1145/2558904.

[105] *Intel RealSense Product Family D400 Series Datasheet*. Intel, 6 2020. Rev. 9.

[106] Daniel Pohl, Sergey Dorodnicov, and Markus Achtelik. Depth map improvements for stereo-based depth cameras on drones. In *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 341–348. IEEE, 2019.

[107] Adrian Battiston, Inna Sharf, and Meyer Nahon. Attitude estimation for normal flight and collision recovery of a quadrotor UAV. In *Unmanned Aircraft Systems (ICUAS), 2017 International Conference on*, pages 840–849. IEEE, 2017.

[108] Guowei Cai, Ben M. Chen, and Tong Heng Lee. *Unmanned Rotorcraft Systems*. Springer, New York, 2011 edition edition, June 2011. ISBN 978-0-85729-634-4.

Appendix A

Hardware-in-the-Loop Simulation

We use a Pixhawk flight controller coupled with the PX4 flight stack to perform HIL testing. The PX4 flight stack has a built in HIL simulation, where the X-Plane physics engine is used to represent the aircraft dynamics. In this setup, the Pixhawk sends artificial actuator commands using the MAVlink protocol, through a USB connection, to the QgroundControl (QGC) user interface software. QGC sends these commands to X-Plane using the UDP protocol. The sensor feedback is sent from X-plane to QGC as a UDP message, which is then sent from QGC to the Pixhawk as a MAVlink message. This communication is depicted in Fig. A.1.

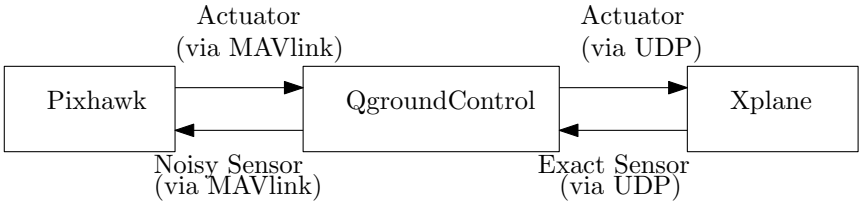


FIGURE A.1: Default HIL Block Diagram

We believe that our flight dynamics model is more accurate than X-plane, particularly for small aerobatic aircraft. For example, we use a detailed model for the propeller slipstream effect, as opposed to X-Plane which uses a simple momentum theory model [84]. We therefore coupled our in-house Simulink simulator to the Pixhawk to produce a more realistic HIL simulator. Not only is this approach useful for our purposes, but it could be useful for others using Pixhawk (e.g., for quadrotors, VTOL, and fixed-wing) wanting to perform HIL validation using their own Simulink dynamics model. Something similar has been done in [95], but few details are provided. In the following, we provide additional detail on this process. With an understanding the format of the UDP messages that allow communication between QGC and X-Plane, the X-plane physics engine can be

replaced by a Simulink model. Note that, while the X-Plane model is replaced, we retain X-Plane as a graphical display of the aircraft in flight. The communication is depicted by a block diagram in Fig. A.2.

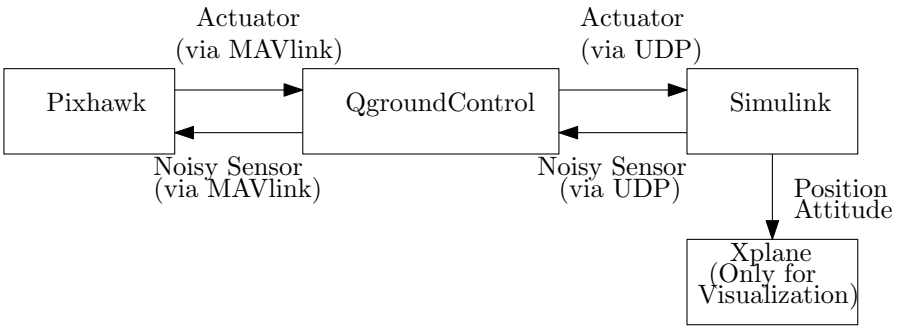


FIGURE A.2: Custom Block Diagram

A.1 UDP Data Format

The artificial sensor measurements and actuations are sent as UDP messages to and from Qgrouncontrol (QGC) in binary format. The structure of these messages can be deduced from the Qgroundcontrol source code in the file /src/comm/QGCXPlaneLink.cc.

A.1.1 Actuator Commands

QGC sends an artificial actuator command as a 41 byte message containing 5 characters (5 bytes), 1 index (4 bytes), and 8 floats (32 bytes). This message can be received in Simulink using the "UDP Receive Binary" block with the IP address set to 127.0.0.1 and the same port as that used in the "HIL config" widget in QGC. The output port width is constant and should be set to 41. The sample time should be that of the simulation.

The signal coming out of the "UDP Receive Binary" block should be sent to an "Unpack" block. The parameters for this block should be the following: Output port dimensions should be 14 1-dimensional vectors, i.e. {[1],[1],...[1]}. The output port data types should have the first 5 be uint8, followed by 1 uint32, followed by 8 singles.

Now, using the index one can see the data being sent from the Pixhawk. With the default QGC, when the index is 8, the first single is the aileron, the second single is the negative elevator, and the third single is the rudder. When the index is 25, all the singles are the throttle. It should be noted that this can be easily changed considering QGC is open

source. It should also be noted that different indices would be used for a quadrotor or VTOL aircraft.

A.1.2 Sensor Feedback

QGC receives measurements from n sensors from Simulink in the form of a $5 + 36n$ byte message containing 5 characters (5 bytes), and then an integer index (4 bytes) and 8 floats (32 bytes) for each sensor. The 5 characters are the encryption key which lets QGC know the rest of the incoming data are sensor measurements. This key must have the first 4 characters be 'D','A','T','A', and the last character can be anything.

