A Guidance and Control System for VTOL UAVs operating in Contested Environments

Paul E. Binder

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Aerospace Engineering

Andrea L'Afflitto, Chair

Craig A. Woolsey

Mathieu Joerger

February 16, 2024

Blacksburg, Virginia

Keywords: OCTree, Path Planning, Autonomous Guidance, Trajectory Planning, MPC, Collision Avoidance, Model Reference Adaptive Control, Rotor Characterization, Quadrotor Bi-Plane, Static UAV Testing

A Guidance and Control System for VTOL UAVs operating in Contested Environments

Paul E. Binder

(ABSTRACT)

This thesis presents the initial components of an integrated guidance, navigation, and control system for vertical take-off and landing (VTOL) autonomous unmanned aerial vehicles (UAVs) such that they may map complex environments that may be hostile. The first part of this thesis presents an autonomous guidance system. For goal selection, the map is partitioned around the presence of obstacles and whether that area has been explored. To perform this partitioning, the *Octree algorithm* is implemented. In this thesis, we test this algorithm to find a parameter set that optimizes this algorithm. Having selected goal points, we perform a comparison of the $LPA*$ and $A*$ path planning algorithms with a custom heuristic that enables reckless or tactical maneuvers as the UAV maps the environment. For trajectory planning, the *fMPC* algorithm is applied to the feedback-linearized equations of motion of a quadcopter. For collision avoidance, standalone versions of 4 different constraint generation algorithms are evaluated to compare their resulting computation times, accuracy, and computed volume on a voxel map that simulates a 2-story house along with fixed paths that vary in length at fixed intervals as basis of tests. The second part of this thesis presents the theory of Model Reference Adaptive Control(MRAC) along with augmentation for output signal tracking and switched-dynamic systems. We then detail the development of longitudinal and lateral controllers a Quad-Rotor Tailsitter(QRBP) style UAV. In order to successfully implement the proposed controller on the QRBP, significant effort was placed upon physical design and testing apparatus.

A Guidance and Control System for VTOL UAVs operating in Contested Environments

Paul E. Binder

(GENERAL AUDIENCE ABSTRACT)

For an autonomously operated, Unmanned Aerial Vehicle (UAV), to operate, it requires a guidance system to determine where and how to go, and a control system to effectively actuate the guidance system's commands. In this thesis, we detail the characterization and optimization of the algorithms comprising the guidance system. We then delve into the theory of MRAC and apply it toward a control system for a QRBP. We then detail additional tools developed to support the testing of the QRBP.

Dedication

In memory of the victims of October 7th terrorist attack. May their memories forever be a blessing.

Acknowledgments

First and foremost, thank you Dr. L'Afflitto for providing me with support, guidance, and incredible opportunities during the course of my time at Virginia Tech. I truly appreciate the time and resources you invested in me, the willingness to stay late to help us work through the day's problems, and the support for unconventional approaches. I would like to thank Dr. Woolsey and Dr. Joerger for taking the time to evaluate my research as members of my committee and for each extending help at points where I was able to benefit from the expertise that they and their lab have accrued. I would like to thank my parents for their continued support in every way possible. I would like to thank all of my friends at Virgina Tech for being their for me and for their friendship. I would like to thank my former labmate Dr. Julius Marshall for the opportunity to be a part of his research and for teaching me invaluable lessons about research, conducting experiments and UAVs in general. Finally I would like to thank my labmates Mattia Gramuglia, Giri Mugundan Kumar, Jyortimoy Mukherjee, and Alex Kim for their friendship, spirited debates, ideas, and help.

This work was partly supported by DARPA under Grant no. D18AP00069, NSF through Grant no. 2137159, and the US Department of the Navy under Grant no. N004212110004

thesis

Contents

List of Figures xii

List of Tables xx

1 Introduction 1

1.1 Motivation and Scope 1

1.2 Thesis Outline 4

1.3 Literature Review 6

1.3.1 Autonomous Guidance for UAVs in Contested Environments 6

1.3.2 Control Algorithms for Quadrotor Bi-Planes 8

1.3.3 Refinements to the Experimental Process of Developing the Quadrotor Bi-Plane 8

2 A Tactical Guidance System of Autonomous UAVs 10

2.1 Overview of the Proposed System 10

2.2 Octree Algorithm for Map Partitioning 12

2.2.1 A Brief Overview 13

2.2.2 Theory 13

2.2.3 Notation 14

2.2.4 Octree Performance Test and Results 15
2.3 Comparison of Path Planning Algorithms 18
2.3.1 A Brief Overview 19
2.3.2 Notation 20
2.3.3 Theoretical Formulation 20
2.3.4 Path-Planning Performance Tests and Results 24
2.4 Trajectory Planning 27
2.4.1 Notation 28
2.4.2 Cost Function Definition 29
2.4.3 Equations of Motion 32
2.4.4 Boundary Conditions 34
2.4.5 Collision Avoidance Constraints 36
2.5 Constraint Generation 37
2.5.1 A Brief Overview 38
2.5.2 Theoretical Formulation 39
2.5.3 Constraint Generation Performance Tests and Results 48
2.6 Numerical Simulations Results 52
2.7 Conclusion 54

3 An Adaptive Control System for Quad-Rotor Biplanes 56

3.1 Introduction . . . 56
3.2 Fundamentals of Model Reference Adaptive Control . . . 60
3.3 Model Reference Adaptive Control with Output Signal Tracking . . . 63
3.4 Switched MRAC . . . 65
3.5 Implementation . . . 68
3.5.1 Longitudinal Control . . . 69
3.5.2 Lateral Control . . . 77
3.6 Conclusion . . . 85

4 Refinement of Experimental Process **86**

4.1 Introduction . . . 86
4.2 The Thrust Stand . . . 87
4.2.1 A Brief Overview . . . 88
4.2.2 Process . . . 92
4.2.3 Results . . . 98
4.2.4 Conclusion . . . 99
4.3 Static Fixture Experimentation . . . 100
4.3.1 A Brief Overview . . . 101
4.3.2 Isolated Controller Testing Stand . . . 101
4.3.3 Inverted Pendulum Test Stand . . . 103

4.3.4 Results . . . 105
4.3.5 Conclusion . . . 107
4.4 Vehicle Design . . . 108
4.4.1 Brief Overview . . . 109
4.4.2 Physical Design . . . 110
4.4.3 Conclusion . . . 111
4.5 Conclusion . . . 111

5 Preliminary Flight Tests **113**
5.1 Brief Overview . . . 113
5.2 Flight Tests . . . 116
5.3 Conclusion . . . 117

6 Conclusion **119**
6.1 Summary of Results . . . 119
6.2 Recommendations for Future Work . . . 121

Bibliography **125**

Appendices **131**

Appendix A First Appendix **132**
A.1 Section one . . . 132

Appendix B Third Appendix **134**

List of Figures

1.1 The Advanced Control Systems Lab's current configuration of a QRBP. . . . 2

1.2 The CRC-20 QRBP, a design inspiration for the ACSL's QRBP implementation. 3

2.1 Computational times of each possible configuration of the sets (μ_1, μ_2) and (μ_3, μ_4) 16

2.2 Computational times of Algorithm 2.2.1 as a function of the set (μ_1, μ_2) over the averaged results of varying (μ_3, μ_4) where both μ_1 and μ_2 are increasing. 17

2.3 Computational times of Algorithm 2.2.1 as a function of the set (μ_1, μ_2) over the averaged results of varying (μ_3, μ_4) where μ_1 increases and μ_2 decreases . 18

2.4 Computational times of Algorithm 2.2.1 as a function of the set (μ_3, μ_4) over the averaged results of varying (μ_1, μ_2) 19

2.5 Computational time data of the A* Algorithm with a reckless parameter set. The average computation time across all obstacle sets is 62.1114ms with a standard deviation of 0.3421ms. 25

2.6 Computational time data of the LPA* Algorithm with a reckless parameter set. The average computation time across all obstacle sets is 11.4467ms with a standard deviation of 17.3388ms. 26

2.7 Computational time data of the A* Algorithm with a tactical parameter set. The average computation time across all obstacle sets is 80.9330ms with a standard deviation of 0.3715ms. 27

2.8 Computational time data of the LPA* Algorithm with a tactical parameter set. The average computation time across all obstacle sets is 15.4479ms with a standard deviation of 26.8677ms. 28

2.9 This example provides a 2-D example of the MDCA algorithm. Steps are presented left-to-right, top-to-bottom. Step 1: An obstacle point nearby the path point is identified. Step 2: The ellipsoid, tangent to the previously identified obstacle point, is stretched such that it is tangent to other nearby obstacles point. Step 3: $N = 7$ points are randomly sampled from the surface of the ellipsoid. Step 4: The final constraint set is calculated as the planes tangent to the randomly sampled points and the surface of the ellipsoid. . . 40

2.10 A 2-D representation of the Bubble-Bath algorithm as used to generate affine constraint sets from the set of occupied points $\mathcal{O}$, originally presented in [1]. Steps are presented left-to-right. Step 1: the parent ellipsoid, which contains the UAV and excludes all obstacle points, is created. Step 2: the children ellipsoids are generated such that they are centered upon the boundary of the parent ellipsoid. Step 3: a subset of $\mathcal{O}$ is computed and denoted $\mathcal{O}_{c,i,k}$ such that these occupied points exist within a user-defined distance from the boundary of the volume captured by the parent and children ellipsoids. Step 4: the affine set is computed as the subset of the state-space that contains the UAV and excludes the entirety of $\mathcal{O}_{c,i,k}$. 44

2.11 A 2-D example of the SFC algorithm adapted from [2]. Steps are presented left-to-right, top-to-bottom. Step 1: An initial sphere, free of obstacles and located at the midpoint of L_I, is expanded until it intersects an occupied point. Step 2: The resulting ellipsoid is shrunk as occupied points closer to the path are found. Step 3: An ellipsoid in contact with the occupied point nearest the path is determined. Step 4: A hyperplane is taken tangent to the ellipsoid and occupied point nearest the path. Step 5: Additional hyperplanes are taken as the ellipsoid is stretch to intersect other obstacle points. Step 6: The resulting polyhedron is then scaled down to accommodate the width of the vehicle. 46

2.12 This 2-D example of how the IRIS algorithm operates is adapted from [3]. Steps are presented left-to-right, top-to-bottom. Step 1: An initial sphere, free of obstacles and located at a point along a path is formed. Step 2: A set of linear constraints(shown in red) are taken tangent to the initial sphere and the surrounding obstacles. Step 3: The ellipsoid is stretched such that it intersects the initial linear constraint set and then continues to expand until it intersects the adjacent constraint set. In this step, the final ellipsoid is found. Step 4: The final constraint set is taken tangent to the local obstacle set and the final ellipsoid. 47

2.13 Wireframe diagram of the voxelmap used for testing. 48

2.14 Paths used for testing. The left image shows the first floor and the right image shows the second floor. The numbers present on the plot signify the length of each path. 49

2.15 The 1^{st} order linear regression relation of path size and computation time. . 50

2.16 Average computation time of each path tested by the four algorithms. 51

2.17 Included here is the data signifying the standard deviation of volume generated by each constraint plane set and the number of unbounded constraint sets per path. 52

2.18 Volume information from the collected data sets. The data presented here is the average volume of constraints sets along each path for each algorithm, and the maximum and minimum volume of a constraint set. The red line signifies the median value of the data collected. 53

2.19 A numerical simulation of the proposed trajectory planner with user-defined parameters set to induce reckless guidance and a systematic goal selection behavior. The UAV travels at an average distance of 3.92m from the obstacle set with a standard deviation of 2.78m. The flight time is 306.52s, during which, the UAV traverses 129.57m while exploring voxels at a rate of 531.58 $\frac{voxels}{s}$. 54

2.20 A numerical simulation of the proposed trajectory planner with user-defined parameters set to induce tactical guidance and greedy goal selection behavior. The UAV travels at an average distance of 1.02m from the obstacle set with a standard deviation of 0.94m. The flight time is 463.65s, during which, the UAV traverses 221.85m while exploring voxels at a rate of 445.87 $\frac{voxels}{s}$. . . 55

3.1 The Quad-Rotor Biplane with Vicon-Motion Capture reflectors as used with indoor flight tests. 58

3.2 A diagram of the Quad-Rotor Biplane with body reference frame axes and Euler angles noted in black, and the outputs of the longitudinal controller and lateral controller denoted in blue and red respectively. 59

3.3 Schematic representation of the QRBP, inertial, longitudinal axes, and pitch angle. The UAV's roll and yaw axes are denoted by x and z, respectively. The thrust generated by the propellers along the $-z^{\mathbb{J}}$ axis is denoted by T_{upper}. The thrust generated by the propellers along the $z^{\mathbb{J}}$ axis is denoted by T_{lower}. 69

3.4 Schematic representation of the thrust forces acting on a quadbiplane UAV. The vector $T_{\text{u,l}}(\cdot)$ denotes the thrust force produced by the left propeller of the upper pair, $T_{\text{u,r}}(\cdot)$ denotes the thrust force produced by the right propeller of the upper pair, $T_{\text{l,r}}(\cdot)$ denotes the thrust force produced by the right propeller of the lower pair, $T_{\text{l,l}}(\cdot)$ denotes the thrust force produced by the left propeller of the lower pair, the upper left propeller and the lower right propeller are assumed to rotate clockwise and the lower left propeller and the upper right propeller are assumed to rotate counter-clockwise. 84

4.1 (Left)An early iteration of the thrust measurement stand. Due to limited machining access, the lever arm swings on 3D printed bushings and the plywood comprising the super structure is bolted in such a way to allow the plate to be realigned as necessary to avoid the apparatus binding up. The electronics necessary to run the experiment are included as they are identical to those used on the QRBP. (Right) A comparison of the thrust generated by a particular throttle input and the data provided by the motor's manufacturer. The experimental thrust values are significantly lower than the manufacturer's data and the relation of throttle to resulting thrust is not linear. 87

4.2 Diagram of the electronics comprising the control system of the QRBP. Boxes with a dashed border indicate values and boxes with solid borders indicate physical components or algorithms. The micro-controller, a Pixhawk 6c, combines and then filters the data received from the listed sensors. This data is provided to the control algorithm described in Chapter 3 which returns the normalized signals $[U_1, U_2, U_3, U_4]^{\mathrm{T}}$. The micro-controller applies these throttle values to the motors via the ESC to each motor. 89

4.3 Diagram representation of the fixed relation of the $\mathbb{J}$ and $\mathbb{P}$ reference frames of the QRBP. 90

4.4 91

4.5 98

4.6 Displayed here is a test of the QRBP's longitudinal control in which a small hover is performed. In this configuration, the polyethelene bracket provides torsion against the QRBP's lateral states of movement. Obstructing the view of the experiment is safety netting that drapes the VICON motion capture system used in these experiments. 100

4.7 Displayed here is a configuration of the static structure that constrains all of the QRBP's motion but roll and yaw. 102

4.8 Displayed here is the initial Inverted-Pendulum Stand shown with (left) and without (right) the QRBP mounted to it. 103

4.9 The redesigned Inverted-Pendulum Stand shown with (left) and without (right) the QRBP mounted to it. 104

4.10 An experiment of the QRBP's attitude control while mounted to the Inverted-Pendulum Stand. 105

4.11 A test of the QRBP's attitude control, given a sinusoidal ϕ command, while mounted to the Inverted-Pendulum Stand. 106

4.12 The previous configuration of the QRBP, originally presented in [4]. 108

4.13 The current configuration of the QRBP. 109

5.1 The QRBP's longitudinal states during hover flight. The actual measured state(black) and the commanded trajectory(red) are displayed. The states are sorted by the order in which they are output from the proposed controller. In this trajectory, we command the vehicle to ascend at a rate of $0.5\frac{\text{m}}{\text{s}}$ to a height of 1m, hold position for 10s, and then descend at a rate of $0.5\frac{\text{m}}{\text{s}}$. . . . 114

5.2 The QRBP's lateral states during hover flight. The actual measured state(black) and the commanded trajectory(red) are displayed. The states are sorted by the order in which they are output from the proposed controller. During this flight the commanded trajectory of the outer-loop lateral states is for y and ϕ to remain at zero. 115

5.3 The QRBP's longitudinal states during forward flight. The actual measured state(black) and the commanded trajectory(red) are displayed. The states are sorted by the order in which they are output from the proposed controller. In this trajectory, we command the vehicle to ascend at $0.5\frac{\text{m}}{\text{s}}$ to a height of 1m, hold position for 5s, we then command a step-input trajectory to x_{cmd} of 6m, hold position for 12s, and then descend at a rate of $0.5\frac{\text{m}}{\text{s}}$. 116

5.4 The QRBP's lateral states during forward flight. The actual measured state(black) and the commanded trajectory(red) are displayed. The states are sorted by the order in which they are output from the proposed controller. During this flight the commanded trajectory of the outer-loop lateral states is for y and ϕ to remain at zero. 118

List of Tables

2.1 Parameters needed to define the boundary conditions for position and yaw control of the proposed trajectory planner for $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$. 35

2.2 Coefficients of 1^{st} order linear regression relating path size and computation time 49

Chapter 1

Introduction

1.1 Motivation and Scope

This thesis presents the first step toward a long-term, multi-stage, research project aimed at the design of an integrated guidance, navigation, and control system for vertical take-off and landing (VTOL) autonomous unmanned aerial vehicles (UAVs) operating in contested, cluttered environments. In particular, the first part of this thesis studies the design of a vision-based guidance system aimed at tactical mapping of unknown, potentially hostile environments. In its second part, this thesis presents a robust adaptive control system for a specific class of VTOL UAVs, namely quadrotor-biplanes (QRBPs), such as the one shown in Figures 1.1 and 1.2. These aircraft are equipped with four propellers that, as quadcopters, allow for VTOL operations, as well as two pairs of wings that provide the efficiency necessary for long-distance cruises. As identified by Colonel Mark A. Moser of the US Army Aviation Center of Excellence in 2011, the Unmanned Aerial Vehicle (UAV) has the potential to dramatically improve reconnaissance and surveillance for individual military units [5]. With the Russo-Ukrainian war, ten years after Colonel Moser's observation, the world has seen a significant emergence in the use of UAVs on the battlefield [6, 7, 8, 9]. As such, the demand for more autonomous and more efficient UAVs operating in contested environments continues to grow.

The proposed guidance system assumes that the UAV is equipped with front-facing cam-

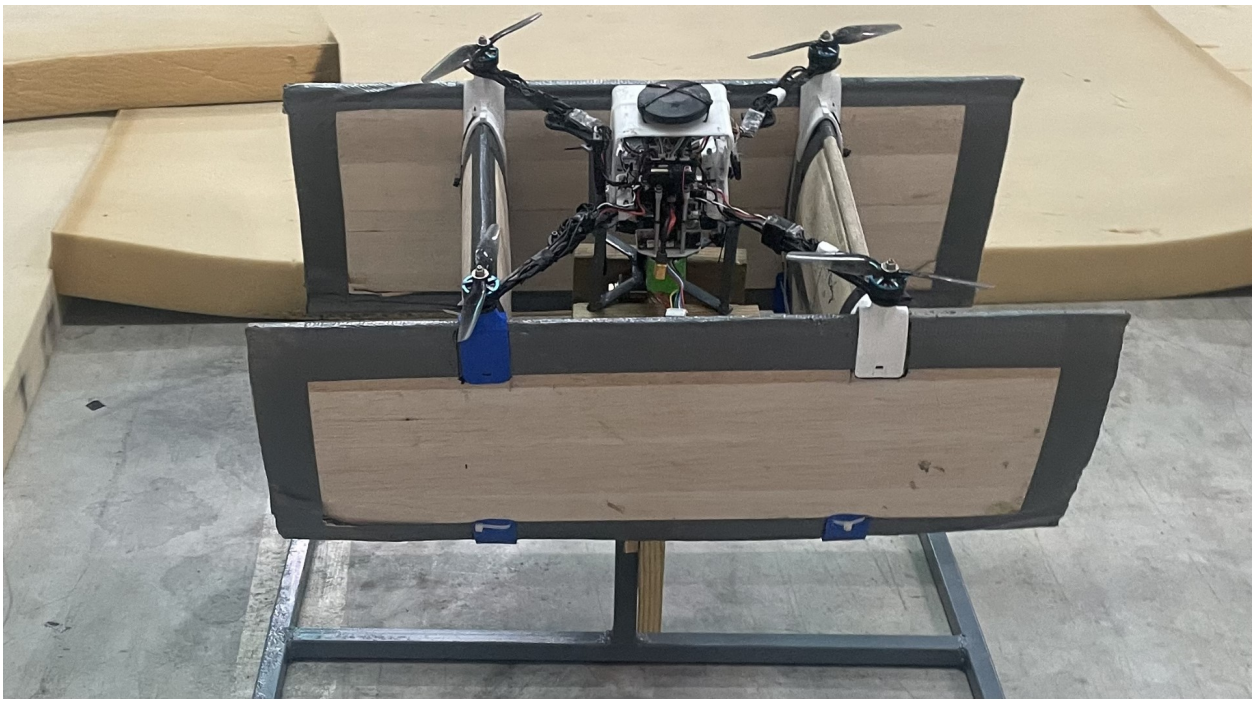


Figure 1.1: The Advanced Control Systems Lab's current configuration of a QRBP.

eras and operates at low altitude in cruise. A key facet of this work allows the user, through specific parameters, to pre-determine the UAV's approach to navigating an environment. Once placed into an environment in which no previous information is provided to the guidance system, these user-defined parameters will determine the degree to which the UAV will navigate recklessly or tactically, that is, in a manner that is more or less prone to exposing the UAV to opponents[1, 10]. This guidance system leverages and further expands the results presented in [11], which were initially designed for point-to-set navigation, and enables mapping of unknown environments. The efficiency of the proposed vision-based guidance system for mapping is proven by means of numerical simulations in highly complex environments, such as two-story buildings.

The second part of this thesis presents a unique control system for QRBPs, which is based on switched systems theory. Indeed, consistent with the literature on classical fixed-wing aircraft, we separate these UAVs' equations of motion into longitudinal and lateral-

Figure 1.2: The CRC-20 QRBP, a design inspiration for the ACSL's QRBP implementation.

directional modes. Within the longitudinal mode, we recognize that the vehicle dynamics are captured by substantially different equations when operating at high pitch angles, that is, when the QRBP operates more like a quadcopter, than at low pitch angles, that is, when the QRBP operates more like a biplane. To enable safe operations across flight modes and prevent instabilities due to rapid switching between modes, a control system based on switched dynamical systems theory is deemed necessary. The proposed control system is validated though numerical simulations as well as outdoor flight tests. To enable such flight tests on a QRBP already developed at the Advanced Control Systems Lab (ACSL) [4, 12], extensive characterizations of the motor-propeller combination were performed. Successively, the UAV's rotational dynamics were tested on a gimbal, custom-designed for this research. Finally, outdoor flight tests were performed. It is worthwhile to remark how the control

system presented in this thesis is considerably more articulated than those presented in [4, 12], which were predicated on the classical model reference adaptive control and the H_∞ architectures, respectively.

1.2 Thesis Outline

This thesis is structured as follows. The rest of this chapter surveys the literature involved with the fields of research covered in this thesis. In particular, we survey the literature on the guidance of autonomous UAVs, with special emphasis on those employed in contested environments. Successively, we discuss the state-of-the-art in control systems for QRBP UAVs. Finally, we review the literature the characterization of rotor for UAVs, set-ups for static experimentation for UAVs, and QRBP modeling.

In Chapter 2, details of a novel guidance system for tactical mapping of unknown environments are presented. Special emphasis is given to the characterization of the algorithms that comprise the guidance system in terms of their computation time and impact of user-defined parameters upon the system's ability to execute tactical maneuvers, that is, maneuvers that reduce the risk of detection by opposing agents, whose location and threat capabilities are unknown. The proposed guidance system, in very broad terms, contains a path planner and a trajectory planner. Within the path planning portion, the proposed guidance system must determine a sequence of goal points to completely explore an unknown environment characterized by some user-defined closed, connected boundary. This goal selection algorithm enables greedy or systemic exploration of the environment. To enable fast computations and rapid explorations of the environment, the map of the environment produced in real time using the onboard cameras is stored employing the Octree algorithm. Part of this thesis assesses the computational speed of this approach. Having selected a goal point, a sequence

of waypoints from the UAV's current location to the goal point is determined by means of heuristic-based algorithms. In particular, two algorithms are considered, namely A* and LPA*, and their performance compared by means of a relevant, high-fidelity numerical simulation. The formulation of A* considered in this thesis allows reference paths to coast obstacle sets and thereby behave more tactically. LPA* is an augmented version of A* in which previously calculated paths are reused when possible, and, hence, is faster and more suitable in the presence of moving obstacles. The trajectory planning portion of this guidance system is built on a fast model predictive control (fMPC) algorithm. To enable more aggressive maneuvers, the fMPC algorithm is applied to the feedback-linearized equations of motion of the UAV. To enable collision avoidance, the guidance algorithm employs dedicated algorithms aimed at identifying convex sets containing the UAV and excluding point clouds that belong to obstacles, which are identified by the onboard cameras. In this thesis, we compare four algorithms, based upon their computation times and generated volumes to find the most suitable for the proposed guidance system. Part of these results are presented in [1, 10].

Chapter 3 marks the beginning of the second part of this thesis, which is dedicated to the design and testing of adaptive control systems for VTOL UAVs. In particular, in Chapter 3, we present a proposed control algorithm for the QRBP, split into a controller for the longitudinal and lateral states. Next, we discuss how classical MRAC can be modified to enable tracking of a user-defined signal instead of tracking the trajectory of a user-defined reference model. Finally, we show how classical MRAC can be extended to regulate plant models, whose dynamics are affected by switching between distinct modes. The implementation of these algorithms is detailed along with the modeling used of the QRBP.

In Chapter 4, we present solutions to the challenges presented in flying the QRBP with the controller proposed in Chapter 3. Namely, we present methods to accurately characterize

the performance of QRBP's rotors, refine the process of tuning and validating the controller, and make improvements to the physical design of an existing QRBP to make it more flight capable. To most effectively apply the forces and moments generated by the proposed control system, we delve into how to characterize the thrust and reaction torque of each motor onboard the QRBP. Existing solutions to this issue [13] use a different micro-controller that is not directly analogous to what is available to us in our flight architecture. Rather than changing the flight control system's architecture, we develop a method to measure our forces and moments with our existing micro-controller. For refinement of the proposed control system, experimental set-ups are developed so that specific modules of the proposed control system may be tuned before actual flights.

In Chapter 5, the theoretical and numerical results presented in Chapter 4 are tested by means of two sets of flight tests of increasing complexity. In Chapter 6, we draw conclusions on the material presented in this thesis. From our own findings and that of the literature, we discuss the implications of this work and future efforts to continue it.

1.3 Literature Review

In this section, we survey the literature concerning the key topics of this thesis, namely guidance for UAVs in contested environments and control of QRBPs. From this, we can draw conclusions about where gaps in knowledge lie and how to improve future work.

1.3.1 Autonomous Guidance for UAVs in Contested Environments

In the available literature, subject matter on autonomous guidance is typically presented as individual algorithms, such as path planners, trajectory planners, or collision avoidance

algorithms, whereas the proposed guidance system is a much more comprehensive approach. As such, we first begin by acknowledging previous attempts to make the processing of an environment more efficient, such as the approach shown in [14]. Furthermore, there is a relatively small literature on the problem on guidance systems for autonomous UAVs operating in contested environments [11]. In this paper, the authors discuss the implementation of a quadtree data structure for representation of the environment in which a mobile robot must navigate. The implication of this is a more efficient way in which an onboard computer can process its environment and subsequently make decisions.

An approach to mapping a potentially hostile environment is to cause the mapping UAV to maneuver in an unpredictable way. To achieve this behavior, [15] presents a guidance system in which a modified logistics map and a modulo tactic are used. Another approach involves the use of nonlinear programming [16]. This approach maximizes the area being scanned by the UAVs sensors and tracks which areas have not been explored recently. The approach is validated on increasingly complex environments and with multiple mapping UAVs. Other approaches rely on genetic algorithms for trajectory planning, such as [17]. This approach uses genetic algorithms to produce reference trajectories for groups of UAVs operating as a swarm in low altitudes and potentially hostile environments.

The subject of generating obstacle free spaces within a contested environment is also studied in the literature. One such approach, as shown in [18], proposes to partition a navigation space, which contains obstacles, with a convex lifting algorithm. They then construct feasible corridors based upon the generated graph of interconnections in the obstacle ridden environment. From this, they can then propose a path with obstacle avoidance guarantee. Another approach, found in [19], presents a novel method of decomposing an obstacle-containing environment into convex sets of obstacle-free regions. This is accomplished via stereographic projection with the sub procedures of convex hull generation and inverse vertex enumera-

tion. The advantage of this approach is its lack of reliance on computationally demanding numerical optimization methods as it instead relies on more simple geometric properties.

1.3.2 Control Algorithms for Quadrotor Bi-Planes

The existing literature on robust, adaptive control specifically for QRBP's is limited. Direct MRAC has been implemented as seen in [4]. This thesis derives a longitudinal control system for the QRBP built on the theory of MRAC and then provides real-world flight tests for validation. Other robust control architectures have been implemented, such as [12]. In this thesis, the theory of H∞ is implemented to produce a robust longitudinal control system for the QRBP. More advanced modifications of the MRAC theory have been developed for the longitudinal control of QRBPs as seen in [20]. In this paper, the authors present MRAC for prescribed performance in which the operator can dictate bounds on the trajectory tracking error and the tracking error's convergence rate without modifying the reference model.

1.3.3 Refinements to the Experimental Process of Developing the Quadrotor Bi-Plane

There is significant literature regarding the modeling and validation of electric multirotor unmanned aerial vehicle system energy dynamics. The primary application is to characterize the thrust of rotors modified for a very particular application. In one such example, the author's present a mathematical model of a rotor for use on an octocopter UAV [21]. They validate the model by measuring the thrust and reaction moment forces with a measuring device. The device sends direct PWM signals and the authors explore the conversion between throttle inputs and the PWM signals. In another example, the author's present design modifications to the rotors used on a UAV intended for mining purposes [22]. To verify

the validity of their designs, they employ a Tyto robotics thrust test stand to perform their measurements.

There is however, not much literature in regards to experimental setups that constrain the movement of a UAV in one manner or another. One example that does exist is a detailed description of the building of a constraint setup for very large quadcopters [23]. In this experimental setup, the vehicles motion is constrained and a load sensor for each rotor allows for the forces of the vehicle to be tracked. The approach is rudimentary relies on more moving parts than the more common Tyto measuring stands.

The literature around the design of QRBP heavily features modeling. In future design considerations, the resulting ability to model the final design should be a very high priority. For example, the impact of the rotors upon the airframe is non-negligible as noted in [24]. The authors derive a model of QRBP that includes the effects of propwash from the rotors on the aerodynamics of the vehicle. They then derive a backstepping algorithm for control of this vehicle and validate it in simulation. In another detailed design, the authors present a methodology with which to understand how the dynamics of a QRBP change with size [25]. The authors present a scalable vehicle configuration for QRBP style vehicles. In doing so, they provide a model for other researchers to base their own QRBP models off of. Naturally, some literature simply details improvements to the physical design of QRBP's as in [26]. The authors present a design and validation of a QRBP. They provide their design methodology, production techniques, and detail their design considerations and trade-offs. They validate their design with flight tests.

Chapter 2

A Tactical Guidance System of Autonomous UAVs

2.1 Overview of the Proposed System

Given the problem of mapping and exploring unknown environments with varying degrees of danger, the use of autonomous UAVs is a proposed solution. In this application, it is assumed that due to the danger presented by the situation, that the UAV will operate completely isolated from external communication and data sources such as GPS.

In this chapter, we propose a novel guidance system for autonomous UAV operating in unknown and potentially hostile environments, where the location and threat capabilities of the opponents, if any, are unknown. In this guidance system, the three-dimensional area in which the UAV operates is considered to be of fixed dimensions and partitioned into parallelipids of fixed size and known as *voxels*. As such, the environment in which the UAV operates will be referred to as the *voxel map* and all of the following algorithms are implemented via this dissection of the environment. The UAV model used in the implementation of this guidance system relies solely on forward-facing cameras to provide motion tracking and obstacle detection.

The first problem addressed in this thesis concerns an efficient way to survey a given

environment that allows for tactical operations. Classical approaches, which consist in systematically sweeping the environment, are clearly inappropriate, and alternative methods are needed. To design sequences of goal points for the UAV to follow and, hence, map the environment with its onboard cameras, we resort to the octree algorithm [27], [28] to store voxel data and choose goal points efficiently. In the Octree algorithm, dara are stored in a hierarchical structure. In this structure, each *parent* node has eight *children* nodes comprised of portions of the original data set. In the proposed guidance system, each node of the Octree algorithm represents a cluster of voxels, also know as bin. These bins yield groups of contiguous voxels that can be classified by various measure of the comprising voxel's explored and occupied status. The goal selection process then uses these voxels that have been grouped into bins as the basis for selecting goal points. This implementation of the Octree algorithm allows for the user to optimize how the voxel map is partitioned based upon a given voxel's state of being explored and occupied. The goal point is selected as the barycenter of the particular bin selected by the goal selection algorithm. In this thesis, the computation times required to generate bins for a particular map are evaluated.

Having selected a goal point, the UAV's reference path, that is, a sequence of waypoints, and then the reference trajectory interpolating those waypoints must be found. The UAV's path and the associated trajectory define the vehicle's behavior. The UAV's behavior will span an infinitely large spectrum, which ranges from purely reckless, wherein the UAV aims for its next goal point irrespective of the presence of threats, to purely tactical, wherein the UAV exploits every possible obstacle to seek shelter while continuing towards the goal point. The higher the degree of tacticality, the longer the UAV's trajectory and subsequent flight time may be. Essentially, the navigation system needs to decide where to go and how to get there. The path planner is selected to optimize the computational time required to find a suitable path. Two path search algorithms, A* and its variation Lifelong-Planning-A*

(LPA*), are selected as candidates due to their incorporation of a heuristic into their search. The LPA* search algorithm is an augmentation to the A* algorithm as it incorporates a mechanism allowing it to reuse paths if the voxel map remains unchanged. Thereby, LPA* should require less computation time as the UAV traverses its environment.

Once a path has been planned, a trajectory interpolating this path must be determined which requires quantifying where the UAV can actually travel. To do so, the free space surrounding the computed path must be defined. This is done by a *constraint generation* algorithm that produce a closed set of planes that separate the obstacles from the free space surrounding the path. Four algorithms are compared on the basis of the time to calculate closed sets of constraint planes and subsequent volume generated.

2.2 Octree Algorithm for Map Partitioning

To autonomously map an environment, the path planner must aim the UAV towards unexplored areas and away from already explored areas while simultaneously avoiding obstacles. In this section, we present the key elements of the algorithmic framework used to partition maps, and, hence, identify goal points for the UAV. This section is structured as follows. In Subsection 2.2.1, we provide a summary of the Octree algorithm. Subsection 2.2.2 explains the theory employed by the Octree algorithm. Subsection 2.2.3 provides an explanation of the notation used in this section. Subsection 2.2.4 shows the testing procedure used to evaluate the Octree algorithm along with the results of this experiment.

2.2.1 A Brief Overview

The voxels comprising the environment have two classifications. A voxel in the map is either explored or unexplored, and occupied or unoccupied. Voxels are considered explored when they have been seen by the UAV's camera system, and voxels are considered occupied when the probability that their interior is occupied reaches a confidence level above a user-defined threshold. By default, voxels are considered unexplored and unoccupied to encourage a more aggressive behavior. However, several guidance systems for UAV's employed to cover unknown environments assume that, by default, voxels are unexplored and unoccupied. This alternative approach enables a more cautious behavior, but prevents the UAV from investigating areas that are difficult to access. The Octree algorithm [28] [27], Algorithm 2.2.1, creates partitions about the multiple ways of defining acceptable levels of exploration and occupation by obstacles. For flights with higher degrees of recklessness, the goal point is taken as the barycenter of *bins* with the highest number of unexplored voxels. For flights with a higher degree of tacticity, the UAV selects the barycenter of adjacent *bins*. Having selected a goal point, the UAV's path in terms of recklessness and tacticity is determined by the heuristic used by the path planner.

2.2.2 Theory

In this thesis, the Octree algorithm is employed by partitioning the space in the parallelepids, also known as *bins*. Thus the barycenter of each bin is identified, and the bin is partitioned into 8 sub-bins. The bin before it has been partioned is referred to as the *parent bin*, and the resulting bins post-partion are known as *children bins*. A corner of each child bin is located at the barycenter of their parent bin. The 4 user-defined parameters to decide when to partition a *bin* are:

Algorithm 2.2.1 Octree iterative algorithm to partition voxel map

Initialize $\hat{\mathcal{P}}_0 =$ minimum bounding box, $n_{\mathcal{P}} = 1$
Compute $n_{\hat{\mathcal{P}}_0}$, $n_{\text{explored},\hat{\mathcal{P}}_0}$, $n_{\text{occupied},\hat{\mathcal{P}}_0}$, $\ell_{\hat{\mathcal{P}}_0,\min}$
procedure DIVIDE($\hat{\mathcal{P}}_i$)
    Compute vertices of child parallelipipeds $\{\hat{\mathcal{P}}_{n_{\mathcal{P}}+1}, \ldots, \hat{\mathcal{P}}_{n_{\mathcal{P}}+8}\}$
    Compute $n_{\hat{\mathcal{P}}_j}$, $n_{\text{explored},\hat{\mathcal{P}}_j}$, $n_{\text{occupied},\hat{\mathcal{P}}_j}$, $\ell_{\hat{\mathcal{P}}_j,\min}$, $j \in \{n_{\mathcal{P}}+1, \ldots, n_{\mathcal{P}}+8\}$
    $n_{\mathcal{P}} = n_{\mathcal{P}} + 8$
    for $j = (n_{\mathcal{P}} - 7){:}n_{\mathcal{P}}$ **do**
        if $\frac{n_{\text{explored},\hat{\mathcal{P}}_j}}{n_{\hat{\mathcal{P}}_j}} < \mu_1$ **and** $\frac{\ell_{\hat{\mathcal{P}}_j,\min}}{2} \geq \mu_2$ **then**
            if $n_{\text{explored},\hat{\mathcal{P}}_j} \geq \mu_3$ **or** $\frac{n_{\text{occupied},\hat{\mathcal{P}}_j}}{n_{\hat{\mathcal{P}}_j}} \geq \mu_4$ **then**
                Divide($\hat{\mathcal{P}}_j$)
            end if
        end if
    end for
end procedure

1. μ_1: the maximum ratio of the number of explored voxels to total voxels.

2. μ_2: $\frac{1}{2}$ the length of the smallest side of a partition

3. μ_3: largest number of explored voxels in a given partition

4. μ_4: maximum ratio of occupied voxels to total voxels.

2.2.3 Notation

A detailed description of how the Octree algorithm is employed to partition maps is shown in Algorithm 2.2.1. The notation used to describe this algorithm is summarized as follows. $\mathcal{V} \subset \mathbb{R}^3$ denotes the set of contiguous voxels that comprise the environment to be mapped. $n_{\mathcal{P}}$ is the number of partitions of the set $\mathcal{V}$. $\hat{\mathcal{P}}_a$ is a rectangular parallepid that contains the set of partitioned voxels, the combination of $\hat{\mathcal{P}}_a, a \in \{1, \ldots, n_{\mathcal{P}}\}$ comprises $\mathcal{V}$. $\hat{\mathcal{P}}_0$ is the smallest partition of $\mathcal{V}$. $n_{\hat{\mathcal{P}}_0}$ is the total number of voxels contained within $\hat{\mathcal{P}}_0$. $n_{\text{explored},\hat{\mathcal{P}}_0}$

is the number of explored voxels contained within $\hat{\mathcal{P}}_0$. $n_{\text{occupied},\hat{\mathcal{P}}_0}$ is the number of occupied voxels contained within $\hat{\mathcal{P}}_0$. $\ell_{\hat{\mathcal{P}}_0,\min}$ is the length of the shortest side of $\hat{\mathcal{P}}_0$.

2.2.4 Octree Performance Test and Results

For testing, a map of dimensions 20x20x6m^3 is employed as it is roughly representative of a potential office/business space to be explored. 10 maps are created that contain randomly placed 1x1x1m^3 obstacles. The percentage of the map occupied by obstacles is increased in 10 even increments until 30% of the map is occupied. For each of these generated maps, 10 sets of occupied voxels are generated. This process is repeated 10 times for a grand total of 1000 maps to be tested and is done to eliminate potential bias.

In Algorithm 2.2.1, the first condition to split a bin is based upon (μ_1, μ_2) and the second condition is based upon (μ_3, μ_4). Therefore, the test is performed by varying each pair of (μ_1, μ_2) against each pair (μ_3, μ_4) on the 1000 maps. The output of each simulation is the computation time required to partition the map.

The values for the user defined parameters are chosen as the following: The parameter μ_1 is varied on the interval $[0.30, 0.90]$ as values less than 0.3 do not elicit partitioning and values over 0.9 create partitions that are impractically small. The parameter μ_2 is varied in the interval $[0.2, 2.40]$ as 0.2m is the edge length of one voxel while the upper bound of 2.40m is selected as it represents an edge length of 12 voxels which is the largest width of an area to be considered for exploration of the UAV. The parameter μ_3 is varied on the interval of $[4, 160]$ as 4 voxels is roughly the size of the model of the UAV and 160 voxels occupy the maximum diameter of the cone representative of the mapping camera. The parameter μ_4 is varied on the interval of $[0, 0.30]$ as areas more than 30% occupied are inaccessible to the UAV.

Figure 2.1 shows a heatmap of the averaged computational times in milliseconds for each combination of (μ_1, μ_2) and (μ_3, μ_4). Figure 2.2 is a boxplot of the Algorithm 2.2.1 as a function of (μ_1, μ_2) with the averaged out results of (μ_3, μ_4). The mean length of the whiskers are 1.46ms and the mean interquartile range is 0.40ms. The median computational tends to increase with larger values of (μ_1, μ_2), however the relation is non-linear and approximated with the line of best fit:

$$p(\tau) = 0.0004\tau^7 - 0.0165\tau^6 + 0.2882\tau^5 - 2.4686\tau^4 + 10.6710\tau^3 - 22.6303\tau^2 + 30.2912\tau + 65.0882, \tau \in [1, 12]$$

where $p(\tau = 1)$ approximates the median computational time in milliseconds of $(\mu_1, \mu_2) = (0.35, 0.2)$ and $p(\tau = 12)$ approximates the median computational time in milliseconds of $(\mu_1, \mu_2) = (0.9, 2.40)$.

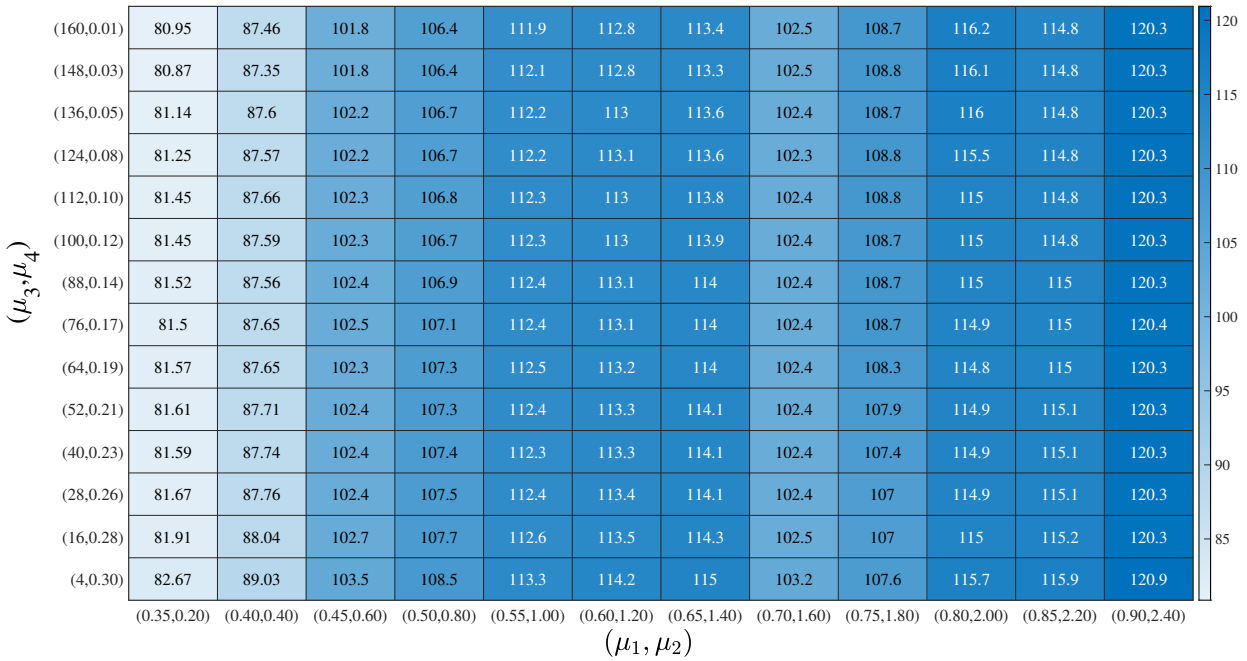


Figure 2.1: Computational times of each possible configuration of the sets (μ_1, μ_2) and (μ_3, μ_4)

Figure 2.3 is a boxplot of the Algorithm 2.2.1 as a function of (μ_1, μ_2) with the averaged out results of (μ_3, μ_4). The mean length of the whiskers are 1.37ms and the mean interquartile range is 0.34ms. The median computational tends to increase with larger values of (μ_1, μ_2), however the relation is non-linear and approximated with the line of best fit

$$p(\tau) = -0.0009\tau^7 + 0.0424\tau^6 - 0.7817\tau^5 + 7.3940\tau^4 \\ - 38.0322\tau^3 + 104.3633\tau^2 + -147.5677\tau + 231.3665, \tau \in [1, 12]$$

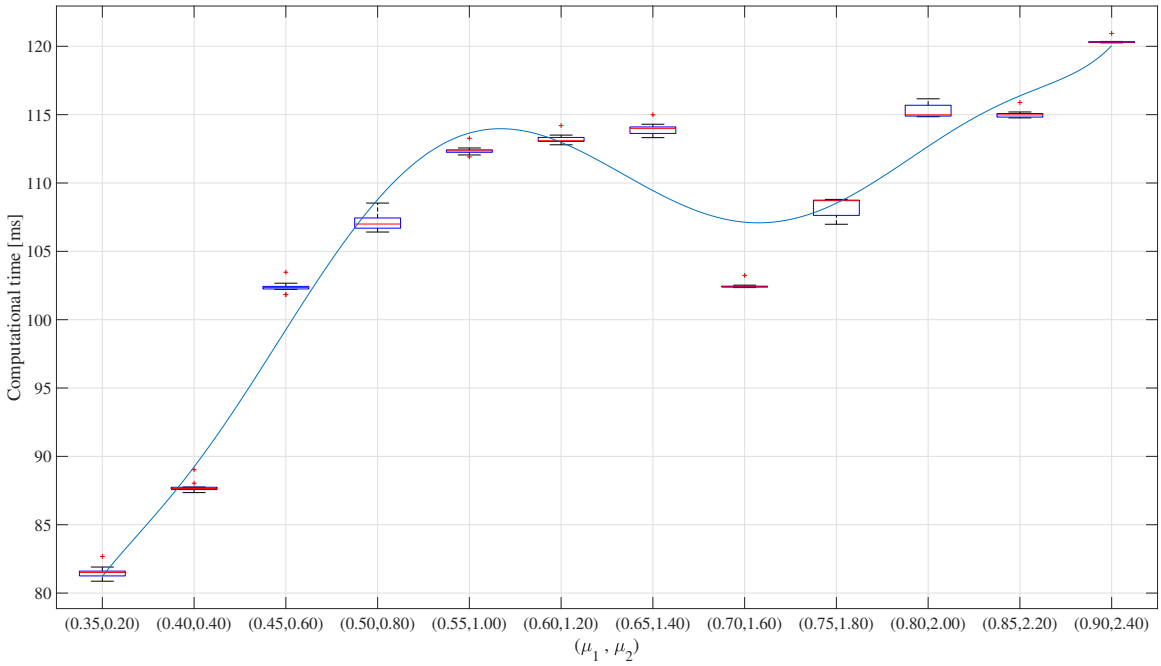


Figure 2.2: Computational times of Algorithm 2.2.1 as a function of the set (μ_1, μ_2) over the averaged results of varying (μ_3, μ_4) where both μ_1 and μ_2 are increasing.

Figure 2.4 is a boxplot of the Algorithm 2.2.1 as a function of (μ_1, μ_2) with the averaged out results of (μ_3, μ_4). The mean length of the whiskers are 38.84ms and the mean interquartile range is 12.00ms.

The positive slope of the median computation times in Figure 2.2 and the negative slope

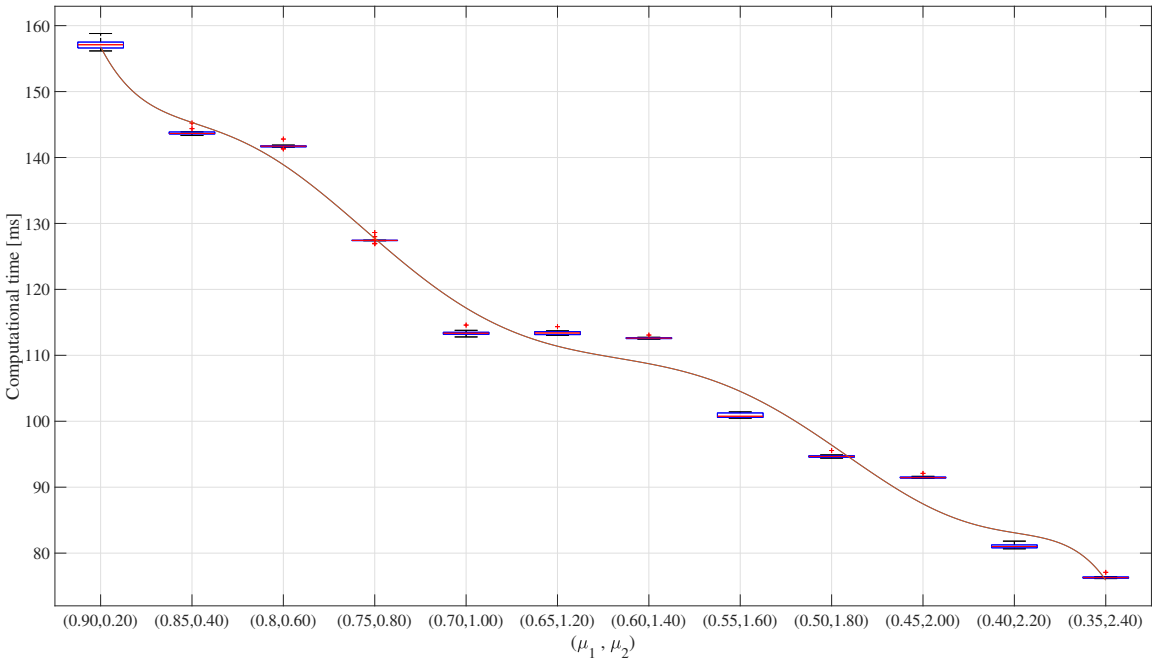


Figure 2.3: Computational times of Algorithm 2.2.1 as a function of the set (μ_1, μ_2) over the averaged results of varying (μ_3, μ_4) where μ_1 increases and μ_2 decreases

displayed in Figure 2.3 show that the dominant parameter in computation time is μ_1. It appears from Figure 2.4 that (μ_3, μ_4) has a minimal impact on the computational time of Algorithm 2.2.1 and can therefore be selected purely upon the basis of the resulting partitions. Therefore, in selection of parameters in terms of reducing computational time cost, small values of μ_1, larger values of μ_2 , smaller values of μ_3 and larger values of μ_4 are ideal.

2.3 Comparison of Path Planning Algorithms

Having partitioned the map as described in Section 2.2.2, a goal point is selected and a path from the current position to said goal point is handled by the path planning algorithm. This section, provides an explanation of the process to compare and select an appropriate path planning algorithm. Subsection 2.3.1 provides an overview of why certain algorithms

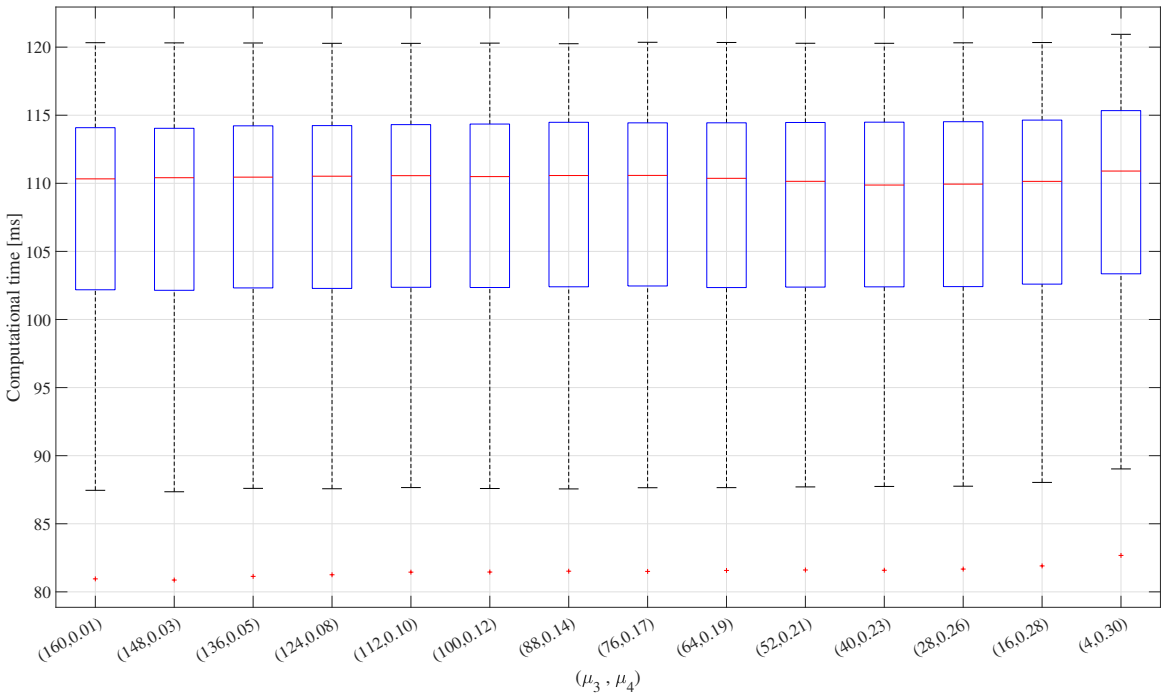


Figure 2.4: Computational times of Algorithm 2.2.1 as a function of the set (μ_3, μ_4) over the averaged results of varying (μ_1, μ_2)

are selected and the criteria upon which they will be compared. Subsection 2.3.2 provides the notation used, specifically as it is relevant to describe the underlying theory. Finally, Subsection 2.3.3 outlines the underlying theory of A* and LPA*

2.3.1 A Brief Overview

Two path planning algorithms have been considered for this study, namely A* [29] and LPA* [30]. A* is used because it features a heuristic that allows for the user to define the degree of tacticality and recklessness of the flight. LPA* is an augmentation of A* that allows for paths to be reused in areas of the map that do not change. It is expected that LPA* is faster than A* at calculating new paths. However, its implementation is more challenging. Through numerical simulation, the computation times of both algorithms are compared to

verify that the added complexity of LPA* provides a benefit over A*.

2.3.2 Notation

A detailed description of how the A* and LPA* algorithm's are employed to plan paths is shown in Algorithm 2.3.1 and Algorithm 2.3.2 respectively. The notation used to describe these algorithms is summarized as follows. A finite graph denoted by $G = (X, A)$ consists of the set of nodes, X and the set of arcs A that connect said nodes. The total number of nodes within G is denoted by n. The number of potential arcs in G is denoted by r. Particular nodes are specified by N, most frequently it refers to the current node being evaluated. The node adjacent to N is denoted by N', such that there is an arc within G that connects N to N'. The node from which the path planning algorithm begins its search is denoted by N_{init}. The goal node which the path-planner seeks to connect via a cost-minimized arc to N_{init} is denoted by N_{goal}. A data tree spanning the arcs so far explored in G is denoted by T. A cost function for nodes within G is denoted by $f(N)$. The list of nodes within G sorted by values of f is denoted by $OPEN$. The function that calculates the cost of each arc in G is denoted as $k : X \times X \rightarrow R^+$.

2.3.3 Theoretical Formulation

A* **Optimization:** At its essence, A* takes the set of nodes G with a given start and end node, then finds a sequence of connected edges that is influenced by a user defined heuristic function. The A* algorithm starts with the assumption that N_{init} and N_{goal} are known and that the other nodes within G are otherwise unvisited. Starting with the start node, paths between nodes are extended one edge at a time until the *cheapest* sequence of edges that reaches the goal is found and the criteria for terminating the search is attained.While

exploring the cheapest path from the N_{init} to N_{goal}, the cost of a path is estimated as the cost of the path from N_{init} to the current node, namely N, and then estimating via the heuristic function the path's cost from N to N_{goal}. If a path from the start node to the goal node exists, then A* will return a path of minimum cost. The paths are saved to the tree T in which all nodes that are visited after N_{init} are placed on this list with a pointer to their parent node. The A* algorithm minimizes the function:

$$f(N) = g(N) + h(N) \tag{2.1}$$

where $g(N)$ is the cost of the path between N_{init} and N and $h(N)$ is a heuristic estimate of the minimum-cost path from the current node and the goal node [31].

In order for the $A*$ algorithm to return the optimal path, the heuristic function must be *admissible* and *consistent*. An *admissible* heuristic does not overestimate the cost of reaching a goal and as such guarantees that the optimal solution is not overlooked by guaranteeing that all arcs are evaluated with the same criteria. That is, all nodes are evaluated such that the path between them is assumed to be ideal and therefore of minimum cost. The *consistence* of the heuristic guarantees that the estimated cost of traveling from N to N_{goal} is less than or equal to the cost of traveling from N to N' and then from N' to N_{goal} [32],[33].

While computing the optimal path, the list $OPEN$ plays a pivotal role, and the operations on this list are:

- $FIRST(OPEN)$: returns the node with the smallest f value and removes it from $OPEN$;
- $INSERT(N, OPEN)$: inserts N into $OPEN$;

- $DELETE(N, OPEN)$: removes N from $OPEN$;
- $MEMBER(N, OPEN)$: checks whether N is contained within $OPEN$, returns *true* if N is within $OPEN$ and *false* if N is not contained in $OPEN$;
- $EMPTY(OPEN)$: checks whether there are still nodes in $OPEN$ to be checked.

Given 2.1 these operations, the A* algorithm works as detailed in Algorithm 2.3.1.

Algorithm 2.3.1 A* Algorithm

```
N_init is set as the first entry of T;
INSERT(N_init, OPEN);
N_init is marked visited;
while EMPTY(OPEN) == false do
    N ← FIRST(OPEN)                                  ▷ current node is selected
    if N == N_goal then                              ▷ Checking if the goal has been reached
        Exit
    end if
    for Each node N' do                              ▷ evaluates each node in G adjacent to N
        if N' is not visited then
            add N' to T by assigning N as the parent node
            INSERT(N', OPEN)
            mark N' as visited
        else if g(N') > g(N) + k(N, N') then
            change T such that the parent node of N' is now N
            if MEMBER(N', OPEN) == true then
                DELETE(N', OPEN)
            end if
            INSERT(N', OPEN)
        end if
    end for
end while
if EMPTY(OPEN) == true then
    return the list of connected nodes from N_goal to N_init
else
end if
```

LPA* **Optimization:** If the map changes in a way that impacts only a select portion of the previously calculated path, then it may not be necessary to recalculate the entire path as parts of the original path can be reused. LPA* provides a valuable tool address these situations. LPA* differs from A* by maintaining 2 estimates of start distance, the *g-value* defined above in 2.1 for A*, and a new value, the *right-rand-side-value(RHS-value). RHS-values* are *one-step lookahead* values based on the $g(\cdot)$ in 2.1 values(the path of least cost between N_{init} and N). If the *g-values* of N's predecessors are not equal to the *RHS-value*

of N, then the node is considered *locally inconsistent* and the path is recalculated from N. Locally inconsistent nodes are placed in a *priority queue*, where they are re-evaluated. These priority values are denoted as *key values*, denoted by k. There are two priority values assigned to each node, namely nodes listed in $OPEN$ are sorted by k_1 and then by k_2:

$$k(N) = \begin{bmatrix} k_1(N) \\ k_2(N) \end{bmatrix} = \begin{bmatrix} \min\left[g(N), rhs(N)\right] + h(N) \\ \min\left[g(N), rhs(N)\right] \end{bmatrix} \tag{2.2}$$

The operations applicable to the list $OPEN$ for use in LPA* are largely the same as A*, and the following additional operations are employed:

- $TOPKEY(OPEN)$: the priority value of the node with lowest priority value is returned unless $OPEN$ is empty, in which case ∞ is returned;
- $FIRST(OPEN)$: the node with the lowest priority value is returned and removed from $OPEN$;
- $CONTAINS(N, OPEN)$: checks whether N is contained within $OPEN$, returns *true* if N is within $OPEN$ and *false* if N is not contained in $OPEN$;
- $INSERT(N, k, OPEN)$: inserts N into $OPEN$ and its corresponding priority value are inserted into $OPEN$;
- $DELETE(N, OPEN)$: removes N from $OPEN$;
- $MEMBER(N, OPEN)$: checks whether N is contained within $OPEN$, returns *true* if N is within $OPEN$ and *false* if N is not contained in $OPEN$.

Given these operations, the LPA* algorithm works as detailed in Algorithm 2.3.2.

Algorithm 2.3.2 LPA* Algorithm

```
It is assumed that a set of connected nodes from N_init to N_goal exists
Initialize()
while Algorithm is running == TRUE do
   computeShortestPath()
   while Map Has Changed == FALSE do
      Sleep
   end while
   for Map Has Changed == TRUE do
      Update Node Costs
      UpdateVertex(N)
   end for
end while
function Initialize(G)                                   ▷ Setting all nodes ∈ G as Locally Consistent
   for All nodes ∈ G do
      g(N) = ∞
      RHS(N) = ∞
   end for
   RHS(N_init) = 0
   Key(N_init) = k_init
   INSERT(N_init, k_init, OPEN)
end function
function ComputeShortestPath(OPEN)
   while TOPKEY(OPEN) < Key(N_goal)||rhs(N_goal) == g(N_goal) do
      N = FIRST(OPEN)
      if g(N) > RHS(N) then
         g(N) = RHS(N)
      else
         g(N) = ∞
         UpdateVertex(N)
      end if
      for succesors of N do
         UpdateVertex(N_Succsesor)
      end for
   end while
end function
function UpdateVertex(N)   ▷ recalculates rhs value of N and removes it from OPEN. If N is locally inconsistent, it is reinserted into OPEN
with the new KEY(N) value
   if N! = N_init then
      RHS(N) = ∞
      for all Predecessors of N do
         RHS(N) = min (RHS(N), g(N_predecessor) + f(N))
      end for
      if MEMBER(N, OPEN) then
         DELETE(N, OPEN)
      end if
      if g(N)! = RHS(N)) then
         INSERT(N, KEY(N, OPEN))
      end if
   end if
end function
function CalculateKeyValue(N)
   Key(N) = [min (g(N), RHS(N)) + h(N), min (g(N), RHS(N))]
end function
```

2.3.4 Path-Planning Performance Tests and Results

To quantify the effectiveness of LPA* versus A* and whether it is worth while implementing this algorithm within the context of tactical path planning, these two algorithms are compared by means of two sets of software-in-the-loop numerical simulations. In the first set of simulations, the heuristic function parameters are set such that the A* and LPA* algorithms seek tactical paths. In the second set of simulations, the heuristic function parameters are set such that the A* and LPA* algorithms seek reckless paths. The test are performed on an Ubuntu running computer with an 8-core Intel *i*7 processor at 4.30GHz.