This message can be sent from Simulink using the "Pack" block to convert the artificial sensor measurements to binary. The parameters to the "Pack" block are the input port data types, which should be 5 uint8 followed by n sequences of 1 uint32 and 8 singles. The first 4 inputs to the "Pack" block should be constant blocks with data type uint8, with the following numbers: 68,65,84,65 (translates to 'D','A','T','A'). The fifth block should be a constant block with uint8 data type, but the number does not matter. The indices for each sensor, and what each float refers to can be found in /src/comm/QGCXPlaneLink.cc. For example, the index 16 corresponds to gyroscope, where the first float is the pitch speed, the second float is the roll speed, and the third float is the yaw speed. The output of the "Pack" block should be sent to the "UDP Send Binary" where the IP address should be set to 127.0.0.2 and and port to 49005. The sample time should the same as the simulation.

Using the default QGC, the artificial sensor measurements given from Simulink will not be exactly what is sent to the Pixhawk. QGC adds noise to the sensors in the following file: src/uas/UAS.cc. We have found it simpler to add noise to the measurements in Simulink, so we have set the variance of each sensor in the src/uas/UAS.cc file to zero, so that no additional noise is added.

A.2 Executing the HIL

In order to execute the HIL, the Pixhawk must be configured in the HIL X-Plane mode. Once connected to QGC via USB cable, open the HIL Config widget. Select X-Plane 10, with IP address of 127.0.0.1 with port corresponding to the UDP receive block in Simulink. Click connect, and then start the Simulink model. The widget should show

"Connecting to X-plane at XX Hz", where XX should correspond to the rate the Simulink model is being executed at. It should be noted that the Simulink model should contain the "Real-Time Sync" block. This will also force the solver to be fixed-step, where the step size should be the same as the UDP send/receive blocks.

A.3 Sensor Measurement Generation

In pure simulation, the controller is directly fed the position and orientation of the aircraft. However, in the HIL simulation, the hardware receives artificial sensor measurements which are generated based on the simulated aircraft motion and the sensors' noise characteristics. Generating these artificial sensor measurements can be done with the following models for the accelerometer, gyroscope, magnetometer, barometer and GPS:

$$\boldsymbol{\zeta}_{acc} = \dot{\mathbf{v}}_b + \boldsymbol{\omega}_b \times \mathbf{v}_b + \mathbf{C}_{bi}\mathbf{g}_i + \boldsymbol{\eta}_{acc} \tag{A.1}$$

$$\boldsymbol{\zeta}_{gyr} = \boldsymbol{\omega}_b + \boldsymbol{\eta}_{gyr} \tag{A.2}$$

$$\boldsymbol{\zeta}_{mag} = \mathbf{C}_{bi}\boldsymbol{\mu} + \boldsymbol{\eta}_{mag} \tag{A.3}$$

$$\zeta_{bar} = -z + \eta_{bar} \tag{A.4}$$

$$\boldsymbol{\zeta}_{gps} = \boldsymbol{\chi}(x, y, z) + \boldsymbol{\eta}_{gps} \tag{A.5}$$

where $\boldsymbol{\zeta}_s$ is the sensor measurement for $s \in \{acc, gyr, mag, bar, gps\}$ and the sensor noise is modeled as a Gaussian: $\boldsymbol{\eta}_s = \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_s)$ where $\boldsymbol{\sigma}_s$ is the standard deviation. The linear and angular velocities expressed in the body frame are denoted by $\mathbf{v_b}$ and $\boldsymbol{\omega}_b$ respectively. The position coordinates are denoted by x, y, z, and the gravity vector, $\mathbf{g}_i$, are both expressed in an NED coordinate frame. The rotation matrix from the inertial to body frame is denoted by $\mathbf{C}_{bi}$, and the earth's magnetic field, $\boldsymbol{\mu} = [.302\ 0\ .950]^T$, corresponds to the magnetic field vector close to Montreal, Canada [107].

Sensor noise arises from a variety of sources, and thus it is most accurate to characterize the noise using aircraft flight data. During straight and level flight, the aircraft orientation is essentially constant, and we can assume all high frequency IMU measurements can be attributed to sensor noise. Therefore, we apply a high-pass filter to level flight IMU data, and consider the variance of the resulting signals to represent the sensor noise, as shown in Table A.1. Generating the GPS measurement is a little more complicated, requiring a transformation from North-East-Down to Geodetic convention, denoted by $\boldsymbol{\chi}(x, y, z)$ and shown in [108]. The GPS noise is approximated by collecting GPS measurements for

TABLE A.1: Sensor Noise Properties

Sensor	Value	Unit
$\boldsymbol{\sigma}_{acc}$	$[0.4975,\ 1.6951,\ 1.4192]^T$	$\frac{m}{s^2}$
$\boldsymbol{\sigma}_{gyr}$	$[0.1357,\ 0.0764,\ 0.0273]^T$	$\frac{rad}{s}$
$\boldsymbol{\sigma}_{mag}$	$[0.0203,\ 0.0162,\ 0.0294]^T$	G
σ_{bar}	.1442	m
$\boldsymbol{\sigma}_{gps_{pos}}$	$[7.20 \times 10^{-6},\ 1.75 \times 10^{-5},\ 1.78]^T$	$[\text{deg},\ \text{deg},\ m]^T$
$\boldsymbol{\sigma}_{gps_{vel}}$	$[0.060,\ 0.077,\ 0.1581]^T$	$\frac{m}{s}$

five minutes while the aircraft remains stationary, and evaluating the variance of these measurements.