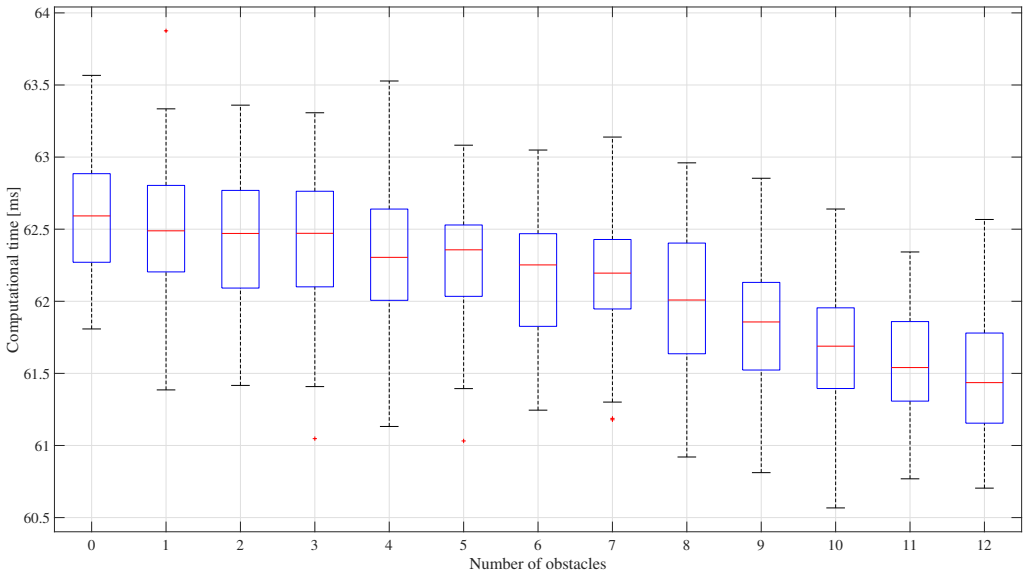


Figure 2.5: Computational time data of the A* Algorithm with a reckless parameter set. The average computation time across all obstacle sets is 62.1114ms with a standard deviation of 0.3421ms.

Standalone versions of Algorithm 2.3.1 and Algorithm 2.3.2 run on a voxel map capturing of an office space by the navigation system presented in [11] with given N_{init} and N_{goal} points that are selected to have calculable path between them. Despite being run on a previously recorded and thereby static G, obstacles are introduced to G to simulate a changing environment.

In testing, an initial path between N_{init} and N_{goal} is found on the recorded office map. Algorithms 2.3.1 and 2.3.2 are then executed to recalculate paths with incrementally increasing number of obstacles are placed along the previously generated path. Obstacles are placed at a distance from one another that is slightly longer than the length of the UAV model. With the selected N_{init} and N_{goal}, this results in each algorithm being run 13 times so that the final test involves 12 added obstacles. This process is repeated 50 times for both algorithms for both sets of tactical and reckless numerical simulations.

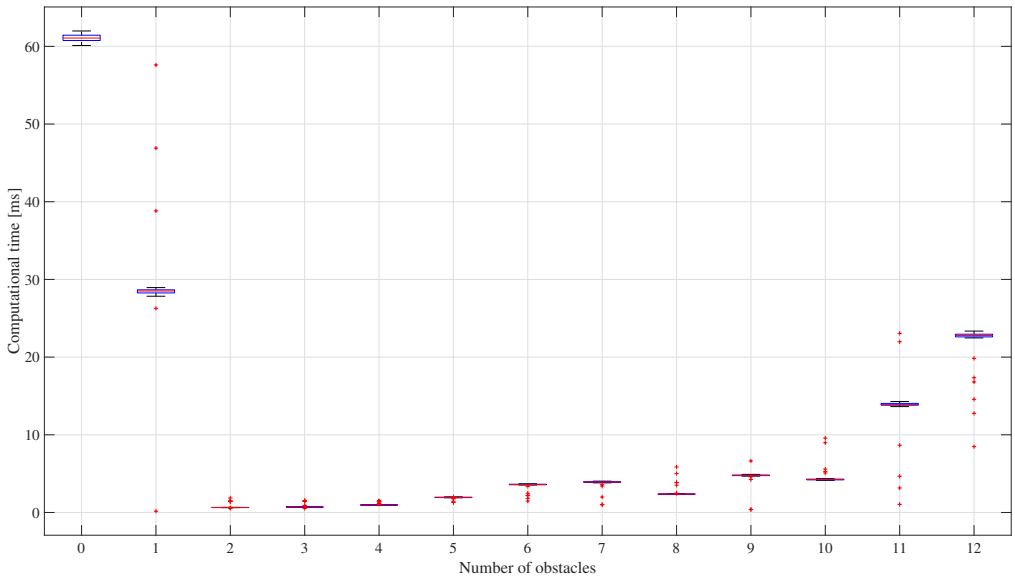


Figure 2.6: Computational time data of the LPA* Algorithm with a reckless parameter set. The average computation time across all obstacle sets is 11.4467ms with a standard deviation of 17.3388ms.

Figures 2.7 and 2.8 show the results of the numerical simulations of the A^* algorithm and the LPA^* algorithm respectively, with a tactical parameter set. Figures 2.5 and 2.6 show the results of the numerical simulations of the A^* algorithm and the LPA^* algorithm respectively, with a reckless parameter set.

Figures 2.5 and 2.7 show a near-constant computational time regardless of the number of additional obstacles added to the map. This is the predicted result as the A* algorithm is forced to recalculate the entirety of the path in each instance of the simulation. Figures 2.6 and 2.8 show the computation times taken by the LPA* algorithm using a reckless and tactical parameter set, respectively. From these figures, the LPA* algorithm generated comparable computation times compared to the A* algorithm for the tests with $[0, 1]$ additional obstacles added. For tests with 2 or more additional obstacles, the LPA* algorithm yielded significantly lower computational times that slowly increased as the number of added ob-

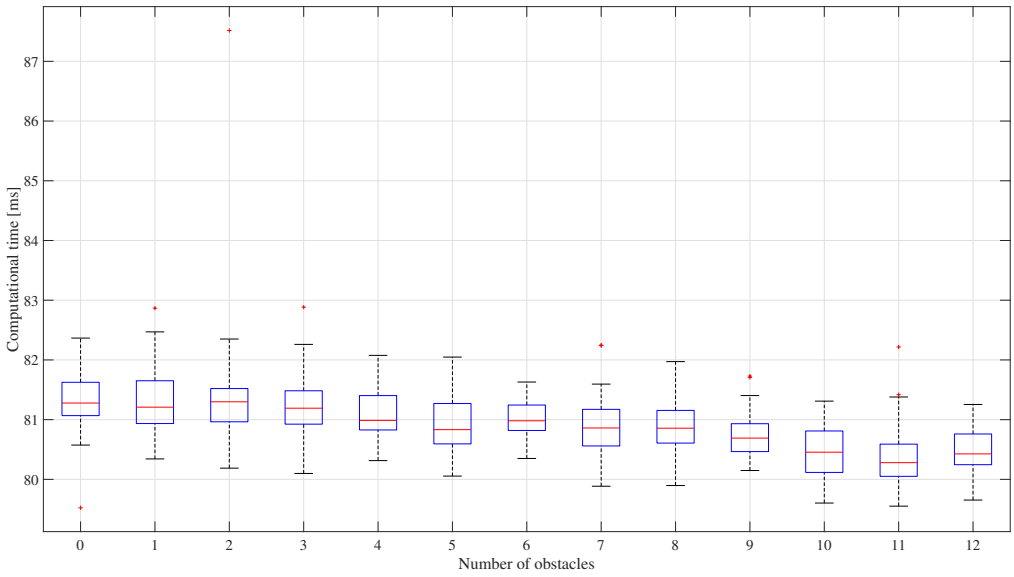


Figure 2.7: Computational time data of the A* Algorithm with a tactical parameter set. The average computation time across all obstacle sets is 80.9330ms with a standard deviation of 0.3715ms.

stacles increases. This increase in time correlated to the number of obstacles is due to the fact that additional obstacles reduce the length of previously generated paths that can be reused. Due to these results, the LPA* algorithm is selected as the better option.

2.4 Trajectory Planning

Having calculated a series of waypoints to follow, in the form of a reference path as discussed in Section 2.3, the proposed guidance system makes use of the MPC framework to produce an interpolation of multiple waypoints such that the UAV has a reference trajectory to follow. The underlying function accounts for the UAV's need to reach the next waypoint as well as thrust allocation. This Section is organized as follows. Subsection 2.4.1 provides the notation used through this section. The cost function that underlies the proposed tra-

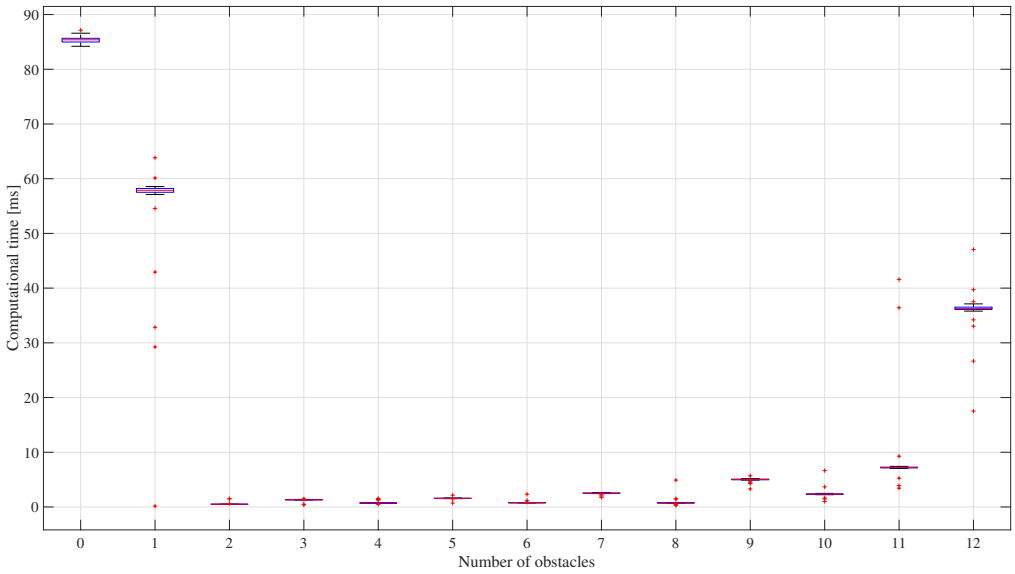


Figure 2.8: Computational time data of the LPA* Algorithm with a tactical parameter set. The average computation time across all obstacle sets is 15.4479ms with a standard deviation of 26.8677ms.

jectory planner is presented in Subsection 2.4.2. The equations of motion employed by the proposed trajectory planner and attitude constraints are shown in Subsection 2.4.3. The boundary conditions are covered by Subsection 2.4.4. Subsection 2.4.5 covers the usage of collision avoidance constraints within the proposed trajectory planner, this topic will be discussed further in Subsection 2.5, where we perform a comparative analysis to find a suitable constraint generation algorithm.

2.4.1 Notation

The system's time is denoted by t and it is assumed that $t \geq 0$. Third-order derivatives of functions with respect to time are denoted by $(\cdot)^{(3)}$. The span of a time interval, $\Delta T > 0$, and number of time intervals, denoted $n_{\mathrm{t}} \in \mathbb{N}$, are both user-defined and generally vary between pairs of consecutive waypoints in the reference path. We assume the UAV is able

to successfully fly between consecutive waypoints. A generic waypoint is denoted $\hat{r}_k$, $k \in \{0, \ldots, n_\mathrm{w} - 1\}$, to$\hat{r}_{k+1}$ in $n_\mathrm{t}\Delta T$ time units.

The *UAV's position* on the interval of $[0, n_\mathrm{t}\Delta T]$ at specific time step is designated $r_k :\rightarrow \mathbb{R}^3 \backslash \mathcal{O}$, $k \in \{0, \ldots, n_\mathrm{w} - 1\}$. The UAV's position is hereby expressed in the inertial reference frame $\mathbb{I}$ with the origin placed at a location conveniently. The *UAV's roll angle* is denoted by $\phi_k : [0, n_\mathrm{t}\Delta T] \rightarrow \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, $k \in \{0, \ldots, n_\mathrm{w} - 1\}$, the *UAV's pitch angle* is denoted by $\theta_k : [0, n_\mathrm{t}\Delta T] \rightarrow \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, the *UAV's yaw angle* is denoted by $\psi_k : [0, n_\mathrm{t}\Delta T] \rightarrow [0, 2\pi)$, the *UAV's velocity* with respect to $\mathbb{I}$ is denoted by $v_k : [0, n_\mathrm{t}\Delta T] \rightarrow \mathbb{R}^3$, the UAV's *angular velocity* with respect to $\mathbb{I}$ is denoted by $\omega_k : [0, n_\mathrm{t}\Delta T] \rightarrow \mathbb{R}^3$. The UAV's *state vector* contains its position, attitude, velocity and angular velocity. As such, it is denoted by $x_k(j\Delta T) \triangleq \left[r_k^\mathrm{T}(j\Delta T), \phi(j\Delta T), \theta(j\Delta T), \psi(j\Delta T), v_k^\mathrm{T}(j\Delta T), \omega_k^\mathrm{T}(j\Delta T)\right]^\mathrm{T}$ for all $j \in \{i, \ldots, n_\mathrm{t}\}$, $i \in \{0, \ldots, n_\mathrm{t}\}$, $k \in \{0, \ldots, n_\mathrm{w} - 1\}$.

2.4.2 Cost Function Definition

The *cost function* underlying the proposed trajectory planner is defined

$$\begin{aligned}\tilde{J}[r_k(0), \psi_k(0), \lambda_k(\cdot)] \triangleq & \sum_{l=k}^{k+\nu_{\mathrm{s},k}-1} \nu_\mathrm{dis}^{l-k} \ell_\mathrm{f}(r_l(n_\mathrm{t}\Delta T), \psi_l(n_\mathrm{t}\Delta T)) \\ & + \sum_{l=k}^{k+\nu_{\mathrm{s},k}-1} \sum_{i=0}^{n_\mathrm{t}-1} \nu_\mathrm{dis}^{l-k} \tilde{\ell}(i\Delta T, r_l(i\Delta T), \psi_l(i\Delta T), \lambda_l(i\Delta T)), \\ & \qquad k \in \{0, \ldots, n_\mathrm{w} - 1\}, \end{aligned} \tag{2.3}$$

where: $\nu_{\text{dis}} \in (0,1)$,

$$\tilde{\ell}(\tau, r_k, \psi_k, \lambda_k) \triangleq \begin{bmatrix} \tilde{r}_k \\ \lambda_k \end{bmatrix}^{\mathrm{T}} \tilde{R}_k(\tau) \begin{bmatrix} \tilde{r}_k \\ \lambda_k \end{bmatrix} + q_\psi(\psi_k - \psi_{\text{f},k})^2 + \tilde{q}_r^{\mathrm{T}} \tilde{r}_k + \tilde{q}_\lambda^{\mathrm{T}} \lambda_k,$$

$$(\tau, r_k, \psi_k, \lambda_k) \in [0, n_{\text{t}} \nu_{\text{s},k} \Delta T] \times \mathbb{R}^3 \times \mathbb{R} \times \mathbb{R}^4, \tag{2.4}$$

$$\ell_{\text{f}}(r_k, \psi_k) \triangleq \left(r_k - \hat{r}_{k+\nu_{\text{s},k}}\right)^{\mathrm{T}} R_{r,\text{f}} \left(r_k - \hat{r}_{k+\nu_{\text{s},k}}\right) + q_\psi(\psi_k - \psi_{\text{f},k})^2, \tag{2.5}$$

$\tilde{R}_k(\tau) \triangleq g_{\text{barrier},k}^{-1}(\tau) \begin{bmatrix} \tilde{R}_r & \tilde{R}_{r,\lambda} \\ \tilde{R}_{r,\lambda}^{\mathrm{T}} & R_\lambda \end{bmatrix}$, $\tilde{R}_r \in \mathbb{R}^{3\times 3}$ is symmetric, $\tilde{R}_{r,\lambda} \in \mathbb{R}^{3\times 4}$, and $R_\lambda \in \mathbb{R}^{4\times 4}$ is positive-definite and defined such that

$$\tilde{R}_r - 2\tilde{R}_{r,\lambda}^{\mathrm{T}} R_\lambda^{-1} \tilde{R}_{r,\lambda} > 0, \tag{2.6}$$

$\tilde{q}_r \in \mathbb{R}^3$, $\tilde{q}_\lambda \in \mathbb{R}^4$,

$$\tilde{r}_k(i\Delta T) \triangleq (1-\mu_{13}) f_{\text{sat}}\left(\mu_{14}(\hat{r}_k - r_{\mathcal{O}_{\text{c},i,k}})\right) \cdot \left[r_k(i\Delta T) - r_{\mathcal{O}_{\text{c},i,k}}\right] + \mu_{13}\left[r_k(i\Delta T) - \hat{r}_{k+\nu_{\text{s},k}}\right],$$

$$i \in \{0, \ldots, n_{\text{t}}\nu_{\text{s},k} - 1\}, \tag{2.7}$$

$\mu_{13} \in (0,1]$ and $\mu_{14} > 0$ are user-defined, $r_k(\cdot)$ denotes the UAV's position, $\lambda_k(\cdot)$ denotes a control input, $f_{\text{sat}}(w) \triangleq \dfrac{\text{sat}\left(\|w\|\right)}{\|w\|}$, $w \in \mathbb{R}^n$, $r_{\mathcal{O}_{\text{c},i,k}} \triangleq \text{d}_2(\hat{r}_k, \mathcal{O}_{\text{c},i,k})$ denotes the Euclidean distance between $\hat{r}_k$ and $\mathcal{O}$, and $R_{r,\text{f}} \in \mathbb{R}^{3\times 3}$ is symmetric and nonnegative-definite, $q_\psi > 0$,

$$g_{\text{barrier},k}(j\Delta T) \triangleq \left[\phi_{\max}^2 - \phi_k^2(j\Delta T)\right]\left[\theta_{\max}^2 - \theta_k^2(j\Delta T)\right] \cdot \prod_{p=1}^{4}[T_{p,k}(j\Delta T) - T_{\min}] \prod_{p=1}^{4}\left[T_{\max} - T_{p,k}(j\Delta T)\right],$$

$$j \in \{i, \ldots, n_{\text{t}}\nu_{\text{s},k}\},\ i \in \{0, \ldots, n_{\text{t}}\nu_{\text{s},k}\}, k \in \{0, \ldots, n_{\text{w}} - 1\}, \tag{2.8}$$

and $\phi_{\max} \in \left(0, \frac{\pi}{2}\right)$, $\theta_{\max} \in \left(0, \frac{\pi}{2}\right)$ are user-defined and denote the maximum roll and pitch angles, respectively, and $0 < T_{\min} < T_{\max}$ are user-defined and capture the minimum and maximum thrust, respectively.

The *barrier function* $g_{\text{barrier}}(\cdot)$ used in the *Lagrangian function* (2.4) is utilized to enforce the nonlinear, time-varying constraints. These constraints set bounds the upper and lower bounds on the UAV's pitch, roll, and the individual thrust delivered by each propeller. As discussed in Section 2.4.4, at the waypoint $\hat{r}_{k+\nu_{\text{s},k}}$, boundary conditions are imposed upon the trajectory planning problem. Therefore, the reference trajectory is only imposed when traversing the waypoints $\hat{r}_k, \ldots, \hat{r}_{k+1}$. The need to outline a reference trajectory that approximates the waypoints $\hat{r}_{k+1}, \ldots, \hat{r}_{k+\nu_{\text{s},k}-1}$ for all $k \in \{0, \ldots, n_{\text{w}} - 1\}$ is captured by the cost function (2.3). In order to reduce the impact of waypoints $\hat{r}_{k+2}, \ldots, \hat{r}_{k+\nu_{\text{s},k}}$, while the UAV approaches $\hat{r}_{k+1}$, the term ν_{dis}^{l-k}, $k \in \{0, \ldots, n_{\text{w}} - 1\}$, in (2.3) is a *discount factor* applied to the objective function.

The proposed trajectory planner requires the UAV's to reach the waypoint $\hat{r}_{k+\nu_{\text{s},k}}$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$ while also pointing the camera, aligned with the vehicle's roll axis, toward the waypoint. The UAV's distance from the waypoint $\hat{r}_{k+\nu_{\text{s},k}}$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$ and distance from the obstacles is captured by the first term and second on the right-hand side of (2.7) respectively.

The UAV's requirements of reaching the next waypoint $\hat{r}_{k+\nu_{\text{s},k}}$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$, coasting the obstacles' set, and pointing the onboard cameras toward $\hat{r}_{k+\nu_{\text{s},k}}$ potentially conflict and as such, are captured by the *Lagrangian function* (2.4). Thereby, allowing for the user to define μ_{13} and μ_{14} such that a reckless or tactical behavior may be expected. Setting $\mu_{13} = 1$, then $\tilde{r}_k(i\Delta T) = r_k(i\Delta T) - \hat{r}_{k+\nu_{\text{s},k}}$, $i \in \{0, \ldots, n_{\text{t}}\nu_{\text{s},k} - 1\}$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$, and additionally, by minimizing (2.3), the user induces a reckless behavior in the UAV, as its only goal becomes reaching the waypoint, and if smaller values of $\mu_{13} \in (0, 1)$ make coasting

the obstacles' set a higher priority. The function $f_{\text{sat}}(\cdot)$ in (2.7) reduces the attractive effect of obstacles at a distance from the waypoint $\hat{r}_k$, $k \in \{0, \dots, n_{\text{w}} - 1\}$, that is larger than μ_{14}^{-1}. Indeed, $f_{\text{sat}}(w) \to 1$ as $\|w\| \to 0$, $f_{\text{sat}}(\mu_{14} w) = 1$ for all $w \in \overline{\mathcal{B}}_{\mu_{14}^{-1}}(0_n)$, $f_{\text{sat}}(\mu_{14} w) < 1$ for all $w \in \mathbb{R}^n \setminus \overline{\mathcal{B}}_{\mu_{14}^{-1}}(0_n)$, and $\lim_{w \to \infty} f_{\text{sat}}(\mu_{14} w) = 0$.

2.4.3 Equations of Motion

The subject of the proposed trajectory planner is the direct control of the UAV's position $r_k(\cdot)$, $k \in \{0, \dots, n_{\text{w}} - 1\}$, along with its yaw angle $\psi_k(\cdot)$. The yaw angle requires specific control via the trajectory planner, as it impacts the ability of the UAV to map an environment with appropriate coverage. The UAV's cameras', which provide pose and point cloud data, have their focal axes aligned with the UAV's roll axis. The *measured output* is $\left[r_k^{\text{T}}(t), \psi_k(t)\right]^{\text{T}}$, $t \in [0, n_{\text{t}} \nu_{\text{s},k} \Delta T]$, $k \in \{0, \dots, n_{\text{w}} - 1\}$, and proceeding as in [34], [35, Prop. 5.1.2], the UAV's output-feedback linearized equations of motion are

$$\dot{\chi}_k(t) = \tilde{A} \chi_k(t) + \tilde{B} \lambda_k(t), \qquad t \in [0, n_{\text{t}} \nu_{\text{s},k} \Delta T], \tag{2.9}$$

where $\chi_k(t) \triangleq \left[r_k^{\text{T}}(t), \dot{r}_k^{\text{T}}(t), \ddot{r}_k^{\text{T}}(t), r_k^{(3)\text{T}}(t), \psi_k(t), \dot{\psi}_k(t)\right]^{\text{T}}$, *virtual control input* is denoted $\lambda_k \in \mathbb{R}^4$, $\tilde{A} \triangleq \text{blockdiag}(\tilde{A}_r, \tilde{A}_\psi)$, $\tilde{B} \triangleq \left[\tilde{B}_r^{\text{T}}, \tilde{B}_\psi^{\text{T}}\right]^{\text{T}}$,

$$\tilde{A}_r \triangleq \begin{bmatrix} 0_{3\times3} & \mathbf{1}_3 & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & \mathbf{1}_3 & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & \mathbf{1}_3 \\ A_{r,0} & A_{r,1} & A_{r,2} & A_{r,3} \end{bmatrix} \in \mathbb{R}^{12\times12}, \tag{2.10}$$

$$\tilde{A}_\psi \triangleq \begin{bmatrix} 0 & 1 \\ A_{\psi,0} & A_{\psi,1} \end{bmatrix} \in \mathbb{R}^{2\times2}, \tag{2.11}$$

are Hurwitz, $A_{r,i} \in \mathbb{R}^{3\times3}$, $i \in \{0,\ldots,3\}$, $A_{\psi,0} \in \mathbb{R}$, and $A_{\psi,1} \in \mathbb{R}$ are user-defined,

$$\tilde{B}_r \triangleq \begin{bmatrix} 0_{9\times4} \\ B_r \end{bmatrix} \in \mathbb{R}^{12\times4}, \qquad \tilde{B}_\psi \triangleq \begin{bmatrix} 0 \\ B_\psi \end{bmatrix} \in \mathbb{R}^{2\times4} \tag{2.12}$$

and the pairs $(\tilde{A}_r, \tilde{B}_r)$ and $(\tilde{A}_\psi, \tilde{B}_\psi)$ are controllable. Using (2.9), the discrete-time, linearized, zero-order hold [36], output-feedback linearized equations of motion of a quadcopter UAV are defined:

$$\begin{aligned} \chi_k((j+1)\Delta T) &= A\chi_k(j\Delta T) + B\lambda_k(j\Delta T), \\ j \in \{i,\ldots,n_{\mathrm{t}}\nu_{\mathrm{s},k}-1\}, &\quad i \in \{0,\ldots,n_{\mathrm{t}}\nu_{\mathrm{s},k}-1\}, \\ &k \in \{0,\ldots,n_{\mathrm{w}}-1\}, \end{aligned} \tag{2.13}$$

where $A = e^{\tilde{A}\Delta T}$ and $B = \int_0^{\Delta T} e^{\tilde{A}\sigma}\mathrm{d}\sigma\tilde{B}$.

Therefore, the equality constraint for the MPC algorithm employed to outline tactical reference trajectories for UAVs is given by (2.13).

The *total thrust force* produced by the UAV's propellers is defined as $u_{1,k}(\cdot)$, $k \in \{0,\ldots,n_{\mathrm{w}}-1\}$, such that $u_{1,k}(j\Delta T) = [1,0]\,\delta_k(j\Delta T)$, $j \in \mathbb{N}$, and

$$\delta_k((j+1)\Delta T) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \delta_k(j\Delta T) + \frac{1}{2}\begin{bmatrix} \Delta T^2 \\ 2\Delta T \end{bmatrix} \eta_{1,k}(j\Delta T). \tag{2.14}$$

Given $\lambda_k(j\Delta T)$, $j \in \{i,\ldots,n_{\mathrm{t}}\nu_{\mathrm{s},k}-1\}$, $i \in \{0,\ldots,n_{\mathrm{t}}\nu_{\mathrm{s},k}-1\}$, $k \in \{0,\ldots,n_{\mathrm{w}}-1\}$, the total thrust force's virtual control input is denoted by $\eta_{1,k}(j\Delta T)$, and the roll, pitch, and yaw moment as produced by the UAV's propellors is denoted by $u_{2,k}(j\Delta T)$, $u_{3,k}(j\Delta T)$, and $u_{4,k}(j\Delta T)$ respectively. The control inputs $[\eta_{1,k}(j\Delta T), u_{2,k}(j\Delta T), u_{3,k}(j\Delta T), u_{4,k}(j\Delta T)]$ are

computed as

$$[\eta_{1,k}(j\Delta T), u_{2,k}(j\Delta T), u_{3,k}(j\Delta T), u_{4,k}(j\Delta T)]^{\mathrm{T}} = \zeta_k(j\Delta T), \tag{2.15}$$

where

$$\begin{aligned}\zeta_k \triangleq G(r_k, \phi_k, \theta_k, \psi_k, \omega_k, u_{1,k})\Bigg(&- f(r_k, \phi_k, \theta_k, \psi_k, \omega_k, u_{1,k}) \\ &+ \begin{bmatrix} A_{r,0} r_k(t) + A_{r,1}\dot{r}_k(t) + A_{r,2}\ddot{r}_k(t) + A_{r,3} r_k^{(3)}(t) \\ A_{\psi,0}\psi_k(t) + A_{\psi,1}\dot{\psi}_k(t) \end{bmatrix} + \begin{bmatrix} B_r \\ B_\psi \end{bmatrix} \lambda_k \Bigg),\end{aligned} \tag{2.16}$$

for all $(r_k, \phi_k, \theta_k, \psi_k, \omega_k, u_{1,k}, \lambda_k) \in \mathbb{R}^3 \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times [0, 2\pi) \times \mathbb{R}^3 \times \mathbb{R} \times \mathbb{R}^4$. The expressions for $f(\cdot)$ and $G(\cdot)$ are omitted for brevity and can be found by applying Proposition 5.1.2 of [35].

2.4.4 Boundary Conditions

To ensure feasible trajectories, the boundary conditions imposed on the UAV's translational dynamic equations (2.9) and (2.13), are given in Table 2.1. For successful initialization of the MPC algorithm, conditions on the UAV's initial position and acceleratio, $r_{\mathrm{init},k}$ and $\ddot{r}_{\mathrm{init},k}$, ensure that the UAV's reference position and acceleration are equal to the UAV's position and acceleration at time of initialization. In order to ensure that the reference trajectory traverses the next waypoint $\hat{r}_{k+\nu_{\mathrm{s},k}}$, $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$, the conditions denoted $r_{\mathrm{end},k}$ is imposed to constrain the UAV's final position. The condition on the UAV's final acceleration are defined so that

$$\hat{a}_{k+\nu_{\mathrm{s},k}} \triangleq \hat{a}\left(\mathrm{d}_2(\hat{r}_{k+\nu_{\mathrm{s},k}}, \mathcal{O})\right)\hat{d}_k, \qquad k \in \{0, \ldots, n_{\mathrm{w}} - 1\}, \tag{2.17}$$

Table 2.1: Parameters needed to define the boundary conditions for position and yaw control of the proposed trajectory planner for $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$.

$i = 0$	$r_{\mathrm{init},k} = r_k(n_{\mathrm{t}}\nu_{\mathrm{s},k}\Delta T)$, $\ddot{r}_{\mathrm{init},k} = \ddot{r}_k(n_{\mathrm{t}}\nu_{\mathrm{s},k}\Delta T)$, $r_{\mathrm{end},k} = \hat{r}_{k+\nu_{\mathrm{s},k}}$, $\ddot{r}_{\mathrm{end},k} = \hat{a}_{k+\nu_{\mathrm{s},k}}$.
$i \in \{1, \ldots, n_{\mathrm{t}}\nu_{\mathrm{s},k} - 1\}$	$r_{\mathrm{init},k} = r_k(i\Delta T)$, $\ddot{r}_{\mathrm{init},k} = \ddot{r}_k(i\Delta T)$, $r_{\mathrm{end},k} = \hat{r}_{k+\nu_{\mathrm{s},k}}$, $\ddot{r}_{\mathrm{end},k} = \hat{a}_{k+\nu_{\mathrm{s},k}}$.

where

$$\hat{a}(\alpha) \triangleq \begin{cases} 0, & \text{if } \alpha \in [0, \mu_{15}], \\ \frac{\mu_{17}}{\mu_{16}-\mu_{15}}(\alpha - \mu_{15}), & \text{if } \alpha \in [\mu_{15}, \leq \mu_{16}], \\ \mu_{17}, & \text{otherwise}, \end{cases} \tag{2.18}$$

$$\hat{d}_k \triangleq \mu_{18}\frac{\hat{r}_{k+\nu_{\mathrm{s},k}} - \hat{r}_{k+\nu_{\mathrm{s},k}-1}}{\|\hat{r}_{k+\nu_{\mathrm{s},k}} - \hat{r}_{k+\nu_{\mathrm{s},k}-1}\|} + (1 - \mu_{18})\frac{\hat{r}_{k+\nu_{\mathrm{s},k}+1} - \hat{r}_{k+\nu_{\mathrm{s},k}}}{\|\hat{r}_{k+\nu_{\mathrm{s},k}+1} - \hat{r}_{k+\nu_{\mathrm{s},k}}\|}, \tag{2.19}$$

and $\mu_{15}, \mu_{16}, \mu_{17} > 0$ and $\mu_{18} \in [0, 1]$ are user-defined.

If the UAV is sheltered by its proximity to an obstacle and therefore operating in a safer area, then it is desirable for the UAV not to accelerate. The UAV is considered to be within safe proximity at a distance smaller than μ_{15} such that $\mathrm{d}_2(\hat{r}_{k+\nu_{\mathrm{s},k}}, \mathcal{O}) \leq \mu_{15}$, $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$, then $\|\ddot{r}_{\mathrm{end},k}\| = 0$. Alternatively, if the UAV is operating outside the proximity of shelter provided by an obstacle, it is desirable for it to rapidly traverse its environment. Therefore if $\mathrm{d}_2(\hat{r}_{k+\nu_{s,k}}, \mathcal{O}) > \mu_{16}$, $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$, then $\|\ddot{r}_{\mathrm{end},k}\| = \mu_{17}$ so that the UAV will move through dangerous areas more quickly. Finally, when $\mathrm{d}_2(\hat{r}_{k+\nu_{\mathrm{s},k}}, \mathcal{O}) \in [\mu_{15}, \mu_{16}]$, $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$, the condition $\|\ddot{r}_{\mathrm{end},k}\|$ increases as a linear function of $\mathrm{d}_2(\hat{r}_{k+\nu_{\mathrm{s},k}}, \mathcal{O})$ from 0 to μ_{17}.

For both values of $\mu_{18} = 1$ and $\mu_{18} = 0$, then $\ddot{r}_{\text{end},k}$, $k \in \{0, \ldots, n_{\text{w}} - 1\}$, is set to point in the direction joining the waypoint $\hat{r}_{k+\nu_{\text{s},k}}$, which is set as the endpoint for the trajectory, and $\hat{r}_{k+\nu_{\text{s},k}-1}$ and $\hat{r}_{k+\nu_{\text{s},k}+1}$, respectively. Setting $\mu_{18} = 1$ allows the proposed trajectory planner set the endpoint acceleration condition such that the trajectory planner's path is an interpolation of all waypoints. Similarly, setting $\mu_{18} = 0$ also allows the trajectory planner to set the endpoint acceleration condition such that it generates a reference path based on the interpolation of all waypoints, however, this option begins the interpolation of all waypoints after $\hat{r}_{k+\nu_{\text{s},k}}$ has been reached. For brevity, and relevance, initial conditions at the beginning of the mission and the endpoint conditions at the end of the mission are not included in Table 2.1.

2.4.5 Collision Avoidance Constraints

The UAV, and an obstacle free space with sufficient room for the trajectory planner to generate viable reference trajectories, is captured by the affine constraint set

$$F_{r,k}(i\Delta T) r_k(i\Delta T) \leq\leq f_{r,k}(i\Delta T), \qquad i \in \{0, \ldots, n_{\text{t}}\nu_{\text{s},k}\}, \qquad k \in \{0, \ldots, n_{\text{w}} - 1\}, \quad (2.20)$$

where the $F_{r,k} : \mathbb{R} \to \mathbb{R}^{n_{\text{coll}} \times 3}$ is such that

$$\mathbf{e}_{q,n_{\text{coll}}}^{\text{T}} F_{r,k}(i\Delta T) \triangleq \begin{bmatrix} \cos\left(\mu_9 + 2\mu_9(q-1)\right)\cos\left(\mu_{10} + 2\mu_{10}(q-1)\right) \\ \sin\left(\mu_9 + 2\mu_9(q-1)\right)\cos\left(\mu_{10} + 2\mu_{10}(q-1)\right) \\ (-1)^q \sin\left(\mu_{10} + 2\mu_{10}(q-1)\right) \end{bmatrix}^{\text{T}},$$

$$q \in \{1, \ldots, n_{\text{coll}}\}, \quad (2.21)$$

$\mu_9 > 0$ and $\mu_{10} > 0$ are user-defined and such that $n_{\text{coll}} \triangleq 16\pi^2/(\mu_9\mu_{10})$ is an even integer and denotes the number of collision avoidance constraints, $f_{r,k} : \mathbb{R} \to \mathbb{R}^{n_{\text{coll}}}$ is such that

$$\mathbf{e}_{q,n_{\text{coll}}}^{\mathrm{T}} f_{r,k}(i\Delta T) \triangleq \min \left\{ \min_{p \in \mathcal{S}_q \cap \mathcal{O}_{\text{c},i,k}} \mathbf{e}_{q,n_{\text{coll}}}^{\mathrm{T}} F_{r,k}(i\Delta T)p, \right. \\ \left. \min_{s \in \{1,\ldots,n_{\text{coll}}\} \setminus \{q\}} \mathbf{e}_{q,n_{\text{coll}}}^{\mathrm{T}} F_{r,k}(i\Delta T)p_s(i\Delta T) \right\}, \tag{2.22}$$

$\mathcal{S}_q \subset \mathbb{R}^3$ is a right square spherical pyramid, whose apex is centered in the UAV, whose bisector is given by (2.21), and whose sides have user-defined length,

$$p_s(i\Delta T) \triangleq \operatorname*{arg\,min}_{p \in \mathcal{S}_s \cap \mathcal{O}_{\text{c},i,k}} F_{r,k}^{\mathrm{T}}(i\Delta T)\mathbf{e}_{s,n_{\text{coll}}}\mathbf{e}_{s,n_{\text{coll}}}^{\mathrm{T}} F_{r,k}(i\Delta T)p, \qquad s \in \{1,\ldots,n_{\text{coll}}\}, \tag{2.23}$$

and $\mathcal{O}_{\text{c},i,k} \subseteq \mathcal{O}$ captures a set of obstacles' points that are more likely to be impacted by the UAV and therefore relevant to the generation of reference trajectories. In practice, the convex constraint set with affine boundaries captured by (2.20) is produced by partitioning a sphere centered in the UAV in n_{coll} right square spherical pyramids, and constructing hyperplanes orthogonal to the bisector of each pyramid [37]. The use of the affine inequalities (2.20) to capture collision avoidance constraints is essential to guarantee fast numerical solutions of the proposed trajectory planning problem. The generation of the collision avoidance constraint set is the subject of the experiment discussed in Section 2.5.

2.5 Constraint Generation

Reference paths, such as those produced by the algorithms presented in Section 2.3, are sequences of waypoints that the vehicle needs to traverse. However, reference paths do not provide any information on the speed and acceleration the vehicle is supposed to have while interpolating these waypoints. To design reference trajectories, that is, curves parameterized

by time that interpolate a reference path's waypoints and are compatible with the vehicle's dynamics, a trajectory planner is required. One of the key problems in trajectory planning is to assure that the reference trajectory does not intersect any obstacle. For this reason, in this section, we discuss the problem of separating the obstacle-free space, where the reference trajectory may lie, from obstacles.

This section is outlined as follows. Section 2.5.1 describes the basic premise of constraint generation, and Section 2.5.2 provides specifics of four algorithms that are compared. The results of the comparison are shown in Section 2.5.3.

2.5.1 A Brief Overview

The problem of separating obstacles from obstacle-free spaces has been explored abundantly in the literature [14, 18, 19, 38, 39, 40]. In general, the techniques employed in this vast topic can be grouped into two broad categories, namely those that aim to bound the set of constraints and exclude the obstacle-free space, and those that aim to bound the obstacle-free space and, hence, exclude obstacles. These two classes of techniques are clearly complementary to one another. The former approach is more appropriate for techniques, such as potential field methods [41], which do not have special requirements on the set where the reference trajectory is to be found. The latter approach, instead, is more suitable for techniques, such as model predictive control, where the geometric properties of the search space do impact the ability to find reference trajectories. In this thesis, we employ a trajectory tracking technique based on the linear-quadratic model predictive control. Thus, the constraints on the search space must be affine in the state vector, and bounding the search space by means of hyperplanes forming a convex search space is a necessity.

In this section, we consider four state-of-the-art techniques that allow finding convex search

spaces bounded by hyperplanes. Specifically, we consider: Safe Flight Corridor (SFC)[2], Iterative Regional Inflation by Semi-Definite Programming (IRIS)[3], Minimum Distance Constraint Algorithm (MDCA)[11], and the Bubble-Bath algorithm[42] [1].

2.5.2 Theoretical Formulation

In this section, we present four state-of-the-art techniques to find convex subsets of the obstacle-free space.

MDCA Constraint Generation. The MDCA algorithm was first presented in [11] as a novel solution for the problem of finding the free space surrounding a UAV's flight path with far less computational time than the SFC and IRIS algorithms. The algorithm begins by generating through a semi-definite programming algorithm, an ellipsoid about a point on the path and of a size such that it touches at least one point of a nearby obstacle. A quantity, N, of points are sampled from the surface of the ellipsoid from which N hyperplanes are calculated tangent to the ellipsoid. These ellipsoids contain an obstacle-free volume and the UAV. If the next waypoint of the path is not contained within the generated polyhedron, such as points located around the corner of the wall, an intermediate point is taken through a ray tracing technique. As the UAV approaches the intermediate point, the algorithm recomputes hyperplanes such that the goal point is now contained. A schematic representation of MDCA is shown in Figure 2.9.

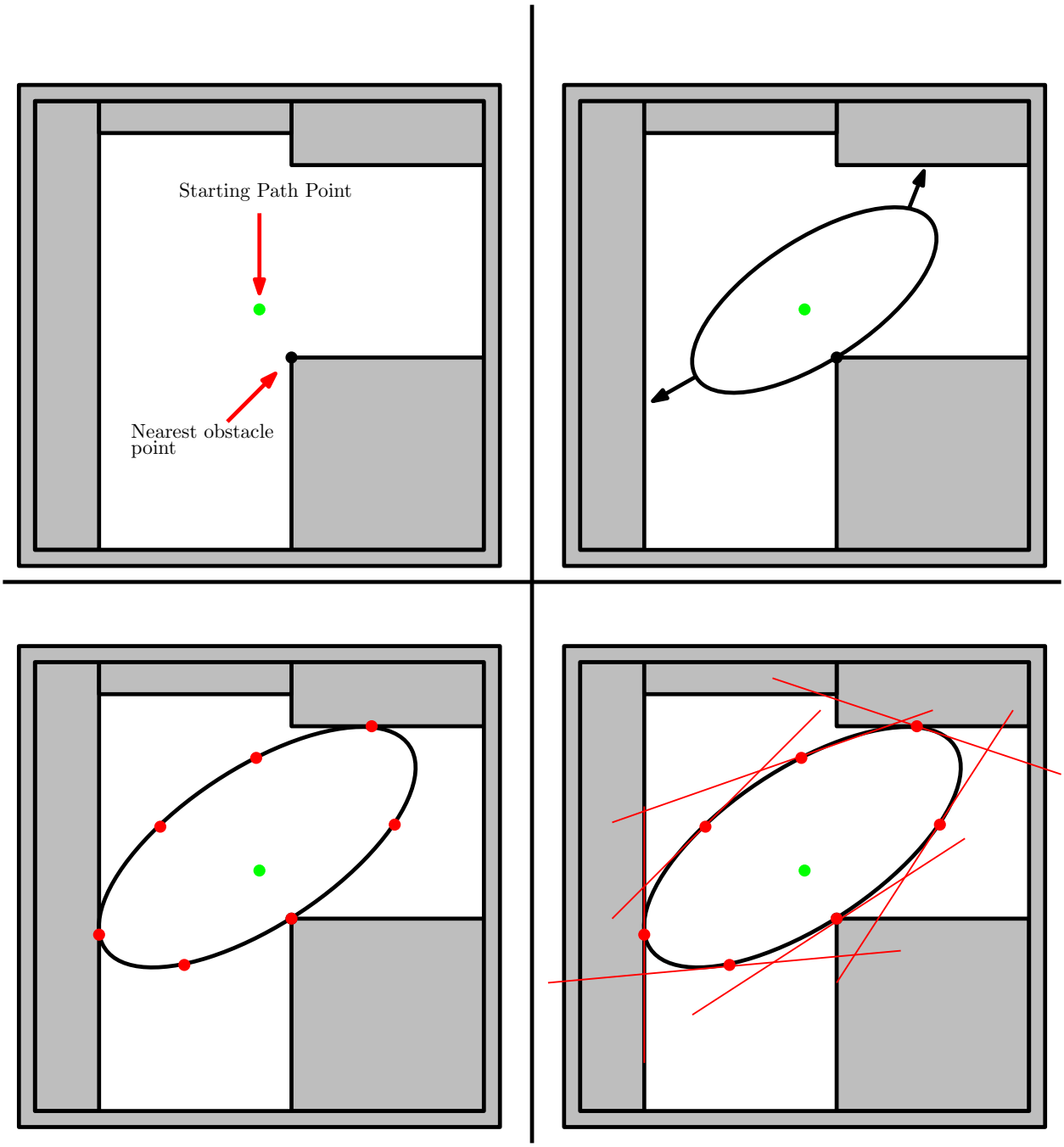


Figure 2.9: This example provides a 2-D example of the MDCA algorithm. Steps are presented left-to-right, top-to-bottom. Step 1: An obstacle point nearby the path point is identified. Step 2: The ellipsoid, tangent to the previously identified obstacle point, is stretched such that it is tangent to other nearby obstacles point. Step 3: $N = 7$ points are randomly sampled from the surface of the ellipsoid. Step 4: The final constraint set is calculated as the planes tangent to the randomly sampled points and the surface of the ellipsoid.

Bubble-bath Constraint Generation. The Bubble-bath algorithm, initially presented in [1], expands the process used by MDCA in order to increase the unoccupied voxels captured within the volume defined by a convex set of hyperplanes. To reduce computational load, only points contained within a sphere of user defined radius and centered at the UAV's current position, the half-space of the UAV's velocity, and accessible to the UAV are considered by this algorithm.

An initial, obstacle-free ellipsoid, denoted as the *parent ellipsoid* is generated such that it is centered about the UAV's current position. This ellipsoid is expanded until it coincides with a single obstacle within the obstacles set. Additional ellipsoids, denoted as *children ellipsoids* are generated such that their origin is located on the surface of the parent ellipsoid. These ellipsoids are also expanded until they intersect an obstacle contained by the UAV's obstacle set. These additional child ellipsoids capture the unoccupied voxels not captured by the parent ellipsoid. It is expected that this algorithm will take slightly more computational time and generate larger volumes than MDCA.

The set of occupied point within an environment is denoted by $\mathcal{O}$. In the bubble bath method, $\mathcal{O}_{\mathrm{c},i,k}$ denotes a subset of $\mathcal{O}$ that are the points most accessible to the UAV. The closed set $\overline{\mathcal{H}}_\rho(r_k(i\Delta T))$, denotes all obstacles points that are within a certain distance, denoted by ρ from the UAV's position at a given time, at $t = i\Delta T$, and are contained in the closed half-space determined by the UAV's velocity vector. We define the closed set as

$$\overline{\mathcal{H}}_\rho(r_k(i\Delta T)) \triangleq \left\{x \in \overline{\mathcal{B}}_\rho(r_k(i\Delta T)) \cap \mathcal{O} : x^{\mathrm{T}} v_k(i\Delta T) \geq 0\right\},$$
$$i \in \{0, \ldots, n_{\mathrm{t}}\nu_{\mathrm{s},k}\}, \qquad k \in \{0, \ldots, n_{\mathrm{w}} - 1\}, \tag{2.24}$$

where $\rho \triangleq \max\{\|v_k(i\Delta T)\|, \overline{v}\} i\Delta T$ and $\overline{v} > 0$ is user-defined. Increasing the UAV's velocity thereby increases the size of ρ, which in turn increases the size of $\mathcal{O}_{\mathrm{c},i,k}$. There exists a

user-defined safety margin, denoted by, $\overline{v}$, which provides an upper bound to ρ. Having defined the set of obstacle points that we are now most concerned with, can now compute the closed *parent ellipsoid* as $\overline{\mathcal{E}}\left(P_k(i\Delta T), r_k(i\Delta T), c_k(i\Delta T)/\mu_{11}\right)$, $i \in \{0, \ldots, n_{\mathrm{t}}\nu_{\mathrm{s},k}\}$, $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$. In the computation of $\overline{\mathcal{E}}$, $P_k(i\Delta T) \in \mathbb{R}^{3\times 3}$ and $c_k(i\Delta T) \in \mathbb{R}$ are solutions to the quadratic programming problem in which the cost function is defined

$$\min c_k(i\Delta T) \tag{2.25}$$

and the constraints of the problem are given by

$$g_1(P_k(i\Delta T), r_k(i\Delta T), w_\alpha, c_k(i\Delta T)) \geq 0, \qquad \alpha \in \{1, \ldots, n_{\mathrm{UAV}}\}, \tag{2.26a}$$

$$g_2(P_k(i\Delta T), r_k(i\Delta T), w_\beta, c_k(i\Delta T)) \leq 0, \qquad \beta \in \{1, \ldots, n_{\mathcal{H}}\}, \tag{2.26b}$$

$$g_3(P_k(i\Delta T)) \leq 0_{3\times 3}, \tag{2.26c}$$

$$g_4(c_k(i\Delta T)) < 0, \tag{2.26d}$$

The equations used in the constraints are as follows

$$g_1(P, r, w, c) \triangleq (w - r)^{\mathrm{T}} P (w - r) - c, \tag{2.27a}$$

$$g_2(P, r, w, c) \triangleq (w - r)^{\mathrm{T}} P (w - r) - c \tag{2.27b}$$

$$g_3(P) \triangleq P + \mathbf{1}_3 \tag{2.27c}$$

$$g_4(c) \triangleq c \tag{2.27d}$$

where $(P, r, w, c) \in \mathbb{R}^{3\times 3} \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}$, $\mu_{11} > 1$ is a user-defined safety margin, and $n_{\mathcal{H}}$ denotes the cardinality of the set $\overline{\mathcal{H}}_\rho(r_k(i\Delta T))$. The conditions imposed by (2.26c) and (2.26d) ensure that $\overline{\mathcal{E}}\left(P_k(i\Delta T), r_k(i\Delta T), c_k(i\Delta T)/\mu_{11}\right)$ is an ellipsoid. While (2.26a) and (2.26b) are employed to guarantee that the parent ellipsoid includes all the points occupied

by the UAV, denoted w_α, $\alpha \in \{1, \ldots, n_{\text{UAV}}\}$, while excluding all points contained within $\overline{\mathcal{H}}_\rho(r_k(i\Delta T))$. Having computing the parent ellipsoid, there may still exist unoccupied points within the vicinity defined by ρ. Therefore, for all $i \in \{0, \ldots, n_{\text{t}}\nu_{\text{s},k}\}$ and $k \in \{0, \ldots, n_{\text{w}}-1\}$, we compute ℓ closed *children ellipsoids*, denoted by $\overline{\mathcal{E}}(\tilde{P}_{k,l}(i\Delta T), \tilde{r}_{k,l}(i\Delta T), \tilde{c}_{k,l}(i\Delta T))$, $l \in \{1, \ldots, \ell\}$. The variable ℓ is user-defined, while $\tilde{P}_{k,l}(i\Delta T) \in \mathbb{R}^{3\times 3}$ and $\tilde{c}_{k,l}(i\Delta T) \in \mathbb{R}$ are solutions of the quadratic programming problem with cost function defined as

$$\min \tilde{c}_{k,l}(i\Delta T) \tag{2.28}$$

and constraints given by (2.26b)–(2.26d) with $P_k(\cdot)$ replaced by $\tilde{P}_{k,l}(\cdot)$, $r_k(\cdot)$ replaced by $\tilde{r}_{k,l}(\cdot)$,

$$\tilde{r}_{k,l}(i\Delta T) \triangleq \sqrt{\frac{c_k(i\Delta T)P_k^{-1}(i\Delta T)}{\mu_{11}}} \begin{bmatrix} \cos\gamma_l \cos\eta_l \\ \sin\gamma_l \cos\eta_l \\ \sin\eta_l \end{bmatrix} + r_k(i\Delta T), \tag{2.29}$$

$c_k(\cdot)$ replaced by $\tilde{c}_{k,l}(\cdot)$, and $\gamma_l, \eta_l \in \mathbb{R}$ are user-defined. These constraints ensure that the children ellipsoids do not contain the obstacle points captured in $\overline{\mathcal{H}}_\rho(r_k(i\Delta T))$ and, specifically due to (2.29), are centered on the boundary of the parent ellipsoid. Finally, we define the subset of $\mathcal{O}$ of interest

$$\begin{aligned} \mathcal{O}_{\text{c},i,k} \triangleq \{r \in \overline{\mathcal{H}}_\rho(r_k(i\Delta T)) : \|c_k^{-1/2}(i\Delta T)P_k^{1/2}(i\Delta T)(r - r_k(i\Delta T))\| \leq 1 + \mu_{12}\} \\ \cup_{l=1}^{\ell} \{r \in \overline{\mathcal{H}}_\rho(r_k(i\Delta T)) : \|\tilde{c}_{k,l}^{-1/2}(i\Delta T)\tilde{P}_{k,l}^{1/2}(i\Delta T)(r - \tilde{r}_{k,l}(i\Delta T))\| \leq 1 + \mu_{12}\}, \\ i \in \{0, \ldots, n_{\text{t}}\nu_{\text{s},k}\}, \qquad k \in \{0, \ldots, n_{\text{w}} - 1\}, \end{aligned} \tag{2.30}$$

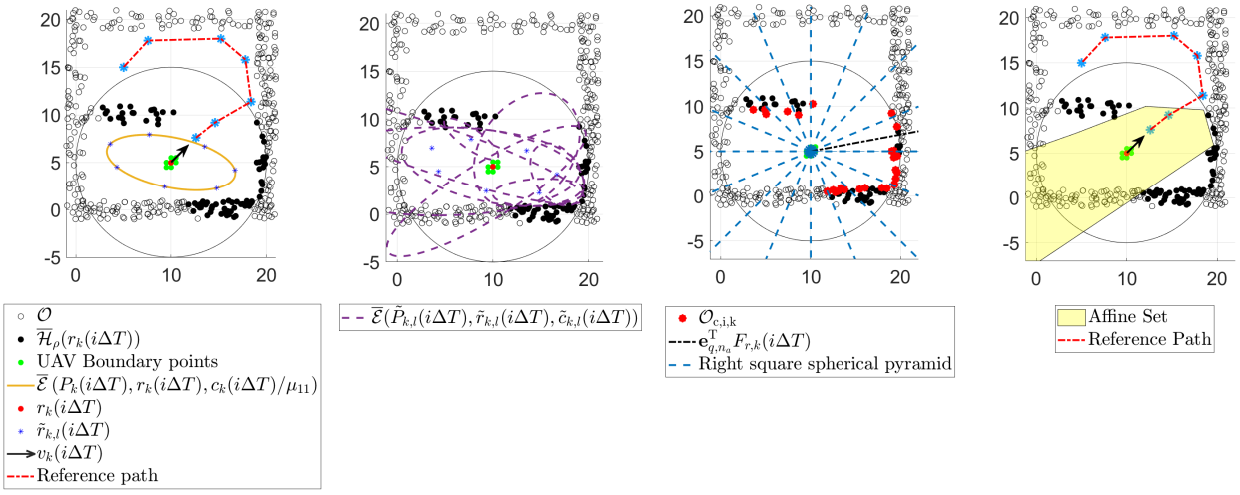


Figure 2.10: A 2-D representation of the Bubble-Bath algorithm as used to generate affine constraint sets from the set of occupied points $\mathcal{O}$, originally presented in [1]. Steps are presented left-to-right. Step 1: the parent ellipsoid, which contains the UAV and excludes all obstacle points, is created. Step 2: the children ellipsoids are generated such that they are centered upon the boundary of the parent ellipsoid. Step 3: a subset of $\mathcal{O}$ is computed and denoted $\mathcal{O}_{\mathrm{c},i,k}$ such that these occupied points exist within a user-defined distance from the boundary of the volume captured by the parent and children ellipsoids. Step 4: the affine set is computed as the subset of the state-space that contains the UAV and excludes the entirety of $\mathcal{O}_{\mathrm{c},i,k}$.

where we denote a user-defined safety margin as $\mu_{12} > 0$. In practice, the subset of obstacle points $\mathcal{O}_{\mathrm{c},i,k}$ is given by the set of all points that are within a distance smaller than or equal to μ_{12} from any of the children ellipsoids or $\overline{\mathcal{E}}\left(P_k(i\Delta T), r_k(i\Delta T), c_k(i\Delta T)\right)$; It is worth noting that the conditions imposed by (2.26b) mean that $\partial\overline{\mathcal{E}}\left(P_k(i\Delta T), r_k(i\Delta T), c_k(i\Delta T)\right)$ comprises at least one point of the obstacles' set $\mathcal{O}$.

This technique is referred to as the *bubble bath method* because the emergence of children ellipsoids from the parent ellipsoid, for the purpose of capturing unoccupied points not captured by the parent ellipsoid, mimic the behavior of bubbles in a bathtub. As the number of children ellipsoids increases, this technique becomes more accurate at identifying which occupied points have the potential to impact the UAV.

Figure 2.10 provides a graphical representation of the bubble bath algorithm's approach to identifying the subset $\mathcal{O}_{\mathrm{c},i,k}$ of the obstacles' set $\mathcal{O}$ for some $i \in \{0, \ldots, n_{\mathrm{t}}\nu_{\mathrm{s},k}\}$ and $k \in \{0, \ldots, n_{\mathrm{w}} - 1\}$, and thereby creating an affine constraint set.

SFC Constraint Generation. The SFC algorithm [2] generates a series of hyperplanes that form a set of intersecting, convex polyhedra that contain a volume within which a UAV may safely traverse. The flight corridor is generated with the following procedure for a given pair of points $(\hat{r}_k, \hat{r}_{k+1})$ contained by the path calculated for the UAV to follow. Let L_i denote the line segment joining $\hat{r}_k$ and $\hat{r}_{k+1}$. The ellipsoid,

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{2.31}$$

where $a, b, c > 0$, devoid of obstacles is found. The x-axis of the ellipsoid is co-linear with $L_i \in \mathbb{R}^3$ and the origin of the ellipsoid is located at the mid-point of L_i. Initially, we set $a = b = c = \frac{|L_i|}{2}$ so that the ellipsoid reduces to a sphere. The semi-axes lengths b and c are then stretched iteratively until the ellipsoid touches the nearest obstacle to the path, denoted p^*. A polyhedron is derived from the previously found ellipsoid. Initially, the intersection of the unaltered ellipsoid and the corresponding obstacle is used to define the first hyperplane. The ellipsoid is then dilated, while maintaining its aspect ratio, and further intersections of the now stretched ellipsoid and other obstacles define additional hyperplanes. To reduce the computational load, only obstacles local to the path are considered. This is done via a *bounding box* which is cubic and centered at the mid-point of L_i such that only obstacles within this bounding box are considered by the SFC algorithm. The resulting polyhedra constructed by the previously found hyperplanes is shrunk down so that a buffer between the edge of the obstacle free volume and the obstacles exists. This is done by modeling the UAV as a sphere of radius r_{UAV} and then moving the hyperplanes along their normal axis a distance of r_{UAV}, thereby guaranteeing that the UAV will always operate in a volume that

is a safe distance from the walls of its environment. An example of this is shown in Figure 2.11.

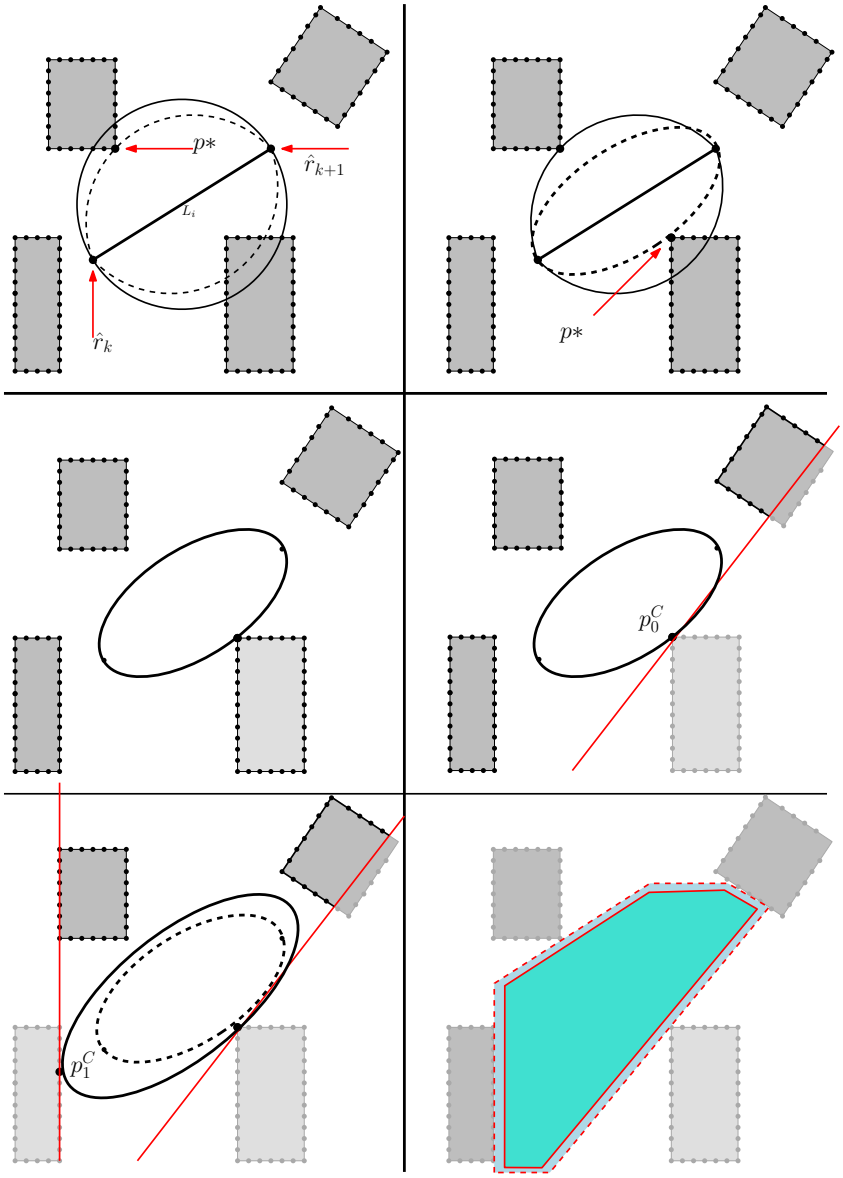


Figure 2.11: A 2-D example of the SFC algorithm adapted from [2]. Steps are presented left-to-right, top-to-bottom. Step 1: An initial sphere, free of obstacles and located at the midpoint of L_I, is expanded until it intersects an occupied point. Step 2: The resulting ellipsoid is shrunk as occupied points closer to the path are found. Step 3: An ellipsoid in contact with the occupied point nearest the path is determined. Step 4: A hyperplane is taken tangent to the ellipsoid and occupied point nearest the path. Step 5: Additional hyperplanes are taken as the ellipsoid is stretch to intersect other obstacle points. Step 6: The resulting polyhedron is then scaled down to accommodate the width of the vehicle.

IRIS Constraint Generation. While SFC generates a set of hyperplanes about the midpoint between two points on the path, IRIS, initially presented in [3], generates a set of constraint planes about each point on the sequence of waypoints comprising the UAV's path. Initially, an ellipsoid with equal semi-axes lengths is generated about a point on the path. The obstacles surrounding the ellipsoid are used to create hyperplanes that intersect the obstacle and are tangent to the ellipsoid. This set of hyperplanes, represented as a set of linear constraints, becomes a polyhedron of free space. A new ellipsoid is generated such that it now maximizes the space contained by the set of hyperplanes. This process is repeated until the change of the volume of the ellipsoid in each cycle falls below a certain threshold. It is guaranteed that between iterations, the volume of the free space contained by the polyhedron does not decrease but it is possible for the initial point about which the algorithm operates to no longer be captured within the polyhedron. An example of this is shown in Figure 2.12.

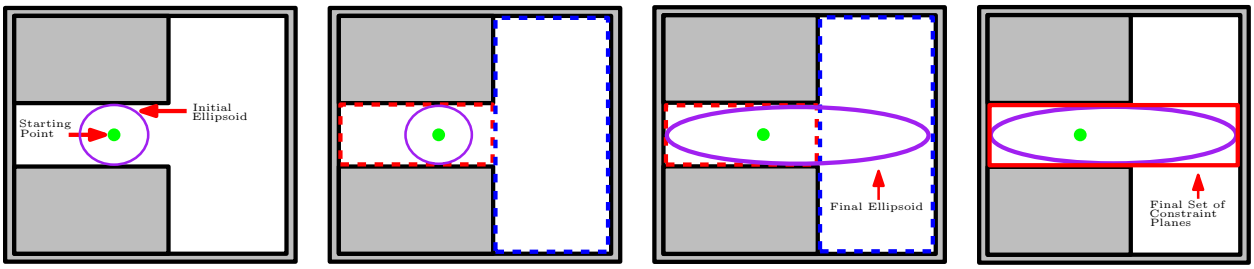


Figure 2.12: This 2-D example of how the IRIS algorithm operates is adapted from [3]. Steps are presented left-to-right, top-to-bottom. Step 1: An initial sphere, free of obstacles and located at a point along a path is formed. Step 2: A set of linear constraints(shown in red) are taken tangent to the initial sphere and the surrounding obstacles. Step 3: The ellipsoid is stretched such that it intersects the initial linear constraint set and then continues to expand until it intersects the adjacent constraint set. In this step, the final ellipsoid is found. Step 4: The final constraint set is taken tangent to the local obstacle set and the final ellipsoid.

2.5.3 Constraint Generation Performance Tests and Results

In this section, we compare the performance of MDCA [11], the Bubble-Bath algorithm [1, 42], SFC [2], and IRIS [3], and asses their performance within the context of a trajectory planner for UAVs employed in tactical missions. A single obstacle set is employed, shown in Figure 2.13, modeled to emulate a two-story house within which occupied voxels that capture shapes such as walls and stairs. Ten paths are manually selected to reflect possible flight paths of the UAV. The paths, shown in Figure 2.14, are comprised of adjacent voxels and vary in length from 175 to 400 voxels long in increments of 25 voxels. Each path converges and shares the approximately 150 path-points leading to a common endpoint. The four algorithms being tested are set up to output the constraint planes generated at each point of a given path. Each algorithm is executed on each path 50 times.

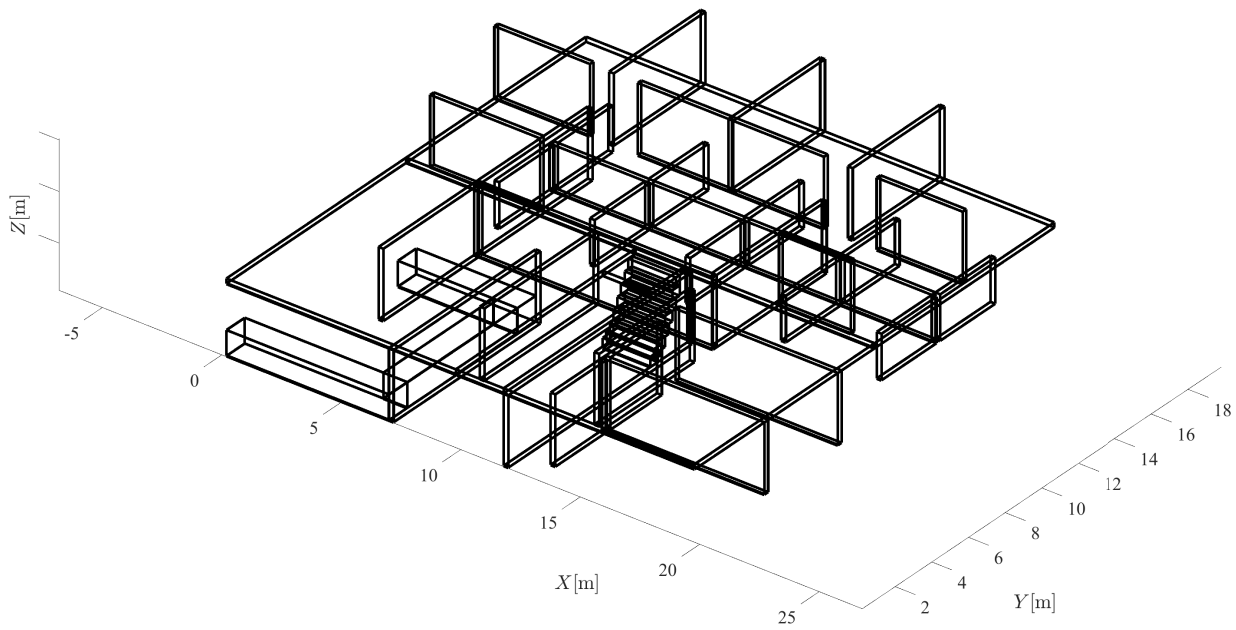


Figure 2.13: Wireframe diagram of the voxelmap used for testing.

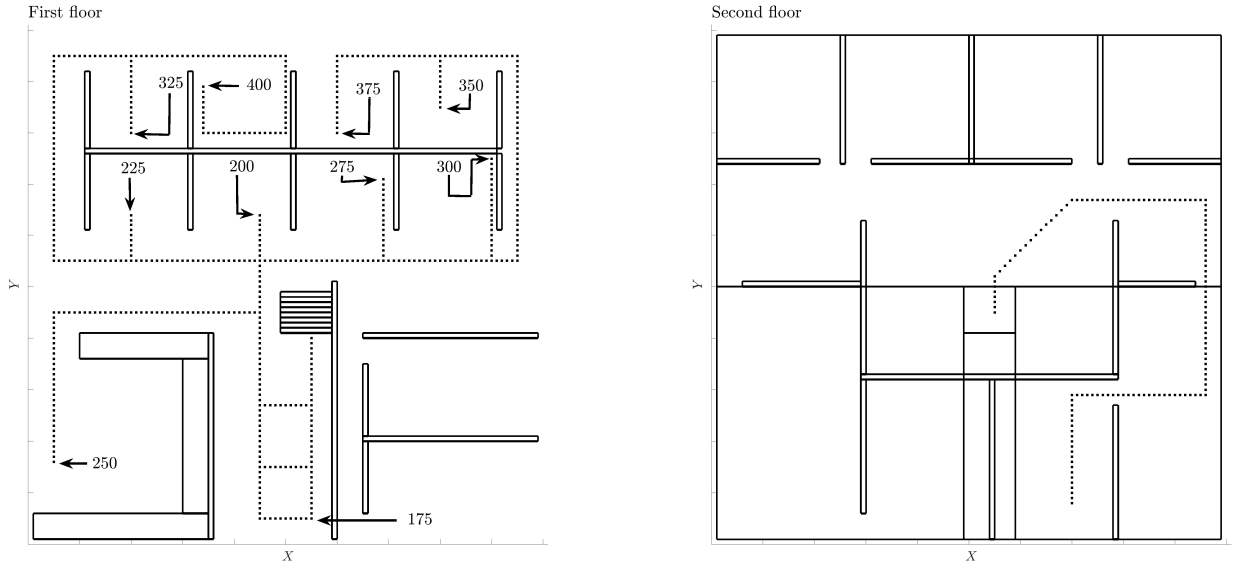


Figure 2.14: Paths used for testing. The left image shows the first floor and the right image shows the second floor. The numbers present on the plot signify the length of each path.

The values displayed in Table 2.2 reflect the coefficients of the 1^{st} order linear regression that fits the relationship between number of points in a path and the computation time of the generation of constraint planes. These lines-of-best-fit are displayed in Figure 2.15. The IRIS and SFC algorithms increase in computation time more rapidly than MDCA and Bubble Bath indicating that for paths longer than 400 points, Bubble Bath and MDCA will generate faster results.

Table 2.2: Coefficients of 1^{st} order linear regression relating path size and computation time

Algorithm	**Slope**	**Offset**
MDCA	3.374	12.920
Bubble-bath	8.669	64.354
SFC	19.684	3.693
IRIS	12.537	16.510

Figure 2.16 shows the average computational times of the algorithms considered in the

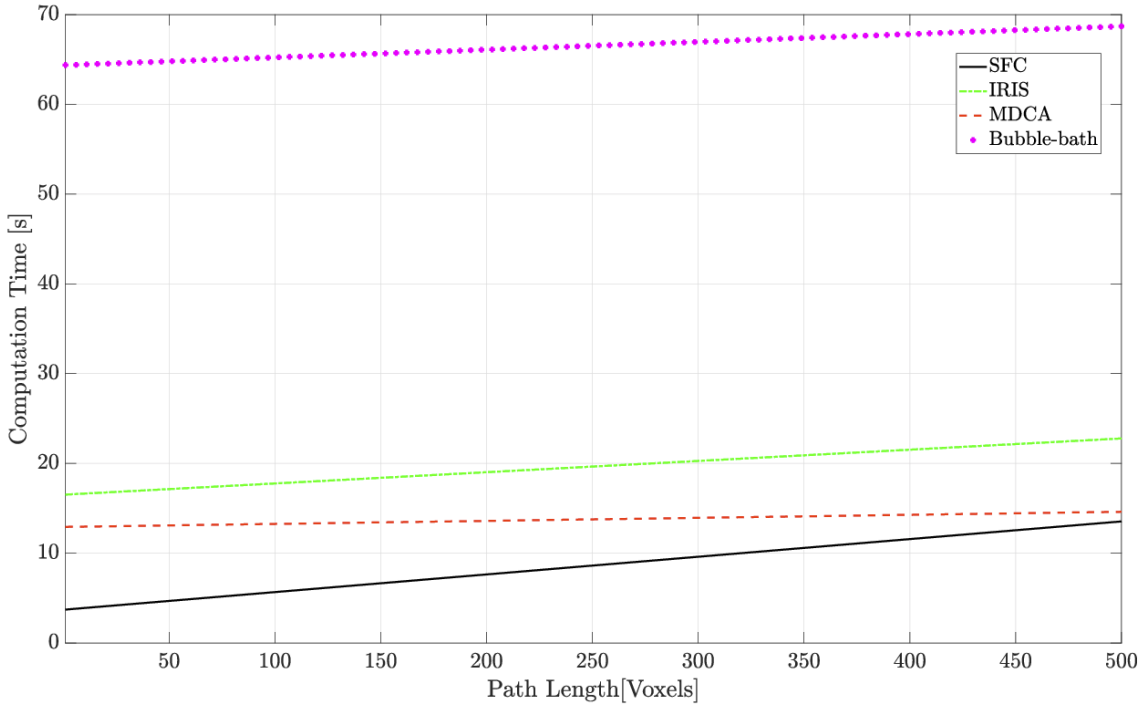


Figure 2.15: The 1^{st} order linear regression relation of path size and computation time.

study. MDCA begins and consistently remains the fastest algorithm. Bubble-bath begins approximately 40 seconds slower than the other three algorithms but completes the final path in approximately the same time as IRIS and approximately 50 seconds faster than SFC. From the size of the whisker plots for each path, Bubble-bath and and MDCA are the most consistent for a given path and therefore for a path of a fixed length, most predictable as to how long a given constraint set will take to be generated.

In terms of expected volumes, it is apparent from Figure 2.18 that the IRIS algorithm generates constraint sets with the largest average volumes. The Bubble-Bath algorithm generates larger constraint sets than the MDCA technique. The maximum and minimum values of volume of a constraint set similarly follow the trends of average constraint set volume. Notably, SFC is the only algorithm to have a deviation in the maximum volume of

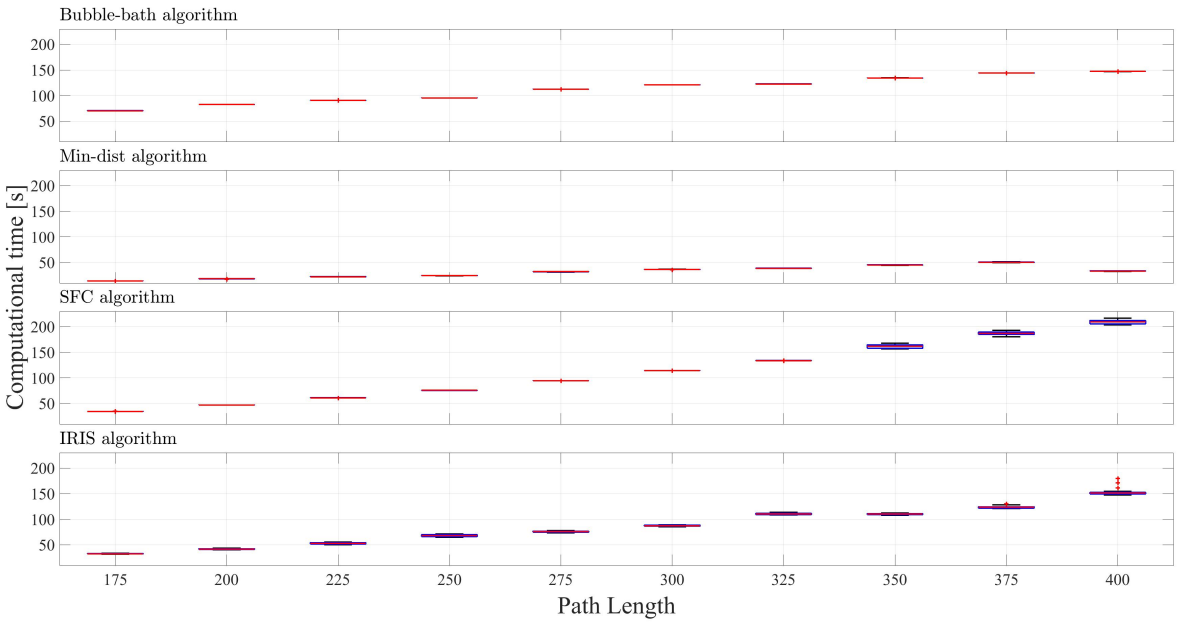


Figure 2.16: Average computation time of each path tested by the four algorithms.

a constraint set.

From Figure 2.17, we deduce that the IRIS algorithm generates constraint sets with the largest deviation in volume while MDCA is the most consistent. The SFC algorithm is the only technique to yield unbounded constraint plane sets along the test path and has the highest median number of obstacles contained within the constrained volume surrounding the path. The MDCA and Bubble-bath algorithm both have the fewest contained obstacles with near-equivalent median values of contained obstacles at 17821 and 18133 respectively.

It can be concluded from this analysis that for consistency of generated volumes, fewest contained obstacles, and speed, MDCA is the ideal algorithm among the state-of-the-art ones considered herein. In the case where volume plays a more relevant role than the computation time, IRIS should be considered. SFC does not generate larger constraint set volumes in a short enough period of time to give it any advantage over the other algorithms. Additionally, as the only algorithm to generate unbounded sets, in this evaluation, it is the most unreliable.

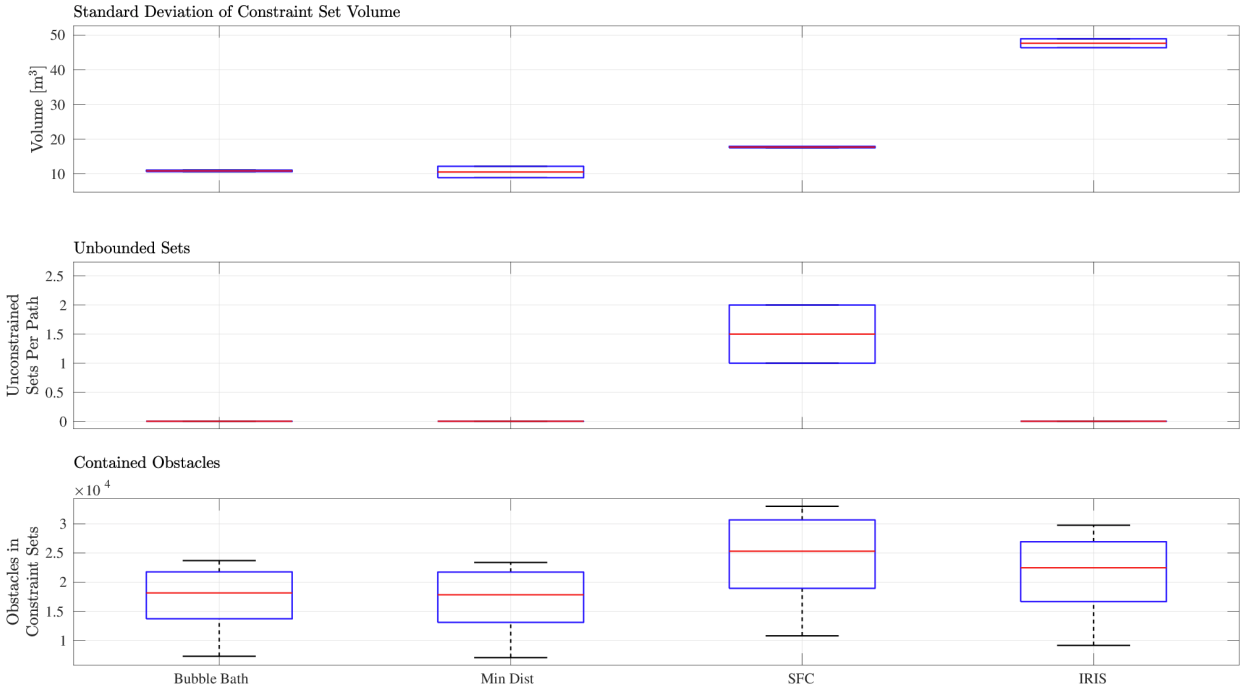


Figure 2.17: Included here is the data signifying the standard deviation of volume generated by each constraint plane set and the number of unbounded constraint sets per path.

Bubble-bath has the most consistent volume generated and computation time, making it worthy of consideration for applications where predictability is a top priority.

2.6 Numerical Simulations Results

The proposed guidance system is validated by means of two numerical simulations in which a tactical and reckless parameter set is employed. The simulation is performed on a map that mimics the layout of an office setting. The boundaries of obstacles are denoted with solid black lines, the UAV's starting position is marked with a green star, the time-history of the UAV's position is indicated with the red line, and the pink and yellow contour map is a projection of the explored set of voxels into the horizontal plane. The UAV is required in each simulation to explore at least 97% of the map.

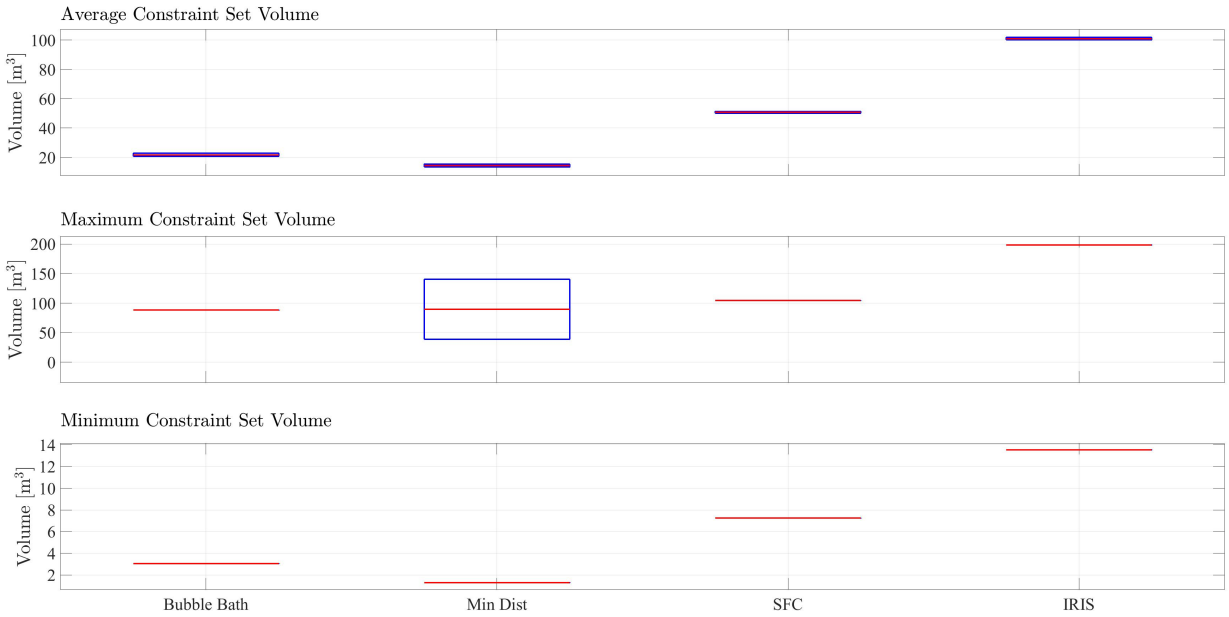


Figure 2.18: Volume information from the collected data sets. The data presented here is the average volume of constraints sets along each path for each algorithm, and the maximum and minimum volume of a constraint set. The red line signifies the median value of the data collected.

Figure 2.19 shows simulation results in a single-floor indoor environment mapped employing a reckless tactic. A heat map showing how many voxels have been explored is projection on three orthogonal planes. Figures 2.20 shows simulation results employing a tactical approach. The horizontal projection of the UAV's coverage of the environment shows that the reckless parameter set produces a less consistent mapping coverage of the environment. In both Figure 2.19 and 2.20, the UAV's trajectory intersects itself, due to Algorithm 2.2.1 not considering the coverage problem explicitly but instead to focus the UAV on nearby unexplored regions.

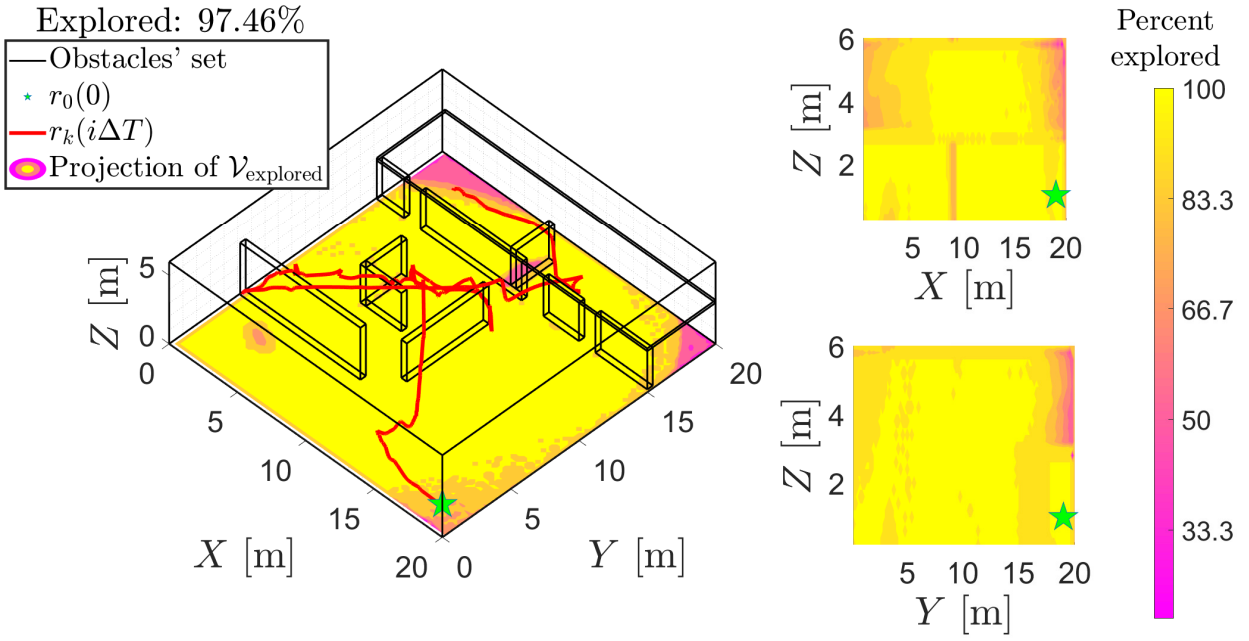


Figure 2.19: A numerical simulation of the proposed trajectory planner with user-defined parameters set to induce reckless guidance and a systematic goal selection behavior. The UAV travels at an average distance of 3.92m from the obstacle set with a standard deviation of 2.78m. The flight time is 306.52s, during which, the UAV traverses 129.57m while exploring voxels at a rate of 531.58 $\frac{voxels}{s}$.

2.7 Conclusion

The novel guidance system presented in this chapter allows for a front-facing-camera equipped UAV to autonomously map an environment while simultaneously seeking cover via nearby obstacles. User-defined parameters allow for the tuning of flight behavior to induce a more tactical or reckless behavior from the UAV. In particular, the emphasis of this chapter is placed upon the study to optimize Algorithm 2.2.1 in order to reduce computational time, the experiment performed to decide the suitability of 2.3.1 versus 2.3.2, and the experiment to compare the effectiveness of the MDCA, Bubble-Bath, SFC and IRIS algorithms in generating obstacle free environments as part of the trajectory planner. The implication of this chapter is an effective, as shown via numerical simulations, guidance system that can be employed on UAVs.

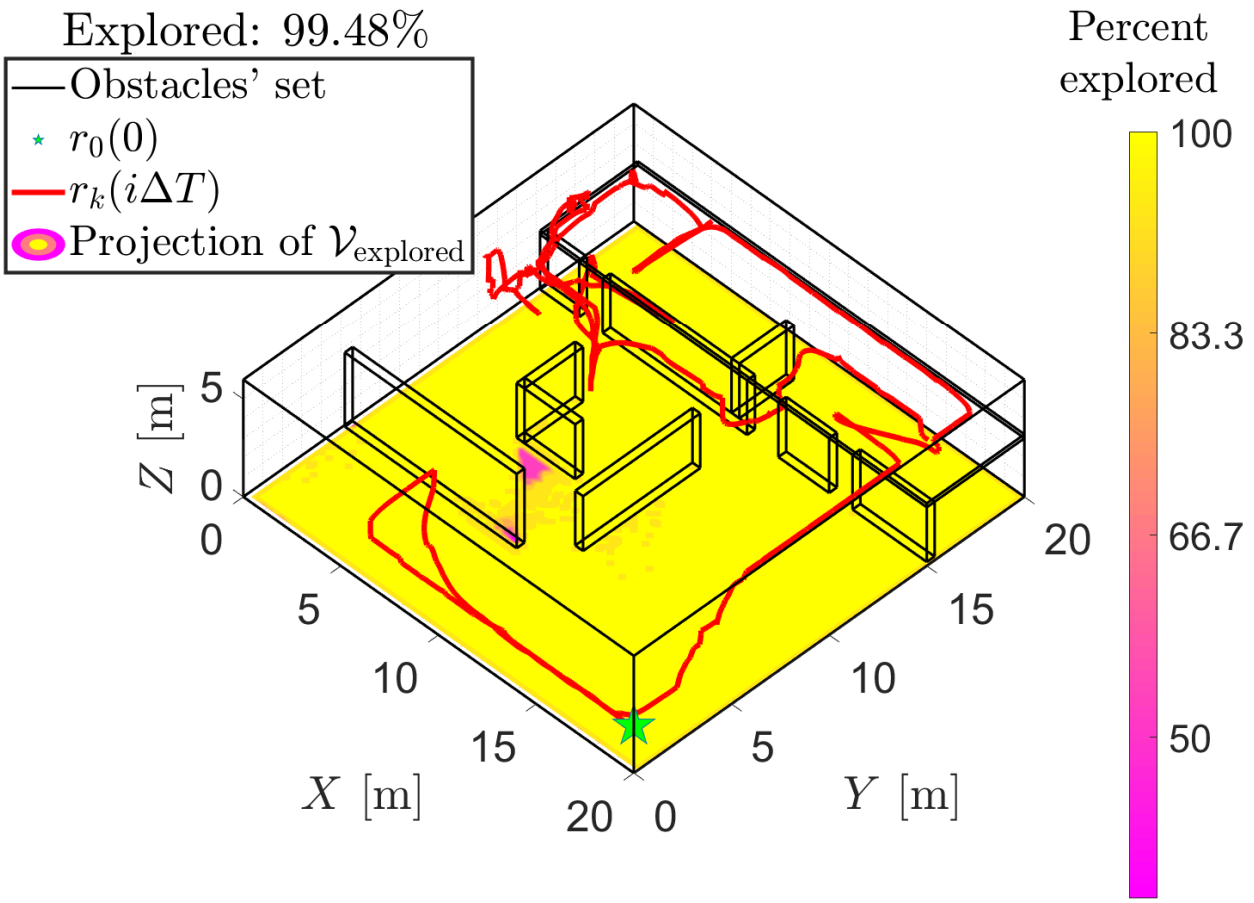


Figure 2.20: A numerical simulation of the proposed trajectory planner with user-defined parameters set to induce tactical guidance and greedy goal selection behavior. The UAV travels at an average distance of 1.02m from the obstacle set with a standard deviation of 0.94m. The flight time is 463.65s, during which, the UAV traverses 221.85m while exploring voxels at a rate of 445.87 $\frac{voxels}{s}$.

Chapter 3

An Adaptive Control System for Quad-Rotor Biplanes

3.1 Introduction

Having examined in Chapter 2 the problem of designing a vision-based guidance system for multi-rotor UAVs, in this chapter, we address the second main problem of this research, namely, the design of the control system for a vertical take-off and landing (VTOL) UAV that is poorly modeled. In particular, we focus on the Quad-Rotor Biplane (QRBP) UAV, pictured in Figure 3.1.

A key feature of this vehicle is that, being equipped with four propellers like a conventional quadcopter, it is able to take-off, land, and hover vertically. Furthermore, being equipped with wings, it is has a considerably longer range than classical quadcopters. When pitched upward, the QRBP's propellers must produce sufficient thrust to sustain the UAV's altitude and follow some user-defined trajectory. However, when pitched forward, the lift produced by the wings sustains the UAV, whereas the propellers need only to produce a sufficiently high forward velocity, and, usually, much less power is required to operate a fixed-wing UAV compared to a quadcopter of the same size and weight. The vehicle employed in this research is identical to the Army Research Lab (ARL) Common Research Configuration (CRC), and a considerable challenge for this vehicle lies in the fact that it has no control surfaces and

as such, all maneuvering is done as a result of the differential thrust as produced by its four propellers.

While the vehicle is traveling in the near vertical regime, it is considered to be in *quadcopter* mode, and while the vehicle is traveling near-horizontally, it is considered to be in *biplane* mode. In this research, the critical point whereby the QRBP switches between quadcopter and biplane mode is identified by the pitch angle such that, in purely longitudinal motion, the wings are not stalled. Ideally, such switching condition should be marked by the UAV's stall angle. However, this approach would not be practical for our vehicle. Indeed, classifying the UAV's behavior as a function of the stall angle would require both a reliable way to measure the UAV's angle of attack relative to the wind velocity and a good characterization of the UAV's aerodynamic profile. These two requirements involve the use of sufficiently reliable Pitot tubes or equivalent meters and some work in a wind tunnel. However, one of the scopes of this research is to produce control systems for poorly characterized VTOL UAVs.

Since the vehicle's dynamics is captured by two alternative models, the underlying control system reflects this separation. Furthermore, as it is common practice in the dynamics and control literature for fixed-wing UAVs, we separate both the dynamical models and the corresponding control systems into longitudinal and lateral-directional modes [43, Ch. 2]. The longitudinal control problem involves regulating the vehicle's pitch angle and its position in a vertical plane in the global reference frame. The lateral-directional control problem consists of regulating the vehicle's roll and yaw angles as well as controlling the vehicle's lateral displacements. Shown in Figure 3.2 is a diagram of the QRBP with the outputs of the longitudinal and lateral controllers noted.

To capture the longitudinal dynamics, we consider the state vector $x_{\text{long}} = [x_c, z_c, \theta, u, w, q]^{\text{T}}$, and to capture the lateral-directional dynamics, we consider the state vector $x_{\text{lat}} = [y_c, \phi, \psi, v, p, r]^{\text{T}} \in \mathbb{R}^6$, where $[x_c, y_c, z_c]^{\text{T}} : [t_0, \infty) \to \mathbb{R}^3$ captures the UAV's position in the global

Figure 3.1: The Quad-Rotor Biplane with Vicon-Motion Capture reflectors as used with indoor flight tests.

reference frame, $\phi : [t_0, \infty) \to [0, 2\pi)$ denotes the UAV's roll angle, $\theta : [t_0, \infty) \to \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ denotes the UAV's roll angle, $\psi : [t_0, \infty) \to [0, 2\pi)$ denotes the UAV's yaw angle, $[u, v, w]^{\mathrm{T}} : [t_0, \infty) \to \mathbb{R}^3$ captures the UAV's position in the global reference frame, and $[p, q, r]^{\mathrm{T}} : [t_0, \infty) \to \mathbb{R}^3$ captures the UAV's angular velocity, expressed in the UAV's frame. The longitudinal dynamics of the vehicle are controlled by an output-feedback MRAC system. The lateral dynamics of the vehicle are controlled by a novel adaptive system. The adaptive controller used for lateral movement is novel in its use of a dynamically switched model to account for the change of dynamics as the QRBP switches between *quadcopter* and *biplane* mode.

This chapter is organized as follows. Section 3.2 provides a derivation of the classical MRAC algorithm, employing the same formulation as in [44], along with the underlying arguments regarding stability. The classical MRAC architecture steers the controlled plant

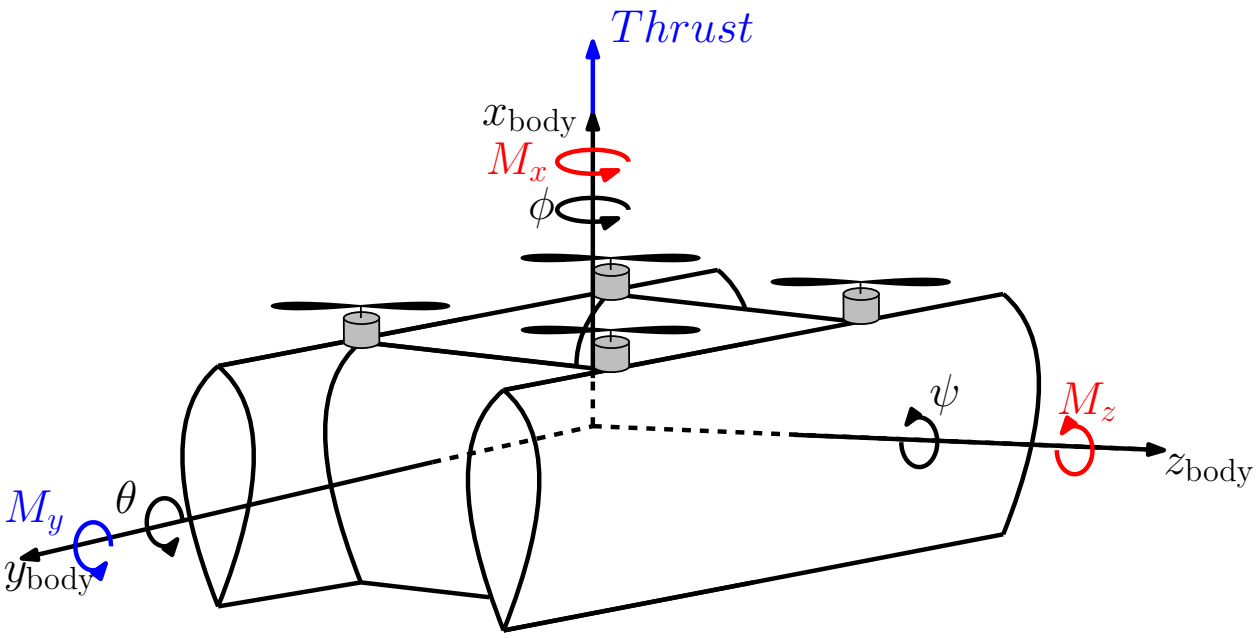


Figure 3.2: A diagram of the Quad-Rotor Biplane with body reference frame axes and Euler angles noted in black, and the outputs of the longitudinal controller and lateral controller denoted in blue and red respectively.

trajectory toward the trajectory of a reference model. Thus, the user unable to provide an arbitrary reference signal to follow. In Section 3.3, we present an adaptive control architecture that modifies the classical MRAC law by means of an integral error term, which steers the plant trajectory toward an arbitrary reference trajectory. Section 3.4 provides the theory of a switched MRAC system which underlies the lateral controller of the UAV. Section 3.5 details the specific implementation of the previously mentioned theories into a control system for the QRBP. Finally, Section 3.6 draws conclusions about this work.

3.2 Fundamentals of Model Reference Adaptive Control

In this section, we recall key elements of the classical MRAC architecture [44]. According to this architecture, we consider the problem of controlling a plant affected by matched and parametric uncertainties. In particular, we consider the problem of steering the plant trajectory toward the trajectory of a user-defined reference model. Later in this chapter, we will discuss how this approach can be extended to address the problem of following a reference signal that is not the generated as the state of a dynamical model.

To present the direct MRAC framework, consider the plant model

$$\dot{x}(t) = Ax(t) + B\Lambda(u(t) + \Theta^{\mathrm{T}}\Phi(t, x(t))), \qquad x(t_0) = x_0, \qquad t \geq t_0. \tag{3.1}$$

The system state is denoted by $x : [t_0, \infty) \to \mathbb{R}^n$. The control input generated by the presented controller is denoted $u : [t_0, \infty) \to \mathbb{R}^m$. The matrix representing the system's dynamic is unknown, assumed constant and denoted by $A \in \mathbb{R}^{n\times n}$. It is worth noting that, in practice, the structure of A is known. The control matrix, known to the user, is denoted by $B \in \mathbb{R}^{n\times m}$. Control failures and modeling errors are modeled by $\Lambda \in \mathbb{R}^{m\times m}$ where Λ is unknown and constant, diagonal, with elements, λ_i, that are positive. The known basis functions $\Phi : [t_0, \infty) \to \mathbb{R}^N$ are assumed to be Lipschitz-continuous, is known as the regressor vector, is denoted by $\Phi(x, t) \in \mathbb{R}^N$.

Part of the controller design is the selection of a reference model that is stable and reflects the user-desired response of the system to bounded command trajectories. The general form

of the reference model is

$$\dot{x}_{\text{ref}}(t) = A_{\text{ref}}x_{\text{ref}}(t) + B_{\text{ref}}r(t), \qquad x_{\text{ref}}(t_0) = x_{\text{ref},0}, \qquad t \geq t_0. \tag{3.2}$$

The reference model state is denoted by $x_{\text{ref}} : [t_0, \infty) \to \mathbb{R}^n$. Additionally, $r : [t_0, \infty) \to \mathbb{R}^m$ is the user defined, externally bounded command trajectory of the reference model. The matrix representing the reference model's dynamics, denoted by $A_{\text{ref}} \in \mathbb{R}^{n\times n}$, is user defined and Hurwitz. The reference model control matrix, is denoted by $B_{\text{ref}} \in \mathbb{R}^{n\times m}$.

To track the error between the system and its reference model, the trajectory tracking error is defined as $e(t) = x(t) - x_{\text{ref}}(t)$. Thus, a control law for the control input $u(t)$ is defined such that $e(t)$ tends globally uniformly asymptotically to 0 as $t \to \infty$. To generate an ideal solution for a completely known system in which there are no parametric uncertainties, the control law is defined as

$$u_{\text{ideal}}(t) = K_x x(t) + K_r r(t) - \Theta^T \Phi(x(t)), \tag{3.3}$$

where $K_x \in \mathbb{R}^{n\times m}$ and $K_r \in \mathbb{R}^{m\times m}$ verify the matching conditions

$$A + B\Lambda K_x^T = A_{\text{ref}}, \tag{3.4}$$

$$B\Lambda K_r^T = B_{\text{ref}}. \tag{3.5}$$

In practice, the matching conditions can not be verified because A and Λ are unknown. Thus, inspired by (3.2), the MRAC control law is given by

$$u(t) = \hat{K}_x^T x(t) + \hat{K}_r^T r(t) - \hat{\Theta}^T \Phi(x(t)), \tag{3.6}$$

where $\hat{K}_x : [t_0, \infty) \to \mathbb{R}^{n\times m}, \hat{K}_r : [t_0, \infty) \to \mathbb{R}^{m\times m}$, and $\hat{\Theta} : [t_0, \infty) \to \mathbb{R}^{N\times m}$ are the

adaptive gains replacing the unknown matrices used in the ideal control law.

Applying the control input given by (3.2), the closed-loop plant trajectory is given by

$$\dot{x}(t) = \left(A + B\Lambda\hat{K}_x^{\mathrm{T}}\right)x(t) + B\Lambda\left(\hat{K}_r^{\mathrm{T}}r(t) - (\hat{\Theta}(t) - \Theta)^T\Phi(x(t))\right), \qquad x(t_0) = x_0, \qquad t \geq t_0, \quad (3.7)$$

and the trajectory tracking error dynamics is given by

$$\begin{aligned}\dot{e}(t) &= \dot{x}(t) - \dot{x}_{\mathrm{ref}}(t) \\ &= A_{\mathrm{ref}}e(t) + B\Lambda\left(\Delta K_x^T x + \Delta K_r^T r - \Delta\Theta^T\Phi(x)\right), \qquad e(t_0) = x_0 - x_{\mathrm{ref}.0}, \qquad t \geq t_0,\end{aligned} \quad (3.8)$$

where $\Delta K_x(t) = \hat{K}_x(t) - K_x$, $\Delta K_r(t) = \hat{K}_r(t) - K_r$, and $\Delta\Theta(t) = \hat{\Theta}(t) - \Theta$.

Adaptive gains are chosen to enforce Lyapunov stability of the origin of (3.2) and uniform asymptotic convergence of the trajectory tracking error to zero. These adaptive laws are given by

$$\dot{\hat{K}}_x = -\Gamma_x x(t)e(t)^{\mathrm{T}}PB, \qquad \hat{K}_x(t_0) = \hat{K}_{x0}, \qquad t \geq t_0, \quad (3.9)$$

$$\dot{\hat{K}}_r = -\Gamma_r r(t)e(t)^{\mathrm{T}}PB, \qquad \hat{K}_r(t_0) = \hat{K}_{r0}, \qquad t \geq t_0, \quad (3.10)$$

$$\dot{\hat{\Theta}} = \Gamma_\theta\Phi(x(t))e(t)^{\mathrm{T}}PB, \qquad \hat{\Theta}(t_0) = \hat{\Theta}_0, \qquad t \geq t_0, \quad (3.11)$$

where the adaptive rate matrices $\Gamma_x \in \mathbb{R}^{n\times n}$, $\Gamma_r \in \mathbb{R}^{m\times m}$, and $\Gamma_\theta \in \mathbb{R}^{N\times N}$ are symmetric, positive-definite, and user-defined and $P \in \mathbb{R}^{n\times n}$ denotes the symmetric, positive-definite

solution of

$$PA_{\text{ref}} + A_{\text{ref}}^{\text{T}} P = -Q \tag{3.12}$$

with $Q \in \mathbb{R}^{n \times n}$ symmetric, positive-definite, and user-defined.

3.3 Model Reference Adaptive Control with Output Signal Tracking

The theory laid out in Section 3.2 is augmented in this section for bounded signal tracking. A challenge in the use of classical model reference adaptive control lies in the fact that the user is unable to design the reference trajectory directly, but through the design of A_{ref}, B_{ref}, and $r(\cdot)$. The approach proposed in this section, instead, allows the user to design the reference trajectory explicitly.

Consider the plant model

$$\dot{x}_p(t) = A_p x_p(t) + B_p \Lambda \left(u(t) + \Theta^T \Phi(x_p(t)) \right), \qquad x(t_0) = x_0, \qquad t \geq t_0. \tag{3.13}$$

The state vector of the system's plant is denoted by $x_p : t[0, \infty) \to \mathbb{R}^{n_p}$. Matched nonlinear uncertainties are captured by the unknown matrix $\Theta \in \mathbb{R}^{N \times m}$ and the user-defined regressor vector $\Phi : [t_0, \infty) \times \mathbb{R}^{n_p} \to \mathbb{R}^N$. The matrix $A_p \in \mathbb{R}^{n \times n}$ is constant and unknown while the plant dynamics matrix $B_p \in \mathbb{R}^{n \times m}$ is constant and known. Control actuation errors are captured by $\Lambda \in \mathbb{R}^{m \times m}$ which is constant, diagonal, unknown, and is comprised of positive elements.

The pair $(A_p, B_p \Lambda)$ is assumed to be controllable. In order to accomplish the goal of

bounded command tracking, $u(\cdot)$ is designed so that the system's output can track any bounded, time-varying command. As such, the system's output is given by

$$y(t) = C_p x_p(t) + D_p[u(t) + \Theta^{\mathrm{T}}\Phi(t, x_p(t))], \tag{3.14}$$

where $C_p \in \mathbb{R}^{m\times n}$ and $D_p \in \mathbb{R}^{m\times m}\backslash\{0\}$. Remarkably, the fact that D_p can not be set equal to zero is not a limitation for the purposes of this control scheme. Now, let $y_{\mathrm{cmd}} : [t_0, \infty) \to \mathbb{R}^m$ denote the user-defined command signal. Thus, the system's output tracking error is given by $e_y(t) = y(t) - y_{\mathrm{cmd}}(t)$, and the integrated output tracking error is defined $e_{yI}(t) = \int_0^t e_y(\tau)\, d\tau$.

The augmented system state-vector is written as $x(t) = \left[e_{yI}^{\mathrm{T}}(t), x_p^{\mathrm{T}}(t)\right]^{\mathrm{T}} \in \mathbb{R}^n$, where $n = n_p + m$. Therefore, the augmented open-loop dynamics can be written as

$$\dot{x}(t) = Ax(t) + B\Lambda\left(u(t) + \Theta^{\mathrm{T}}\Phi(t, x(t))\right) + B_{\mathrm{ref}} y_{\mathrm{cmd}}(t), \tag{3.15}$$

$$y(t) = Cx(t) + Du(t), \tag{3.16}$$

where

$$A = \begin{bmatrix} 0_{m\times m} & C_p \\ 0_{n_p\times m} & A_p \end{bmatrix}, \quad B = \begin{bmatrix} 0_{m\times m} \\ B_p \end{bmatrix}, \quad B_{\mathrm{ref}} = \begin{bmatrix} -I_{m\times m} \\ 0_{n_p\times m} \end{bmatrix}, \quad \text{and} \quad C = \begin{bmatrix} 0_{m\times m} & C_p \end{bmatrix}.$$

Assuming that the matching condition in Equation (3.4) and Equation (3.5) are verified, the reference model of the system is given by

$$\dot{x}_{\mathrm{ref}}(t) = A_{\mathrm{ref}} x_{\mathrm{ref}}(t) + B_{\mathrm{ref}} y_{\mathrm{cmd}}(t), \qquad x_{\mathrm{ref}}(t_0) = x_{\mathrm{ref},0}, \qquad t \geq t_0, \tag{3.17}$$

$$y_{\mathrm{ref}}(t) = Cx_{\mathrm{ref}}(t). \tag{3.18}$$

for this regulation problem, the MRAC control input is given by

$$u(t) = \hat{K}_x^T x(t) \tag{3.19}$$

where $\hat{K}_x(\cdot)$ verifies Equation (3.9).

3.4 Switched MRAC

This section presents an MRAC architecture for nonlinear, time-varying switched dynamical systems, that is, for systems whose dynamics vary instantaneously at unknown time instants [45]. This architecture is employed to derive a control law capable of handling the QRBP operating in the two distinct modes of quadcopter and biplane mode.

As discussed in [46], there exist multiple notions of solution of switched differential equation. In this thesis, we focus on the Carathéodory solutions. Given the switched dynamical model

$$\dot{x}(t) = f_{\sigma(t)}(t, x(t)), \quad x(t_0) = x_0, \quad t \geq t_0, \tag{3.20}$$

Carathéodory solutions are absolutely continuous functions [47, Def. 6.1.1] so that

$$x(t) = x_0 + \int_{t_0}^{t} f(\tau, x(\tau))\mathrm{d}\tau, \tag{3.21}$$

where $\sigma : [t_0, \infty) \to \Sigma \subseteq \mathbb{N}$ and the integral is defined in the sense of Lebesgue. As shown in [45], thus far, there exist MRAC systems for switched plants only in the Carathéodory and the Filippov framework. In this thesis, we consider the former only, and omit the latter for brevity.

Consider the nonlinear system

$$\dot{x}(t) = A_{\sigma(t)}x(t) + B_{\sigma(t)}[u(t) + \Theta^{\mathrm{T}}\Phi_{\sigma(t)}(t, x(t))], \qquad x(t_0) = x_0, \qquad t \geq t_0, \tag{3.22}$$

where the set $\mathcal{D} \subseteq \mathbb{R}^n$ is open, connected and such that $0 \in \mathcal{D}$. The plant's trajectory is denoted by $x : [t_0, \infty) \to \mathcal{D}$, in the sense of Carathéodory, and the control input is given by $u : [t_0, \infty) \to \mathbb{R}^m$. The user-defined switching signal is denoted $\sigma : [t_0, \infty) \to \Sigma$, and $\Sigma \subset \mathbb{N}$ is bounded and, hence, we assume that the first $\overline{\sigma}$ positive integers comprise Σ, where $\overline{\sigma}$ denotes the cardinality of Σ. The parametric uncertainties of the plant are captured by the unknown matrix $A_s \in \mathbb{R}^{n\times n}, s \in \Sigma$, with the mapping $s \mapsto A_s$ also considered to be unknown. While the elements of A_s are unknown, in practical applications, the structure of A_s is known. The control matrix, denoted by $B_s \in \mathbb{R}^{n\times m}$ is known. We assume that the corresponding pairs (A_s, B_s) are controllable for all $s \in \Sigma$. The term $\Theta^{\mathrm{T}}\Phi_s(t, x), (s, t, x) \in \Sigma \times [t_0, \infty) \times \mathbb{R}$, captures matched uncertainties where $\Theta \in \mathbb{R}^{N\times m}$, is unknown and the regressor vector, denoted by $\Phi_s : [t_0, \infty) \times \mathbb{R}^n \to \mathbb{R}^N$ is known, Lebesgue integrable, and jointly continuous in its arguments. The regressor vector, $\Phi_s(t, \cdot)$ is Lipschitz continuous in x uniformly in t on compact subsets of $[t_0, \infty)$.

The plant's dynamical reference model, with time dependent switching, is given by

$$\dot{x}_{\mathrm{ref}}(t) = A_{\mathrm{ref},\sigma(\mathrm{t})}x_{\mathrm{ref}}(t) + B_{\mathrm{ref},\sigma(\mathrm{t})}r(t), \qquad x_{\mathrm{ref}}(t_0) = x_{\mathrm{ref},0}, \qquad t \geq t_0. \tag{3.23}$$

where the reference model trajectory is denoted by $x_{\mathrm{ref}} : [t_0, \infty) \to \mathbb{R}^n$. Additionally, $r : [t_0, \infty) \to \mathbb{R}^m$ is the piecewise continuous and bounded command input of the reference model. The matrix representing the reference model's dynamics, denoted by $A_{\mathrm{ref,s}} \in \mathbb{R}^{n\times n}$, is user defined and Hurwitz. The reference model control matrix, is denoted by $B_{\mathrm{ref,s}} \in \mathbb{R}^{n\times m}$.

For all $s \in \Sigma$, we assume that pairs $(K_{x,s}, K_{r,s}) \in \mathbb{R}^{n\times m} \times \mathbb{R}^{m\times m}$ exist such that matching conditions

$$A_{\text{ref,s}} = A_s + B_s K_{x,s}^{\mathrm{T}}, \qquad s \in \Sigma \tag{3.24}$$

$$B_{\text{ref,s}} = B_s K_{r,s}^{\mathrm{T}} \tag{3.25}$$

are verified. Each linear, time-invariant, dynamical system comprising the reference model (3.23) is input-to-state stable, therefore we assume that there is a dwell-time such that the reference dynamical model (3.23) is stable. The dwell-time is the parameter such that, if the switching time between two consecutive modes is larger than the dwell time, then no instability is induced by the switching.

Let $P \in \mathbb{R}^{n\times n}$ denote the symmetric, positive-definite solution of Lyapunov matrix inequalities

$$A_{\text{ref,s}}^{\mathrm{T}} P + P A_{\text{ref,s}} < -Q, \qquad s \in \Sigma, \tag{3.26}$$

where $Q \in \mathbb{R}^{n\times n}$ is symmetric, positive-definite, and user-defined. If $u(t) = \phi(\hat{\Theta}(t), \tilde{\Phi}_{\sigma(t)}(t, x(t)))$, $t \geq t_0$, the control law, denoted by

$$\phi(\hat{\Theta}, \tilde{\Phi}_s) = \hat{\Theta}^{\mathrm{T}} \tilde{\Phi}_s, \qquad (s, \hat{\Theta}, \tilde{\Phi}_s) \in \Sigma \times \mathbb{R}^{(\overline{\sigma}(n+m)+N)\times m} \times \mathbb{R}^{\overline{\sigma}(n+m)+N}, \tag{3.27}$$

$\Delta\Theta(t) \triangleq \hat{\Theta}(t) - \tilde{\Theta}, \hat{\Theta} : [t_0, \infty) \to \mathbb{R}^{(\bar{\sigma}(n+m)+N)\times m}$

$$\dot{\hat{\Theta}}(t) = -\Gamma\tilde{\Phi}_{\sigma(t)}(t, x(t))e^T(t)PB_{\sigma(t)}, \qquad \hat{\Theta}(t_0) = \hat{\Theta}_0, \tag{3.28}$$

$$\tilde{\Phi}_s(t, x) \triangleq \begin{bmatrix} \mathcal{I}(s) \otimes x(t) \\ \mathcal{I}(s) \otimes r(t) \\ -\Phi_s(t, x) \end{bmatrix}, \qquad (s, t, x) \in \Sigma \times [t_0, \infty) \times \mathbb{R}^n, \tag{3.29}$$

$$\mathcal{I}(s) \triangleq \left[\mathbf{1}_{\{s\in\Sigma:s-1=0\}}(s), \ldots, \mathbf{1}_{\{s\in\Sigma:s-\bar{\sigma}=0\}}(s)\right]^{\mathrm{T}}, \tag{3.30}$$

$$\tilde{\Theta} \triangleq \left[K_{x,1}^{\mathrm{T}}, \ldots, K_{x,\bar{\sigma}^{\mathrm{T}}}, K_{r,1}^{\mathrm{T}}, \ldots, K_{r,\bar{\sigma}}^{\mathrm{T}}, \Theta^{\mathrm{T}}\right]^{\mathrm{T}}, \tag{3.31}$$

$\Gamma \in \mathbb{R}^{(\bar{\sigma}(n+m)+N)\times(\bar{\sigma}(n+m)+N)}$ denotes the adaptive rate matrix, $\sigma : [t_0, \infty) \to \Sigma$ denotes the user-defined switching signal, $\Sigma \subset \mathbb{N}$ is bounded, and Σ comprises the first $\bar{\sigma}$ positive integers, where $\bar{\sigma}$ denotes the cardinality of Σ. The trajectory tracking error dynamics are given by

$$\dot{e}(t) = A_{\mathrm{ref},\sigma(t)}e(t) + B_{\sigma(t)}\Delta\Theta^T(t)\tilde{\Phi}_{\sigma(t)}(t, x(t)), \qquad e(t_0) = x_0 - x_{\mathrm{ref},0}, \qquad t \geq t_0. \tag{3.32}$$

Theorem 3.1. *Consider the closed-loop trajectory tracking error dynamics* (3.32) *and the adaptive law* (3.28)*. Assume that the matching conditions* (3.24) *and* (3.25) *are verified, and there exists positive-definite matrices* $P, Q \in \mathbb{R}^{n\times n}$ *so that both the trajectory tracking error* $e(\cdot)$ *and the adaptive gain matrix* $\hat{\Theta}(\cdot)$ *are bounded uniformly in both* $t_0 \in [0, \infty)$ *and* $\sigma(\cdot)$*, and* $e(t) \to 0$ *as* $t \to \infty$ *uniformly in both* t_0 *and* $\sigma(\cdot)$*.*

3.5 Implementation

This section details how the theory of Model Reference Adaptive Control, discussed in Section 3.2, and its derivations are utilized in the control of a QRBP.

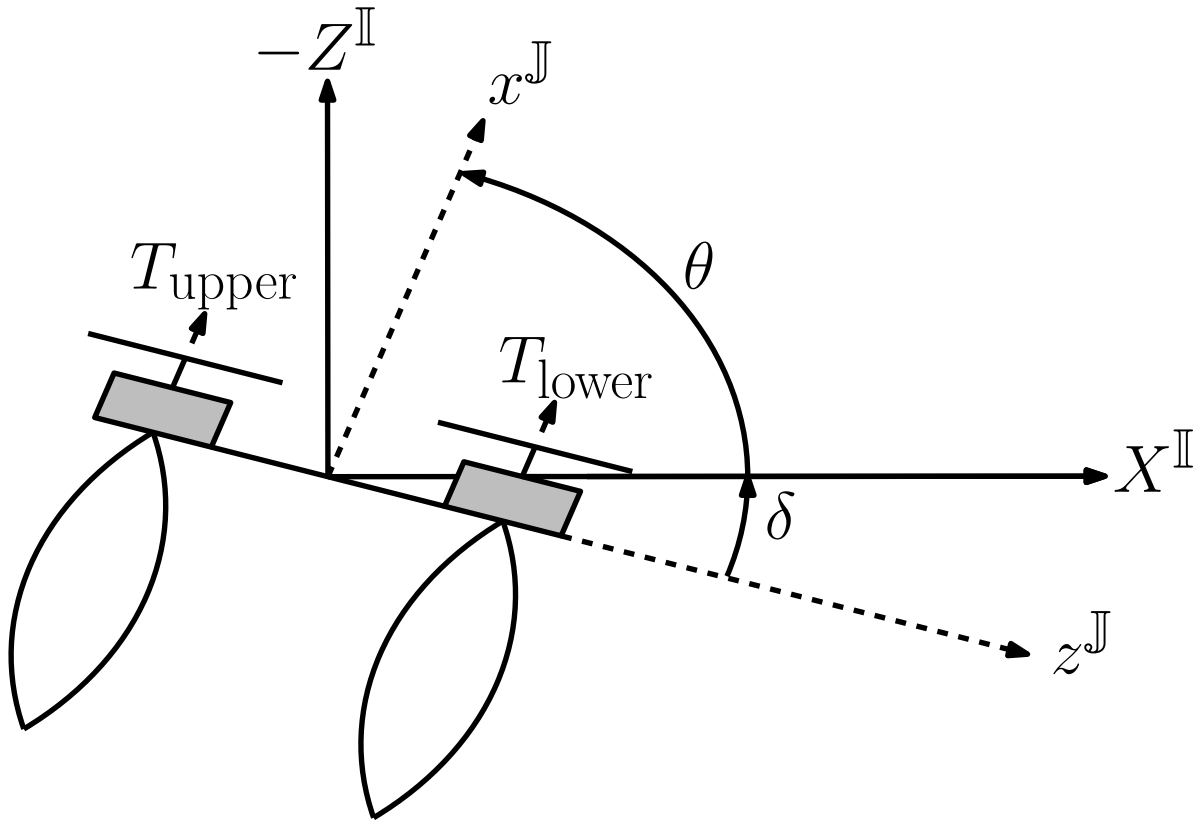


Figure 3.3: Schematic representation of the QRBP, inertial, longitudinal axes, and pitch angle. The UAV's roll and yaw axes are denoted by x and z, respectively. The thrust generated by the propellers along the $-z^{\mathbb{J}}$ axis is denoted by T_{upper}. The thrust generated by the propellers along the $z^{\mathbb{J}}$ axis is denoted by T_{lower}.

By proceeding as in [20], longitudinal control of the QRBP is performed by applying MRAC with output tracking discussed in Section 3.3. The lateral control is handled by a switched MRAC system discussed in Section 3.4.

3.5.1 Longitudinal Control

The longitudinal controller features an inner and outer loop controller to control attitude and position, respectively. A diagram of the QRBP in purely longitudinal motion is shown in Figure 3.3.

Remarkably, in light of the proposed approach, the pitch angle is not constrained in the interval $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$. Initially, we assume the UAV is symmetric and moving only in the plane created by the x and z axes of the inertial frame, denoted by $\mathbb{I}$. The translational dynamic equations of the QRBP are given by

$$\ddot{x}_A(t) = m^{-1}\left[F_X(t) - L(t, v(t))\sin\gamma(t) - D(t, v(t))\cos\gamma(t)\right], \tag{3.33}$$
$$x_A(t_0) = x_{A,0}, \quad \dot{x}_A(t_0) = x_{A,0,d},$$
$$\ddot{z}_A(t) = m^{-1}\left[-F_Z(t) + mg - L(t, v(t))\cos\gamma(t) + D(t, v(t))\sin\gamma(t)\right], \tag{3.34}$$
$$z_A(t_0) = z_{A,0}, \quad \dot{z}_A(t_0) = z_{A,0,d},$$

where the components of thrust force are defined as

$$F_x^{\mathbb{I}}(t) = (T_{\text{upper}}(t) + T_{\text{lower}}(t))\cos\theta(t), \tag{3.35}$$
$$F_z^{\mathbb{I}}(t) = (T_{\text{upper}}(t) + T_{\text{lower}}(t))\sin\theta(t), \tag{3.36}$$

the lift and drag of the vehicle, respectively, are given by

$$L(t, v) = \frac{1}{2}\rho||v||^2 S C_l(\alpha(t)), \tag{3.37}$$
$$D(t, v) = \frac{1}{2}\rho||v||^2 S C_d(\alpha(t)), \tag{3.38}$$

air density is denoted by $\rho > 0$, the platform area of the wing is denoted $S > 0$, the aerodynamic lift coefficient is denoted by $C_l : [0, 2\pi) \to [0, \infty)$, the aerodynamic drag coefficient is denoted by $C_d : [0, 2\pi) \to (0, \infty)$, the angle of attack is denoted by

$$\alpha(t) = \text{atan2}(\dot{z}_A(t), \dot{x}_A(t)), \tag{3.39}$$

with the function

$$\operatorname{atan2}(y,x) \triangleq \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{if } x > 0, \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{if } x < 0 \text{ and } y \geq 0, \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{if } x < 0 \text{ and } y < 0, \\ \frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0, \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0, \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0, \end{cases},$$

and $\gamma(t) \triangleq \theta(t) - \alpha(t)$ denotes the UAV's flight path angle.

The rotational dynamic equation of the QRBP is given by

$$\ddot{\theta}(t) = I_y^{-1}\left[M_y(t) - M_a(t, v(t))\right], \quad \theta(t_0) = \theta_0, \quad \dot{\theta}(t_0) = \theta_{0,d}, \quad t \geq t_0, \tag{3.40}$$

where $I_y > 0$ denotes the UAV's principal moment of inertia about the $y^{\mathbb{J}}$ axis, $M_a(t, v) \triangleq \frac{1}{2}\rho||v||^2 SlC_m(\alpha(t))$ denotes the aerodynamic moment, and $C_m : \mathbb{R} \to \mathbb{R}$ denotes the aerodynamic moment coefficient. The moment of the thrust is denoted by

$$M_y(t) = l\left[T_{\text{lower}}(t) - T_{\text{upper}}(t)\right], \quad t \geq t_0, \tag{3.41}$$

where $l > 0$ denotes the distance from the center of mass to the rotors.

The outer loop dynamics are captured by

$$\dot{x}_{\text{outer}}(t) = A_{\text{outer}}\, x_{\text{outer}}\,(t) + B_{\text{outer}} \begin{bmatrix} F_X^{\mathbb{I}}(t) \\ -F_Z^{\mathbb{I}}(t) \end{bmatrix} + B_{\text{outer}} \begin{bmatrix} -\frac{1}{2}\rho\|v(t)\|^2 S\left[(C_{l,0} + C_{l,\alpha}\alpha(t))\sin\gamma(t) + (C_{d,0} + C_{d,\alpha}\alpha(t))\cos\gamma(t)\right] \\ mg + \frac{1}{2}\rho\|v(t)\|^2 S\left[(C_{d,0} + C_{d,\alpha}\alpha(t))\sin\gamma(t) - (C_{l,0} + C_{l,\alpha}\alpha(t))\cos\gamma(t)\right] \end{bmatrix},$$

$$x_{\text{outer}}\,(t_0) = [x_{A,0}, z_{A,0}, x_{A,0,\ \mathrm{d}}, z_{0,\ \mathrm{d}}]^{\mathrm{T}}, \quad t \geq t_0, \quad (3.42)$$

where $A_{\text{outer}} = \begin{bmatrix} 0_{2\times 2} & I_2 \\ 0_{2\times 2} & 0_{2\times 2} \end{bmatrix}$, $B_{\text{outer}} = m^{-1}\begin{bmatrix} 0_{2\times 2} \\ I_2 \end{bmatrix}$, and $x_{\text{outer}} = \begin{bmatrix} x_A(t) & z_A(t) & \dot{x}_A & \dot{z}_A \end{bmatrix}^T$,

$$F_X^{\mathbb{I}}(t) = u_{\text{outer},1}(t) + \frac{1}{2}\hat{\rho}\|v(t)\|^2\hat{S}[\left(\hat{C}_{l,0} + \hat{C}_{l,\alpha}\alpha(t)\right)\sin\gamma(t) + \left(\hat{C}_{d,0} + \hat{C}_{d,\alpha}\alpha(t)\right)\cos\gamma(t)], \quad t \geq t_0 \quad (3.43)$$

and

$$-F_Z^{\mathbb{I}}(t) = u_{\text{outer},2}(t) - \frac{1}{2}\hat{\rho}\|v(t)\|^2\hat{S}\left(\hat{C}_{d,0} + \hat{C}_{d,\alpha}\alpha(t)\right)\sin\gamma(t), t \geq t_0. \quad (3.44)$$

Therefore, (3.42) reduces to

$$\dot{x}_{\text{outer}}\,(t) = A_{\text{outer}}\, x_{\text{outer}}\,(t) + B_{\text{outer}}\left[u_{\text{outer}}\,(t) + \Theta_{\text{outer}}^{\mathrm{T}}\,\Phi_{\text{outer}}\,(t)\right]$$

$$x_{\text{outer}}\,(t_0) = [x_{A,0}, z_{A,0}, x_{A,0,\ \mathrm{d}}, z_{A,0,\ \mathrm{d}}]^{\mathrm{T}}, \quad t \geq t_0, \quad (3.45)$$

where the virtual control input is denoted $u_{\text{outer}} \triangleq [u_{\text{outer},1}(t), u_{\text{outer},2}(t)]^{\mathrm{T}} \in \mathbb{R}^2$ and the expressions comprising $\Theta_{\text{outer}}(t) \in \mathbb{R}^{5\times 2}$ and $\Phi_{\text{outer}}(t) \in \mathbb{R}^5$ are given in the Appendix B.

For inner loop control, the pitch command is given by

$$\theta_{\text{cmd}}(t) \triangleq \begin{cases} \text{atan2}\left(F_Z^{\mathbb{I}}(t), F_X^{\mathbb{I}}(t)\right), & \text{if } \left|F_X^{\mathbb{I}}(t)\right| > \varepsilon_X \quad \text{and } -F_Z^{\mathbb{I}}(t) + mg > \varepsilon_Z, \\ \text{atan2}\left(\left|F_Z^{\mathbb{I}}(t)\right|, F_X^{\mathbb{I}}(t)\right), & \text{if } \left|F_X^{\mathbb{I}}(t)\right| > \varepsilon_X \quad \text{and } -F_Z^{\mathbb{I}}(t) + mg \leq -\varepsilon_Z, \\ \text{atan2}\left(\left|F_Z^{\mathbb{I}}(t)\right|, F_X^{\mathbb{I}}(t)\right), & \text{if } \left|F_X^{\mathbb{I}}(t)\right| > \varepsilon_X \quad \text{and } \left|mg - F_Z^{\mathbb{I}}(t)\right| < \varepsilon_Z, \\ \text{atan2}\left(\left|F_Z^{\mathbb{I}}(t)\right|, F_X^{\mathbb{I}}(t)\right), & \text{if } \left|F_X^{\mathbb{I}}(t)\right| \leq \varepsilon_X \quad \text{and } -F_Z^{\mathbb{I}}(t) + mg \geq \varepsilon_Z, \\ \frac{\pi}{2}, & \text{if } \left|F_X^{\mathbb{I}}(t)\right| \leq \varepsilon_X \quad \text{and } -mg + F_Z^{\mathbb{I}}(t) > \varepsilon_Z, \\ \frac{\pi}{2}, & \text{if } |F_X^{\mathbb{I}}(t)| \leq \varepsilon_X \quad \text{and } \left|F_Z^{\mathbb{I}}(t) - mg\right| \leq \varepsilon_Z, \end{cases}$$
$$t \geq t_0. \tag{3.46}$$

where ε_X and ε_Z are user defined constraints used to eliminate arbitrarily small reference trajectories.

The inner loop dynamics are given by

$$\dot{x}_{\text{inner}}(t) = A_{\text{inner}} x_{\text{inner}}(t) + B_{\text{inner}} \left[M_y(t) - \frac{1}{2}\rho \|v(t)\|^2 Sl \left(C_{m,0} + C_{m,\alpha}\alpha(t)\right) \right],$$
$$x_{\text{inner}}(t_0) = \left[\theta_0, \theta_{0,\text{d}}\right]^{\text{T}}, \qquad t \geq t_0, \tag{3.47}$$

where $A_{\text{inner}} \triangleq \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, $B_{\text{inner}} \triangleq \begin{bmatrix} 0 \\ I_y^{-1} \end{bmatrix}$, and $x_{\text{inner}}(t) = \begin{bmatrix} \theta(t), \dot{\theta}(t) \end{bmatrix}^{\text{T}}$.

The pitch moment is given by

$$M_y(t) = u_{\text{inner}}(t) + \frac{1}{2}\hat{\rho}\|v(t)\|^2 \hat{S}l \left(\hat{C}_{m,0} + \hat{C}_{m,\alpha}\alpha(t)\right), \qquad t \geq t_0, \tag{3.48}$$

such that the system comprising the inner loop of the QRBP is given by

$$\dot{x}_{\text{inner}}(t) = A_{\text{inner}} x_{\text{inner}}(t) + B_{\text{inner}} \left[u_{\text{inner}}(t) + \Theta_{\text{inner}}^{\text{T}} \Phi_{\text{inner}}(t) \right], \\ x_{\text{inner}}(t_0) = \left[\theta_0, \theta_{0,\text{d}} \right]^{\text{T}}, \qquad t \geq t_0. \tag{3.49}$$

Combining the dynamics equations of outer (3.42) and inner (3.47) loops yields a system in which we define the state $x_p(t) = \left[x_{\text{inner}}^{\text{T}}(t), x_{\text{outer}}^{\text{T}}(t) \right]^{\text{T}}$, the control signal is denoted by $u(t) = \left[u_{\text{inner}}^{\text{T}}(t), u_{\text{outer}}^{\text{T}}(t) \right]^{\text{T}}$, and the matrices comprising this system are

$$A_{\text{p}} = \text{blockdiag}\left(A_{\text{inner}}, A_{\text{outer}} \right), B_{\text{p}} = \begin{bmatrix} B_{\text{inner}} & 0_2 \\ 0_{4\times 1} & B_{\text{outer}} \end{bmatrix}, \text{ and } C_{\text{p}} = \begin{bmatrix} [1, 0] & 0_{1\times 2} & 0_{1\times 2} \\ 0_{2\times 2} & I_2 & 0_{2\times 2} \end{bmatrix}.$$

We define $\tilde{A} = \begin{bmatrix} 0_{m\times m} & C_{\text{p}} \\ 0_{n\times m} & A_{\text{p}} \end{bmatrix}$, $\tilde{B} = \begin{bmatrix} D_{\text{p}} \\ B_{\text{p}} \end{bmatrix}$, and $\tilde{C} = [0_{m\times m}, C_{\text{p}}]$. We let $P_{\text{LQR}} \in \mathbb{R}^{(n+m)\times(n+m)}$ denote the symmetric positive-definite solution of the *algebraic Riccati equation*

$$0 = \tilde{A}^{\text{T}} P_{\text{LQR}} + P_{\text{LQR}} \tilde{A} + Q_{\text{LQR}} - P_{\text{LQR}} \tilde{B} R^{-1} \tilde{B}^{\text{T}} P_{\text{LQR}}, \tag{3.50}$$

where $R \in \mathbb{R}^{m\times m}$ and $Q_{\text{LQR}} \in \mathbb{R}^{(n+m)\times(n+m)}$ are user-defined, symmetric, and positive-definite.

The integral tracking error is defined as $e_{y,\text{I}}(t) \triangleq \int_{t_0}^{t} [y(\tau) - y_{\text{cmd}}(\tau)] \text{d}\tau$, $t \geq t_0$, such that the augmented state vector is written $x(t) = \left[e_{y,\text{I}}^{\text{T}}(t), x_{\text{p}}^{\text{T}}(t) \right]^{\text{T}}$. We define $\tilde{A}_{\text{ref}} \triangleq \tilde{A} - \tilde{B} R^{-1} \tilde{B}^{\text{T}} P$, $\tilde{B}_{\text{ref}} = [-I_m, 0_{m\times n}]^{\text{T}}$, and $\tilde{C}_{\text{ref}} = \tilde{C} - D_{\text{p}} R^{-1} \tilde{B}^{\text{T}} P$ such that $\tilde{x}_{\text{ref}}(\cdot)$ is the solution to

$$\dot{\tilde{x}}_{\text{ref}}(t) = \tilde{A}_{\text{ref}} \tilde{x}_{\text{ref}} + \tilde{B}_{\text{ref}} y_{\text{cmd}}, \qquad \tilde{x}_{\text{ref}}(t_0) = \left[0_m^{\text{T}}, x_{\text{ref},0}^{\text{T}} \right]^{\text{T}}, \qquad t \geq t_0. \tag{3.51}$$

The augemented trajectory tracking error is defined $\tilde{e}(t) \triangleq x(t) - \tilde{x}_{\text{ref}}$. The feedback control law is given by

$$\phi_{\text{MRAC,tracking}}(\pi_{\text{tracking}}(t,x),\tilde{K}) \triangleq -K_x x + \tilde{K}^{\text{T}}\pi_{\text{tracking}}(t,x),$$
$$(t,x,\tilde{K}) \in [t_0,\infty) \times \mathbb{R}^{n+m} \times \mathbb{R}^{(n+m+N)\times m}, \quad (3.52)$$

where the LQR gain matrix is denoted $K_x = R^{-1}\tilde{B}^{\text{T}}P_{\text{LQR}}x$, and

$$\pi_{\text{tracking}}(t,x) \triangleq \left[-\left(R^{-1}\tilde{B}^{\text{T}}P_{\text{LQR}}x\right)^{\text{T}}, -\Phi^{\text{T}}(t,x_{\text{p}})\right]^{\text{T}} \quad (3.53)$$

denotes the *vector of feedback variables,*

$$\dot{\tilde{K}}(t) = \tilde{\Gamma}\pi_{\text{tracking}}(t,x(t))\tilde{e}^{\text{T}}(t)P_y\tilde{B}, \qquad \tilde{K}(t_0) = \tilde{K}_0, \qquad t \geq t_0, \quad (3.54)$$

$\tilde{\Gamma} \triangleq \text{blockdiag}\left(\Gamma_{\text{LQR}}, \Gamma_\Theta\right)$, the user-defined adaptive rate matrices $\Gamma_{\text{LQR}} \in \mathbb{R}^{(n+m)\times(n+m)}$ and $\Gamma_\Theta \in \mathbb{R}^{N\times N}$ are symmetric and positive-definite, $P_y \in \mathbb{R}^{(n+m)\times(n+m)}$ denotes the symmetric, positive-definite solution of the *algebraic Lyapunov equation*

$$0 = \tilde{A}_{\text{ref}}^{\text{T}}P_y + P_y\tilde{A}_{\text{ref}} + Q_y, \quad (3.55)$$

and $Q_y \in \mathbb{R}^{(n+m)\times(n+m)}$ is symmetric, positive-definite, and user-defined.

The thrust force is computed as

$$\hat{T}(t) \triangleq \begin{cases} \dfrac{\sqrt{\left(F_X^{\mathbb{I}}(t)\right)^2 + \left(F_Z^{\mathbb{I}}(t)\right)^2}}{\cos\left(\theta_{\text{cmd}}(t) - \theta(t)\right)}, & \text{if } |\theta_{\text{cmd}}(t) - \ \theta(t)| < \theta_{\max} \quad \text{and} \quad F_Z^{\mathbb{I}}(t) \leq 0, \qquad t \geq t_0, \\ \dfrac{\left|F_X^{\mathbb{I}}(t)\right|}{\cos\left(\theta_{\text{cmd}}(t) - \theta(t)\right)}, & \text{if } |\theta_{\text{cmd}}(t) - \ \theta(t)| < \theta_{\max} \quad \text{and} \quad F_Z^{\mathbb{I}}(t) > 0, \\ \sqrt{\left(F_X^{\mathbb{I}}(t)\right)^2 + \left(F_Z^{\mathbb{I}}(t)\right)^2}, & \text{otherwise,} \end{cases} \tag{3.56}$$

where $\theta_{\max} \in \left(0, \frac{\pi}{2}\right)$ is a user-defined value that specifies a maximum allowable error in the QRBP's pitch. The thrust force $\hat{T}(t)$, $t \geq t_0$, and the moment of the thrust force $M_y(t)$ are realized by setting

$$T_{\text{upper}}(t) = \begin{cases} 0, & \text{if } \hat{T}(t) < \dfrac{M_y(t)}{l} \text{ and } M_y(t) > 0, \qquad t \geq t_0, \\ -\dfrac{M_y(t)}{l} & \text{if } \hat{T}(t) < -\dfrac{M_y(t)}{l} \text{ and } M_y(t) < 0, \\ \dfrac{\hat{T}(t)}{2} - \dfrac{M_y(t)}{2l}, & \text{if } \hat{T}(t) \geq \dfrac{|M_y(t)|}{l} \text{ and } |\theta_{\text{cmd}}(t) - \theta(t)| \leq \theta_{\max}, \\ 0, & \text{if } \hat{T}(t) \geq \dfrac{|M_y(t)|}{l} \text{ and } \theta_{\text{cmd}}(t) - \theta(t) > \theta_{\max}, \\ \dfrac{|M_y(t)|}{l}, & \text{if } \hat{T}(t) \geq \dfrac{|M_y(t)|}{l} \text{ and } \theta_{\text{cmd}}(t) - \theta(t) < -\theta_{\max}, \end{cases} \tag{3.57}$$

and

$$T_{\text{lower}}(t) = \begin{cases} \dfrac{M_y(t)}{l}, & \text{if } \hat{T}(t) < \dfrac{M_y(t)}{l} \text{ and } M_y(t) > 0, \\ 0 & \text{if } \hat{T}(t) < -\dfrac{M_y(t)}{l}, \text{ and } M_y(t) < 0, \\ \dfrac{\hat{T}(t)}{2} + \dfrac{M_y(t)}{2l}, & \text{if } \hat{T}(t) \geq \dfrac{|M_y(t)|}{l} \text{ and } |\theta_{\text{cmd}}(t) - \theta(t)| \leq \theta_{\max}, \\ \dfrac{|M_y(t)|}{l}, & \text{if } \hat{T}(t) \geq \dfrac{|M_y(t)|}{l} \text{ and } \theta_{\text{cmd}}(t) - \theta(t) > \theta_{\max}, \\ 0, & \text{if } \hat{T}(t) \geq \dfrac{|M_y(t)|}{l} \text{ and } \theta_{\text{cmd}}(t) - \theta(t) < -\theta_{\max}. \end{cases} \tag{3.58}$$

3.5.2 Lateral Control

In order to control the lateral states of the QRBP, the following novel switched adaptive controller is employed. This architecture is employed to handle the different dynamics of the QRBP as it operates in quadcopter and biplane mode. An additional facet of this novel controller, is its integration into the Longitudinal Controller presented in Section 3.5.1.

The implemented lateral control architecture produces a force along the Y direction of the inertial reference frame, denoted $Y^{\mathbb{I}}$, such that the QRBP tracks the commanded trajectory along $Y^{\mathbb{I}}$. The dynamics are given by

$$\begin{bmatrix} \dot{y}_A(t) \\ \ddot{y}_A(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y_A(t) \\ \dot{y}_A(t) \end{bmatrix} + \begin{bmatrix} 0 \\ m^{-1} \end{bmatrix} \left[F_Y(t) + \Theta^{\mathrm{T}} \Phi(t, \dot{y}(t)) \right], \qquad t \geq t_0,$$

$$\begin{bmatrix} y_A(t_0) \\ \dot{y}_A(t_0) \end{bmatrix} = \begin{bmatrix} y_{A,0} \\ \dot{y}_{A,0} \end{bmatrix}, \tag{3.59}$$

where $y_A : [t_0, \infty) \to \mathbb{R}$ denotes the UAV's position along $Y^{\mathbb{I}}$, $m > 0$ denotes the UAV's

mass, $\Theta = C_S \rho S$, $C_S > 0$ denotes the *lateral drag coefficient*, $\rho > 0$ denotes the *air density*,

$$\Theta = -\rho S \left[C_D, C_S, C_L\right]^{\mathrm{T}} \in \mathbb{R}^3, \tag{3.60}$$

$$\Phi(t, \dot{y}(t)) = \|v(t)\| \begin{bmatrix} R_{2,1}(\phi(t), \theta(t), \psi(t)) v_1(t) \\ R_{2,2}(\phi(t), \theta(t), \psi(t)) v_2(t) \\ R_{2,3}(\phi(t), \theta(t), \psi(t)) v_3(t) \end{bmatrix} \in \mathbb{R}^3, \tag{3.61}$$

$R_{i,j}(\cdot)$ denotes the element of $R^{\mathbb{I}}_{\mathbb{J}}(\cdot)$ on the i-th row and j-th column, and

$$R^{\mathbb{I}}_{\mathbb{J}}(\phi, \theta, \psi) \triangleq \begin{bmatrix} \cos\psi(t) & -\sin\psi(t) & 0 \\ \sin\psi(t) & \cos\psi(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta(t) & 0 & \sin\theta(t) \\ 0 & 1 & 0 \\ -\sin\theta(t) & 0 & \cos\theta(t) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi(t) & -\sin\phi(t) \\ 0 & \sin\phi(t) & \cos\phi(t) \end{bmatrix},$$
$$(\phi, \theta, \psi) \in [0, 2\pi) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times [0, 2\pi). \tag{3.62}$$

Consider the reference model

$$\begin{bmatrix} \dot{y}_{A,\mathrm{ref}}(t) \\ \ddot{y}_{A,\mathrm{ref}}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_{p,A} & -k_{d,A} \end{bmatrix} \begin{bmatrix} y_{A,\mathrm{ref}}(t) \\ \dot{y}_{A,\mathrm{ref}}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} y_{\mathrm{cmd}}(t), \qquad t \geq t_0,$$
$$\begin{bmatrix} y_{A,\mathrm{ref}}(t_0) \\ \dot{y}_{A,\mathrm{ref}}(t_0) \end{bmatrix} = \begin{bmatrix} y_{A,\mathrm{ref},0} \\ \dot{y}_{A,\mathrm{ref},0}, \end{bmatrix}, \tag{3.63}$$

where $k_{p,A}, k_{d,A} > 0$ are user-defined. If $m > 0$ were known, then matching conditions

$$\begin{bmatrix} 0 & 1 \\ -k_{p,A} & -k_{d,A} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ m^{-1} \end{bmatrix} K_x^{\mathrm{T}}, \tag{3.64}$$

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ m^{-1} \end{bmatrix} K_r, \tag{3.65}$$

can be verified explicitly by setting

$$K_x = -m \left[k_{p,A}, k_{d,A}\right]^{\mathrm{T}} \in \mathbb{R}^2, \tag{3.66}$$

$$K_r = m. \tag{3.67}$$

Thus, we set

$$F_Y(t) = -mk_{p,A}y_A(t) - mk_{d,A}\dot{y}_A(t) + my_{\mathrm{cmd}}(t) - \tilde{\Theta}^{\mathrm{T}}\Phi(t, \dot{y}(t)) + \hat{K}^{\mathrm{T}}(t)\Phi(t, \dot{y}_A(t)), \quad t \geq t_0, \tag{3.68}$$

where $\tilde{\Theta} \in \mathbb{R}^3$ denotes an estimate of Θ,

$$\dot{\hat{K}}(t) = -\Gamma\Phi(t, y_{\mathrm{A}}(t))e_y^{\mathrm{T}}P_y \begin{bmatrix} 0 \\ m^{-1} \end{bmatrix}, \tag{3.69}$$

the matrix $P \in \mathbb{R}^{2\times 2}$ is the symmetric, positive-definite solution of the algrebraic Lyapunov

equation

$$\begin{bmatrix} 0 & 1 \\ -k_{p,A} & -k_{d,A} \end{bmatrix}^{\mathrm{T}} P_y + P_y \begin{bmatrix} 0 & 1 \\ -k_{p,A} & -k_{d,A} \end{bmatrix} = -Q_y, \tag{3.70}$$

the matrix $Q_y \in \mathbb{R}^{2\times 2}$ is user-defined, symmetric, and positive-definite, the *adaptive rate matrix* $\Gamma \in \mathbb{R}^{3\times 3}$ is symmetrix, positive-definite, and $e_y(t) \triangleq \begin{bmatrix} y_A(t) \\ \dot{y}_A(t) \end{bmatrix} - \begin{bmatrix} y_{A,\mathrm{ref}}(t) \\ \dot{y}_{A,\mathrm{ref}}(t) \end{bmatrix}$.

Since the UAV's $y(\cdot)$ axis and the inertial $Y^{\mathbb{I}}$ axes are not necessarily coaxial, we let

$$\begin{bmatrix} \tilde{F}_X(t) \\ \tilde{F}_Y(t) \\ F_Z(t) \end{bmatrix} \triangleq \begin{bmatrix} \cos\phi(t) & \sin\phi(t) & 0 \\ -\sin\phi(t) & \cos\phi(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_X(t) \\ F_Y(t) \\ F_Z(t) \end{bmatrix}, \qquad t \geq t_0, \tag{3.71}$$

where $F_X(t)$ and $F_Z(t)$ are calculated using (3.43) and (3.44) respectively.

Thus, we compute the reference yaw roll angle as

$$\psi_{\mathrm{cmd}}(t) \triangleq \begin{cases} \mathrm{atan2}\left(F_Z(t), \tilde{F}_y(t)\right) - \frac{\pi}{2}, & \text{if } \left|\tilde{F}_y(t)\right| > \varepsilon_Y \quad \text{and } -F_Z(t) + mg > \varepsilon_Z, \\ \mathrm{atan2}\left(|F_Z(t)|, \tilde{F}_y(t)\right) - \frac{\pi}{2}, & \text{if } \left|\tilde{F}_y(t)\right| > \varepsilon_Y \quad \text{and } -F_Z(t) + mg \leq -\varepsilon_Z, \\ \mathrm{atan2}\left(|F_Z(t)|, \tilde{F}_y(t)\right) - \frac{\pi}{2}, & \text{if } \left|\tilde{F}_y(t)\right| > \varepsilon_Y \quad \text{and } |mg - F_Z(t)| < \varepsilon_Z, \\ \mathrm{atan2}\left(|F_Z(t)|, \tilde{F}_y(t)\right) - \frac{\pi}{2}, & \text{if } \left|\tilde{F}_y(t)\right| \leq \varepsilon_Y \quad \text{and } -F_Z(t) + mg \geq \varepsilon_Z, \\ 0, & \text{if } \left|\tilde{F}_y(t)\right| \leq \varepsilon_Y \quad \text{and } -mg + F_Z(t) > \varepsilon_Z, \\ 0, & \text{if } |\tilde{F}_y(t)| \leq \varepsilon_Y \quad \text{and } |F_Z(t) - mg| \leq \varepsilon_Z, \end{cases} \tag{3.72}$$

where ε_Y and ε_Z are user defined constraints used to eliminate arbitrarily small reference

trajectories. Considering the QRBP's roll dynamics, the reference signal $\Phi_{\text{cmd}}(t) \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ is user-defined and arbitrary.

We note that the angular kinematics is given by

$$\begin{bmatrix} \dot{\phi}(t) \\ \dot{\psi}(t) \end{bmatrix} = B_{\text{kin},\sigma(\text{t})} \begin{bmatrix} p(t) \\ r(t) \end{bmatrix}, \qquad \begin{bmatrix} \phi(t_0) \\ \psi(t_0) \end{bmatrix} = \begin{bmatrix} \phi_0 \\ \psi_0 \end{bmatrix}, \qquad t \geq t_0, \tag{3.73}$$

where

$$B_{\text{kin},1} = \begin{bmatrix} 1 & \tan\theta_{\text{e},\sigma(\text{t})} \\ 0 & 1 \end{bmatrix}, \tag{3.74}$$

$$B_{\text{kin},2} = \mathbf{1}_2, \tag{3.75}$$

the subscript "1" denotes the biplane mode, and the subscript "2" denotes the quadcopter mode. If the UAV is tasked with flying in biplane mode, then the UAV's pitch angle at trim is assumed to be given by $\theta_{\text{e},1} \in \left(-\overline{\theta}, \overline{\theta}\right)$, where $\overline{\theta} \in (0, \pi/2)$.

Thus, we choose

$$\begin{bmatrix} p_{\text{cmd}}(t) \\ r_{\text{cmd}}(t) \end{bmatrix} = B^{-1}_{\text{kin},\sigma(\text{t})} \left[\begin{bmatrix} \dot{\phi}_{\text{cmd}}(t) \\ \dot{\psi}_{\text{cmd}}(t) \end{bmatrix} - K_{P,\text{kin}} \left(\begin{bmatrix} \phi(t) \\ \psi(t) \end{bmatrix} - \begin{bmatrix} \phi_{\text{cmd}}(t) \\ \psi_{\text{cmd}}(t) \end{bmatrix} \right) \right], \qquad t \geq t_0, \tag{3.76}$$

where $K_{P,\text{kin}} \in \mathbb{R}^{2\times 2}$ is user-defined, symmetric, and positive-definite and

$$B^{-1}_{\text{kin},1} = \begin{bmatrix} 1 & -\tan\theta_{\text{e},\sigma(\text{t})} \\ 0 & 1 \end{bmatrix}, \tag{3.77}$$

$$B^{-1}_{\text{kin},2} = \mathbf{1}_2. \tag{3.78}$$

Next, we design $M_x, M_z : [t_0, \infty) \to \mathbb{R}$ such that $\lim_{t\to\infty} \left\| \begin{bmatrix} p(t) \\ r(t) \end{bmatrix} - \begin{bmatrix} p_{\text{cmd}}(t) \\ r_{\text{cmd}}(t) \end{bmatrix} \right\| = 0$ uniformly in $t_0 \geq 0$. To this goal, note that

$$\begin{bmatrix} \dot{p}(t) \\ \dot{r}(t) \end{bmatrix} = A_{\text{dyn},\sigma(t)} \begin{bmatrix} p(t) \\ r(t) \end{bmatrix} + B_{\text{dyn}} \begin{bmatrix} M_x(t) \\ M_z(t) \end{bmatrix}, \qquad \begin{bmatrix} p(t_0) \\ r(t_0) \end{bmatrix} = \begin{bmatrix} p_0 \\ r_0 \end{bmatrix}, \qquad t \geq t_0, \tag{3.79}$$

where

$$A_{\text{dyn},1} \triangleq \begin{bmatrix} -\frac{I_{xz}}{I_x z^2 - I_x I_z}\frac{\partial N}{\partial p} - \frac{I_z}{I_{xz}^2 - I_x I_z}\frac{\partial L}{\partial p} & -\frac{I_{xz}}{I_{xz}^2 - I_x I_z}\frac{\partial N}{\partial r} - \frac{I_z}{I_{xz}^2 - I_x I_z}\frac{\partial L}{\partial r} \\ \frac{-I_{xz}}{I_{xz}^2 - I_x I_z}\frac{\partial L}{\partial p} - \frac{I_x}{I_{xz}^2 - I_x I_z}\frac{\partial N}{\partial p} & -\frac{I_{xz}}{I_{xz}^2 - I_x I_z}\frac{\partial L}{\partial r} - \frac{I_x}{I_{xz}^2 - I_x I_z}\frac{\partial N}{\partial r} \end{bmatrix}, \tag{3.80}$$

$$A_{\text{dyn},2} \triangleq 0_{2\times 2}, \tag{3.81}$$

$$B_{\text{dyn}} \triangleq \begin{bmatrix} -\frac{I_z}{I_{xz}^2 - I_x I_z} & -\frac{I_{xz}}{I_{xz}^2 - I_x I_z} \\ -\frac{I_{xz}}{I_{xz}^2 - I_x I_z} & -\frac{I_x}{I_{xz}^2 - I_x I_z} \end{bmatrix}. \tag{3.82}$$

Furthermore, we consider the reference model

$$\begin{bmatrix} \dot{p}_{\text{ref}}(t) \\ \dot{r}_{\text{ref}}(t) \end{bmatrix} = A_{\text{dyn,ref},\sigma(t)} \begin{bmatrix} p_{\text{ref}}(t) \\ r_{\text{ref}}(t) \end{bmatrix} + B_{\text{dyn}} \rho_{\text{cmd}}(t), \qquad \begin{bmatrix} p_{\text{ref}}(t_0) \\ r_{\text{ref}}(t_0) \end{bmatrix} = \begin{bmatrix} p_{\text{ref},0} \\ r_{\text{ref},0} \end{bmatrix}, \qquad t \geq t_0, \tag{3.83}$$

where $A_{\text{dyn,ref},\sigma} \in \mathbb{R}^{2\times 2}$ is a full matrix (just like $A_{\text{dyn},\sigma}$) and Hurwitz, and

$$\rho_{\text{cmd}}(t) = B_{\text{dyn}}^{-1} \left(-A_{\text{dyn,ref},\sigma(t)} \begin{bmatrix} p_{\text{cmd}}(t) \\ r_{\text{cmd}}(t) \end{bmatrix} + \begin{bmatrix} \dot{p}_{\text{cmd}}(t) \\ \dot{r}_{\text{cmd}}(t) \end{bmatrix} \right) \in \mathbb{R}^2, \tag{3.84}$$

$$\begin{bmatrix} \dot{p}_{\text{cmd}}(t) \\ \dot{r}_{\text{cmd}}(t) \end{bmatrix} = B_{\text{kin},\sigma(t)}^{-1} \left[\begin{bmatrix} \ddot{\phi}_{\text{cmd}}(t) \\ \ddot{\psi}_{\text{cmd}}(t) \end{bmatrix} - K_{P,\text{kin}} \left(\begin{bmatrix} \dot{\phi}(t) \\ \dot{\psi}(t) \end{bmatrix} - \begin{bmatrix} \dot{\phi}_{\text{cmd}}(t) \\ \dot{\psi}_{\text{cmd}}(t) \end{bmatrix} \right) \right]. \tag{3.85}$$

Thus, we set

$$K_{\mathrm{PD},\sigma(\mathrm{t})} = \begin{bmatrix} K_{\mathrm{p}\phi,\sigma(\mathrm{t})} & 0 & K_{\mathrm{d}\phi,\sigma(\mathrm{t})} & 0 \\ 0 & K_{\mathrm{p}\psi,\sigma(\mathrm{t})} & 0 & K_{\mathrm{d}\psi,\sigma(\mathrm{t})} \end{bmatrix}$$

such that

$$\begin{bmatrix} M_x(t) \\ M_z(t) \end{bmatrix} = -K_{\mathrm{PD},\sigma(\mathrm{t})} \begin{bmatrix} \phi(t) \\ \psi(t) \\ p(t) \\ r(t) \end{bmatrix} + \hat{K}_{\mathrm{dyn}}^{T}(t) \begin{bmatrix} p(t) \\ r(t) \\ \rho_{\mathrm{cmd}}(t) \end{bmatrix}, \tag{3.86}$$

where $K_{\mathrm{p}\phi,\sigma(\mathrm{t})}, K_{\mathrm{d}\phi,\sigma(\mathrm{t})}, K_{\mathrm{p}\psi,\sigma(\mathrm{t})}, K_{\mathrm{d}\psi,\sigma(\mathrm{t})}$ are user defined for $\sigma \in \{1,2\}$ the adaptive gain $\hat{K}_{\mathrm{dyn}} : [t_0, \infty) \to \mathbb{R}^{4\times 2}$ is such that

$$\dot{\hat{K}}_{\mathrm{dyn}}(t) = -\Gamma^{-1} \begin{bmatrix} p(t) \\ r(t) \\ \rho_{\mathrm{cmd}}(t) \end{bmatrix} \left(\begin{bmatrix} p(t) \\ r(t) \end{bmatrix} - \begin{bmatrix} p_{\mathrm{ref}}(t) \\ r_{\mathrm{ref}}(t) \end{bmatrix} \right)^{\mathrm{T}} P_{\mathrm{dyn}} B_{\mathrm{dyn}}, \qquad \hat{K}_{\mathrm{dyn}}(t_0) = \hat{K}_{\mathrm{dyn},0}, \tag{3.87}$$

$\Gamma \in \mathbb{R}^{4\times 4}$ is user-defined, symmetric, and positive-definite, and $P_{\mathrm{dyn}} \in \mathbb{R}^{2\times 2}$ is symmetric, positive-definite, and verifies the *Lyapunov inequality*

$$A_{\mathrm{dyn,ref},s}^{\mathrm{T}} P_{\mathrm{dyn}} + P_{\mathrm{dyn}} A_{\mathrm{dyn,ref},s} \leq -Q_{\mathrm{dyn}} \tag{3.88}$$

for all $s \in \{1,2\}$ with $Q_{\mathrm{dyn}} \in \mathbb{R}^{2\times 2}$ user-defined, symmetric, and positive-definite. The thrust force to be produced by the upper pair of propellers $T_{\mathrm{u}} : [0,\infty) \to \mathbb{R}$, the thrust force to be produced by the lower pair of propellers $T_{\mathrm{l}} : [0,\infty) \to \mathbb{R}$, as calculated via the

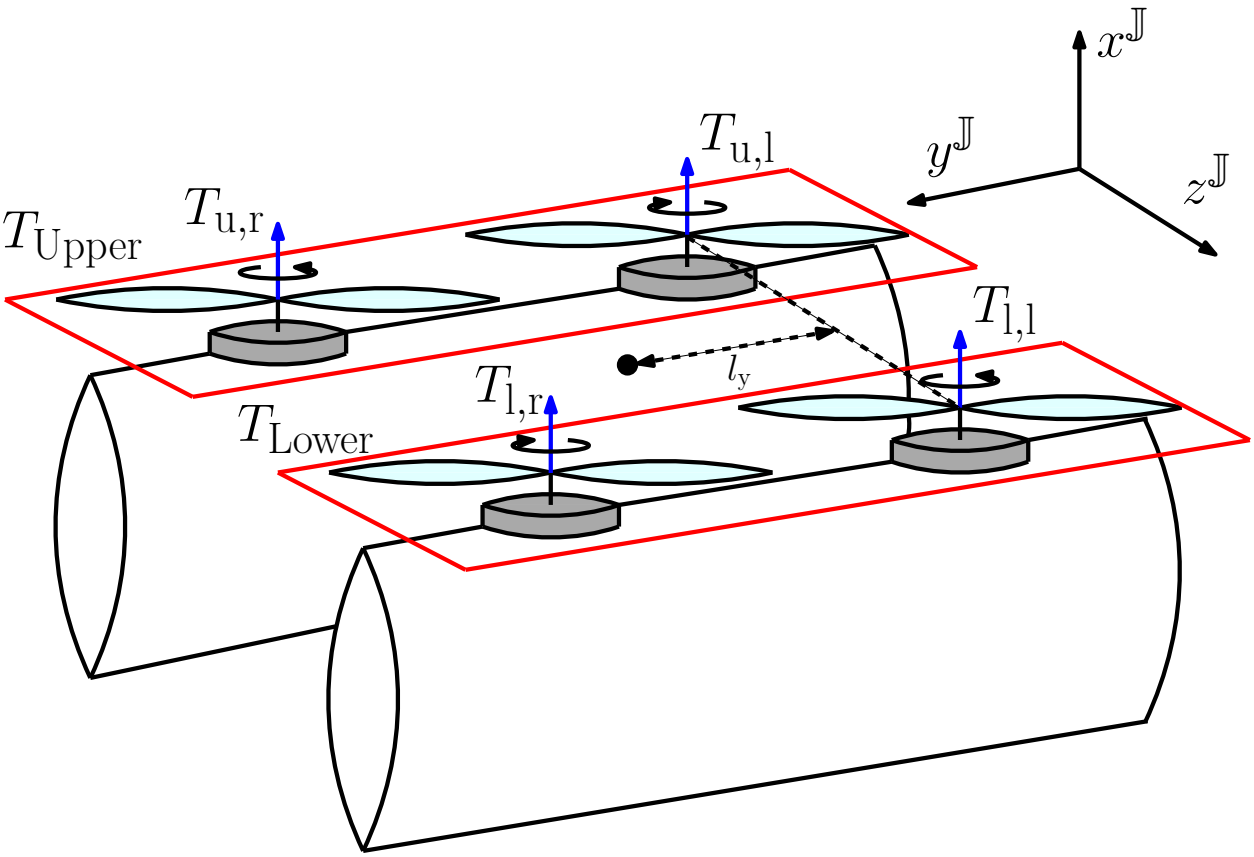


Figure 3.4: Schematic representation of the thrust forces acting on a quadbiplane UAV. The vector $T_{\text{u,l}}(\cdot)$ denotes the thrust force produced by the left propeller of the upper pair, $T_{\text{u,r}}(\cdot)$ denotes the thrust force produced by the right propeller of the upper pair, $T_{\text{l,r}}(\cdot)$ denotes the thrust force produced by the right propeller of the lower pair, $T_{\text{l,l}}(\cdot)$ denotes the thrust force produced by the left propeller of the lower pair, the upper left propeller and the lower right propeller are assumed to rotate clockwise and the lower left propeller and the upper right propeller are assumed to rotate counter-clockwise.

controller in Section 3.5.1, and the moments of the thrust force about the roll and yaw axes $M_x, M_z : [0, \infty) \to \mathbb{R}$, the thrust force produced by each propeller can be computed as

$$T(t) = M^{-1} \begin{bmatrix} T_{\text{u}}(t) \\ T_{\text{l}}(t) \\ M_x(t) \\ M_z(t) \end{bmatrix}, \qquad t \geq 0, \tag{3.89}$$

where $T(t) \triangleq [T_{\mathrm{u,l}}(t), T_{\mathrm{u,r}}(t), T_{\mathrm{l,r}}(t), T_{\mathrm{l,l}}(t)]^{\mathrm{T}}$, the mixer matrix is given by

$$M \triangleq \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ -c_{\mathrm{T}} & c_{\mathrm{T}} & -c_{\mathrm{T}} & c_{\mathrm{T}} \\ l_{\mathrm{y}} & -l_{\mathrm{y}} & -l_{\mathrm{y}} & l_{\mathrm{y}} \end{bmatrix}, \tag{3.90}$$

$c_{\mathrm{T}} \triangleq c_{\mathrm{D}} \rho D^5$, $c_{\mathrm{D}} > 0$ denotes the *drag coefficient*, l_{y} denotes the distance of the rotors from the vehicle's center of mass along the $y^{\mathbb{J}}$ axis, and $D > 0$ denotes the *propeller's diameter*. A diagram is provided of the corresponding layout of the distribution of thrust in Figure 3.4.

3.6 Conclusion

The MRAC theory presented in this chapter is the basis of the proposed control algorithm for the QRBP. In its simplest form, MRAC allows for the control of a nonlinear system to adapt to unknown perturbations and modeling errors. The addition of output state tracking is the foundation for the proposed longitudinal control algorithm of the QRBP. Similarly, the theory of MRAC for dynamically switched systems is employed for the lateral controller of the QRBP. We detail the implementation of these theories with models of the longitudinal and lateral dynamics of a QRBP. In effect, we provide the control architecture of a QRBP that can effectively actuate user defined position commands. In future applications, this ability to maneuver the QRBP could be paired with a guidance system, such as the one presented in Chapter 2. The tools developed to refine the implementation of the proposed architecture are presented in Chapter 4 and the validation of the proposed architecture is presented in Chapter 5.

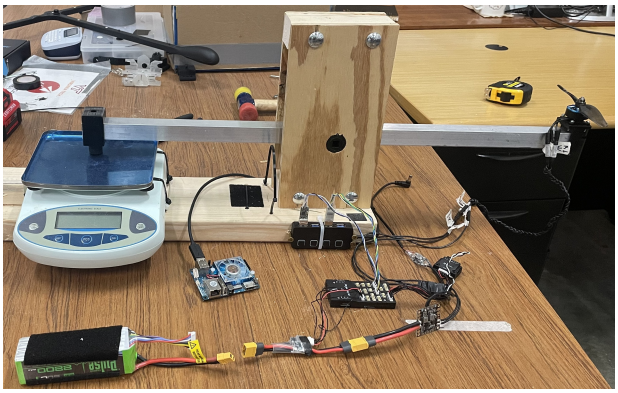
Chapter 4

Refinement of Experimental Process

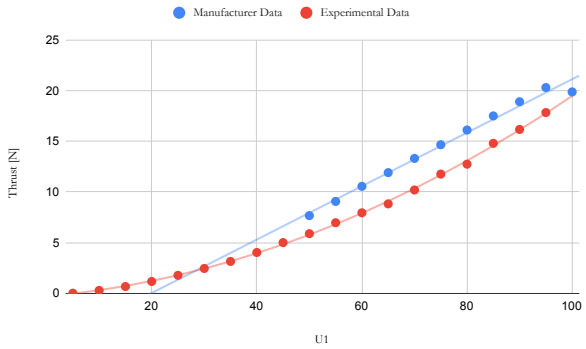
4.1 Introduction

Efforts to streamline the validation of the control algorithm presented in Chapter 3 with real-world flight tests present unique challenges. This chapter details those challenges and the solutions proposed to resolve them.

Flying the QRBP requires accurate thrust realization, the tuning of many user-defined parameters, and a particularly robust physical design of the vehicle. The motion of the QRBP is actuated purely by thrust, which cannot be directly controlled with the available flight-companion computer that executes the proposed control system. In the current architecture, the companion computer is interfaced to each motor through a micro-controller and a dedicated ESC (electronic speed controller). A procedure and mechanism are employed such that the output of the controller, namely $[T_{\text{upper}}, T_{\text{lower}}, M_x, M_z]^{\text{T}}$ discussed in Section 3.5.2, can be accurately mapped to the normalized control inputs of the micro-controller $[U_1, U_2, U_3, U_4]^{\text{T}} \in \mathbb{R}^4$, where $U_1 \in [0, 1]$ denotes the desired total thrust and $U_2, U_3, U_4 \in [-0.5, 0.5]$ denote the desired roll, pitch, and yaw moment of the total thrust in the micro-controller's frame, respectively. In a typical flight, a poor quality controller response results in the UAV crashing. To mitigate this likelihood, and restrict the degrees of freedom of the UAV, we designed apparatus to streamline the tuning process.



(a)



(b)

Figure 4.1: (Left)An early iteration of the thrust measurement stand. Due to limited machining access, the lever arm swings on 3D printed bushings and the plywood comprising the super structure is bolted in such a way to allow the plate to be realigned as necessary to avoid the apparatus binding up. The electronics necessary to run the experiment are included as they are identical to those used on the QRBP. (Right) A comparison of the thrust generated by a particular throttle input and the data provided by the motor's manufacturer. The experimental thrust values are significantly lower than the manufacturer's data and the relation of throttle to resulting thrust is not linear.

This chapter is structured as follows. Section 4.2 shows the development of a mechanism and process used to effectively characterize the actuation of commercial-off-the-shelf (COTS) UAV components. Section 4.3 details the development and implementation of a static structure from which to test the UAV's rotational dynamics. Section 4.4 details the physical design of the QRBP such that it is capable of handling flight. Finally, Section 4.5 draws conclusions on the material presented in this chapter.

4.2 The Thrust Stand

It is customary for manufacturers of brushless motors for UAVs to provide thrust curves for multiple, common types of propellers. Before attempting flight tests, the thrust curves for the chosen motors, an AirA 2508, with a 7045 propeller, were verified employing the

thrust stand designed as part of this research and shown in Figure 4.1a. As it appears in Figure 4.1b, the manufacturer data resulted to be inaccurate, especially for larger values of the normalized values of the input parameter U_1, which captures the desired thrust force.

This section is structured as follows. Section 4.2.1 provides a brief overview of the system to be tested and the motivation for conducting accurate thrust measurements. In Section 4.2.2, we describe the process by which we map throttle inputs to resulting thrust forces. In Section 4.2.3, we present the results of this experiment. Section 4.2.4 provides a discussion on the results of this effort.

4.2.1 A Brief Overview

To operate autonomously, with the only user-input being a pre-determined trajectory, the onboard control system of the QRBP features a Pixhawk micro-controller and an Odroid-XU4 single board computer. The Odroid-XU4 receives estimates on the vehicle's state from the Pixhawk, executes the control system as described in Chapter 3, and translates the resulting forces and moments into a series of throttle values. The Pixhawk executes the *PX4 Autopilot* operating system and effectively acts as the control system's interface with the world, in that it filters the various sensor data into the vehicle state and applies power via the ESCs to the motors. A diagram of this architecture is provided in Figure 4.2.

The control system discussed in Chapter 3 determines $[T_{\text{upper}}(t), T_{\text{lower}}(t), M_x(t), M_z(t)]^{\text{T}}$, $t \geq t_0$, where $T_{\text{upper}}(\cdot)$ denotes the thrust of the two rotors located along the $-z^{\mathbb{J}}$ axis(see Figure 4.3), $T_{\text{lower}}(\cdot)$ denotes the thrust of the two rotors located along the $z^{\mathbb{J}}$ axis, $M_x(\cdot)$ denotes the moment to be generated about the $x^{\mathbb{J}}$ axis and $M_z(\cdot)$ denotes the moment to be generated about the $z^{\mathbb{J}}$ axis. The dimensional input to the micro-controller is given by the

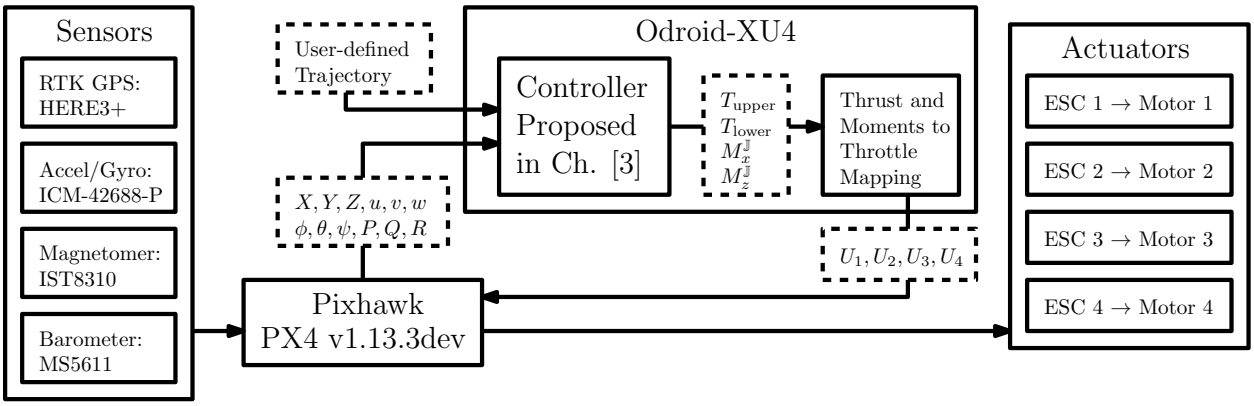


Figure 4.2: Diagram of the electronics comprising the control system of the QRBP. Boxes with a dashed border indicate values and boxes with solid borders indicate physical components or algorithms. The micro-controller, a Pixhawk 6c, combines and then filters the data received from the listed sensors. This data is provided to the control algorithm described in Chapter 3 which returns the normalized signals $[U_1, U_2, U_3, U_4]^{\mathrm{T}}$. The micro-controller applies these throttle values to the motors via the ESC to each motor.

total thrust $T \in \mathbb{R}$, and the moment of the thrust $[M_x^{\mathbb{P}}, M_y^{\mathbb{P}}, M_z^{\mathbb{P}}]^{\mathrm{T}} \in \mathbb{R}^3$ so that

$$\begin{bmatrix} T \\ M_x^{\mathbb{P}} \\ M_y^{\mathbb{P}} \\ M_z^{\mathbb{P}} \end{bmatrix} = M \begin{bmatrix} T_{\text{upper}} \\ T_{\text{lower}} \\ M_x \\ M_z \end{bmatrix}, \tag{4.1}$$

where the *mixer* matrix is given by

$$M \triangleq \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -l_x & -l_x & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{4.2}$$

the distance from the center of the vehicle to the rotors along the $x^{\mathbb{P}}$ axis is denoted by l_x and the superscript $\mathbb{P}$ denotes the local reference frame of the micro-controller. A diagram of

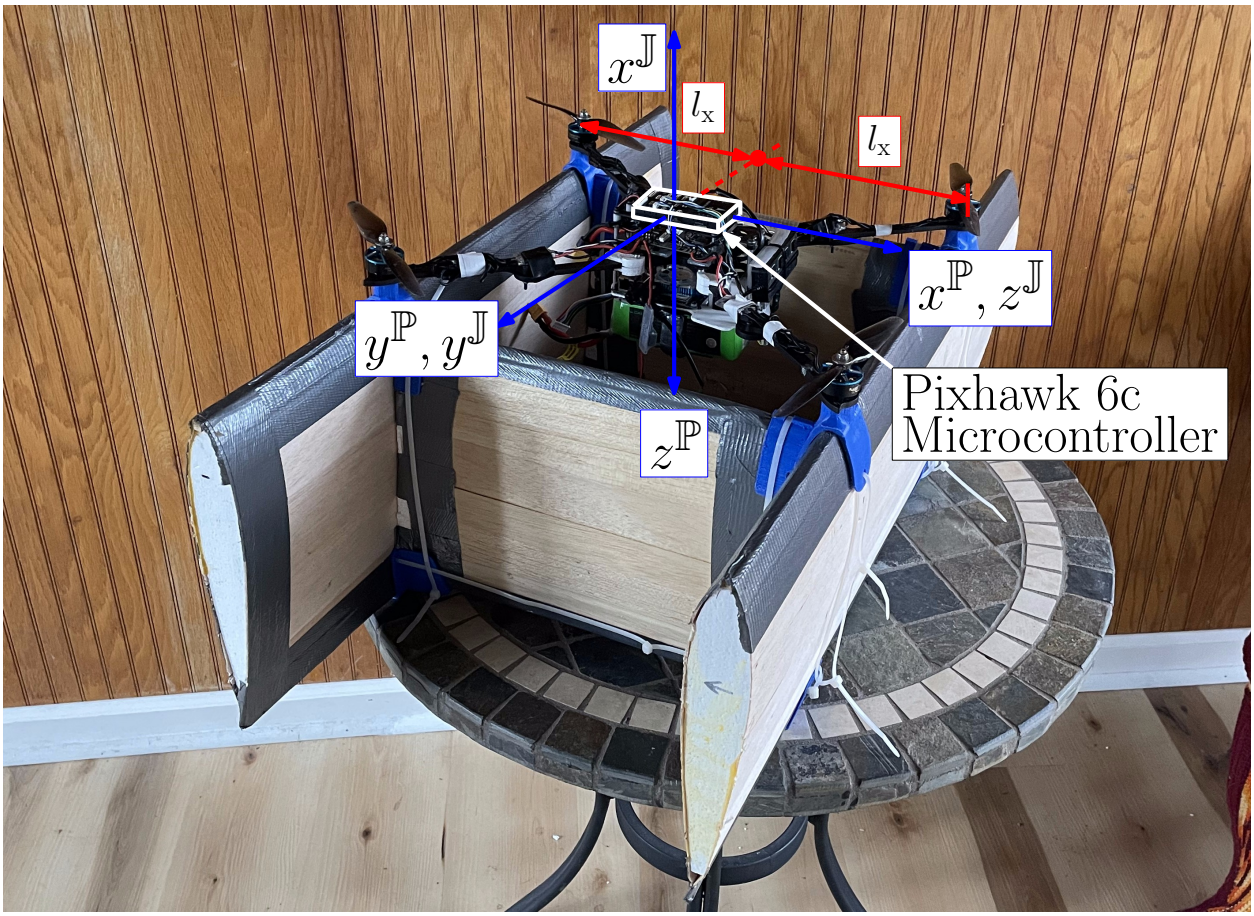
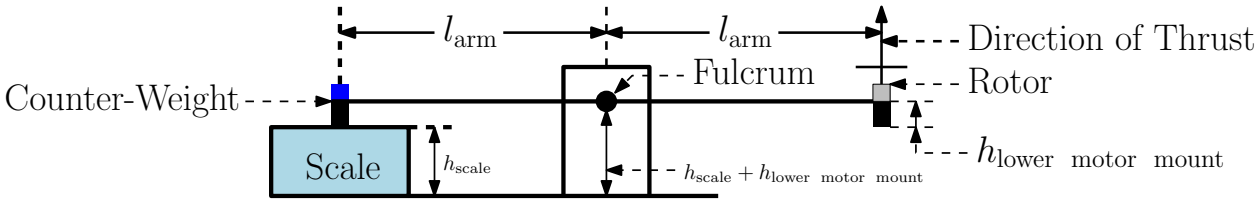


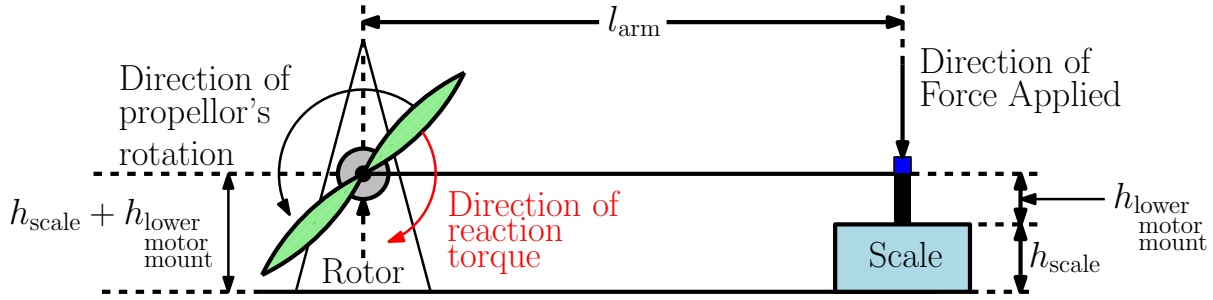
Figure 4.3: Diagram representation of the fixed relation of the $\mathbb{J}$ and $\mathbb{P}$ reference frames of the QRBP.

the relation between the QRBP's reference frame and the micro-controller's reference frame, along with l_{x} is shown in Figure 4.3.

The control signal U_1 is an input to the micro-controller for the normalized thrust force of the entire vehicle. We measure the effect of this control input by measuring the force produced when various values of U_1 are applied. To measure the moment created by the control signals of $[U_2, U_3, U_4]^{\mathrm{T}}$, we measure the underlying force, and, from there, deduce the generated moment. With the knowledge of how $[U_2, U_3, U_4]^{\mathrm{T}}$ can impact the thrust of each motor, we deduce the resulting moments from the geometry of the vehicle. The measurement of the thrust forces and moments will unavoidably suffer from errors. However, these can be



(a) Diagram of the stand used to measure the thrust generated by varying values of $[U_1, U_2, U_3]$. Special attention is paid so that the center of the rotor and the the point of the test-arm that contacts the scale are equi-distant from the fulcrum point. The test arm is rigid and balanced with a counterweight so that it remains approximately level in the absence of external forces. To reduce aerodynamic interference such as *ground effect*, the stand is positioned so that the air traveling through rotor is unimpeded by foreign objects such as furniture.



(b) Diagram of the stand used to measure the resulting moment of varying values of $[U_1, U_4]$. Special attention is paid to the geometry of l_{arm} such that it should be level when force is applied

Figure 4.4

reasonably modeled as parametric uncertainties, and hence, accounted for by the proposed MRAC systems. Additionally, we did not have access to a power supply that is capable of providing a constant voltage source and are reliant upon a battery. Thereby, while not a primary concern, it is considered during testing that we operate within a certain operating range in terms of voltage.

A diagram of the contraption used to take measurements of resulting thrust, as used to measure the impacts of $[U_1, U_2, U_3]^{\mathrm{T}}$, is shown in Figure 4.4a. Similarly, a diagram of the contraption used to take measurements of resulting moment, as used to measure the impacts of $[U_1, U_4]^{\mathrm{T}}$, is shown in Figure 4.4b.

For this experiment, we employed the *Racerstar AirA 2508* 1200KV motor paired with a propeller with a 7" diameter and 4.5" of pitch. Mirroring the electronics of the QRBP, we used a *Holybro PM07* power distribution board and *T-Motor F35A* 35 Ampere ESC. We employ a digital scale capable of 0.1g precision to measure force. For the test stand designed to measure thrust, $l_{\text{arm}} = 0.28\text{m}$. For the test stand designed to measure reaction torque, $l_{\text{arm}} = 0.085\text{m}$.

4.2.2 Process

The procedure used to measure the relationship between the inputs $[T(\cdot), M_x^{\mathbb{P}}(\cdot), M_y^{\mathbb{P}}(\cdot), M_z^{\mathbb{P}}(\cdot)]$ generated by the control algorithm and the normalized control inputs $[U_1(\cdot), U_2(\cdot), U_3(\cdot), U_4(\cdot)]^{\text{T}}$ while considering the micro-controller as a black box, is detailed in the following. To map the impact of our control inputs, the ACSL flightstack is modified such that direct values of $[U_1(\cdot), U_2(\cdot), U_3(\cdot), U_4(\cdot)]^{\text{T}}$ may be applied and all other controller components negated. For testing values of $[U_1(\cdot), U_2(\cdot), U_3(\cdot)]^{\text{T}}$, we measure the resulting thrust via the experimental setup shown in Figure 4.4a. For testing the impact of $U_4(\cdot)$, we employ the experimental setup shown in Figure 4.4b.

To begin the mapping of control inputs with the overall thrust created by the vehicle, we follow the procedure laid out in Algorithm 4.2.1 such that we measure the resulting thrust

Algorithm 4.2.1 Thrust Mapping

for $U_1 = 0.05j, j \in \{1, 2, 3, \ldots 20\}$ **do** ▷ Net effect of U_1
  for $i \in \{1, \ldots, 4\}$ **do** ▷ Accounting for all 4 rotors
    $T_{\text{rotor},i}(U_1) \leftarrow$ Scale Reading
  end for
  $T_{\text{total}}(U_1) = \sum_{i=1}^{4} T_{\text{rotor},i}(U_1)$ ▷ Thrust of each rotor combined
end for

generated by each motor over a series of test values given by

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} j \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where $j \in \{0.05, 0.10, \ldots, 1\}$. The thrust measured for a single motor as a result of the applied value of $U_1(\cdot)$ is denoted $T_{\text{rotor},i}(U_1(\cdot)), i \in 1, \ldots, 4$. A summation of these thrusts is then taken as the total thrust that the vehicle can produce and denoted $T_{\text{total}}(U_1(\cdot))$.

Algorithm 4.2.2 $M_x^{\mathbb{P}}$ Mapping

for $U_2 = 0.05k, k \in \{-10, -9, \ldots, 10\}$ **do** ▷ Applying range of values for U_2
  for $U_1 = 0.1j, j \in \{4, 5, 6\}$ **do** ▷ Accounting for a range of U_1
    for $i \in \{1, \ldots, 4\}$ **do** ▷ Accounting for all 4 motors
      $T_{\text{rotor},ijk}(U_1, U_2) \leftarrow$ Scale Reading
      $T_{\text{rotor},ijk,\text{net}}(U_2) = T_{\text{rotor},ijk}(U_1, U_2) - T_{\text{rotor},ijk}(U_1, 0)$
      $M_{\text{rotor},ijk}(U_2) = \text{SIGN}(i) \cdot l_x \cdot T_{\text{rotor},ijk,\text{net}}(U_2)$
    end for
    $M_{\text{total},jk}(U_2) = \sum_{i=1}^{4} M_{\text{rotor},ijk}(U_2)$ ▷ Moment due to each rotor combined
  end for
  $M_{\text{total},k}(U_2) = \frac{1}{3} \sum_{j=1}^{3} M_{\text{total},jk}(U_2)$
end for

function SIGN(i) ▷ Accounts for the direction that thrust is applied about $x^{\mathbb{P}}$
  if $i == 2$ or $i == 3$ **then**
    Return 1 ▷ Thrust of rotor creates positive moment about $x^{\mathbb{P}}$
  else
    Return -1 ▷ Thrust of rotor creates negative moment about $x^{\mathbb{P}}$
  end if
end function

We then proceed to map the moment related to U_2 with Algorithm 4.2.2 We measure the

thrust generated by each motor over a series of test values given by

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} j \\ k \\ 0 \\ 0 \end{bmatrix}$$

where $j \in \{0.40, 0.50, 0.60\}$ and $k \in \{-0.5, -0.45, \ldots, 0.45, 0.5\}$. The range of j is selected as it reflects the intended throttle value at which the UAV hovers. We select a range of values for U_1 to verify that the moment created due to U_2 is largely unaffected by U_1. However, U_1 does impact the amount of thrust available to create $M_x^{\mathbb{P}}$. The range of k is selected such that it leaves power available for $M_y^{\mathbb{P}}$ and $M_z^{\mathbb{P}}$.

We conduct the experiment by applying $U_2 = 0.05k, k \in \{-10, -9, \ldots, 10\}$, fixed $U_1 = 0.1j, j \in \{4, 5, 6\}$, for each rotor and measuring the thrust output of the rotor, denoted by $T_{\text{rotor},ijk}(U_1, U_2)$. The thrust produced for a particular value of U_1 when U_2 is set to zero is denoted $T_{\text{rotor},ijk}(U_1, 0)$. We account for the net impact of U_2 on the measured thrust of a single rotor, denoted by $T_{\text{rotor},ijk,\text{net}}(U_2) \triangleq T_{\text{rotor},ijk}(U_1, 0) - T_{\text{rotor},ijk}(U_1, U_2)$. The contribution of a single rotor to $M_x^{\mathbb{P}}$ produced by the UAV for a particular combination of U_1 and U_2 is defined as $M_{\text{rotor},ijk}(U_2) \triangleq T_{\text{rotor},ijk,\text{net}}(U_2) l_x$ and accounting for the direction to which the rotor's thrust impacts $M_x^{\mathbb{P}}$, denoted by the $\text{SIGN}(i)$ function. For each value U_1 and U_2, we perform a summation of each rotor's contribution to $M_x^{\mathbb{P}}$, denoted by $M_{\text{total},jk}(U_2)$. We then average the net moment about $M_{\text{total},jk}$ across $U_1 = 0.1j, j \in \{4, 5, 6\}$, such that we have a set of moments, denoted $M_{\text{total},k}(U_2)$, that correspond to the control inputs $U_2 = 0.05k, k \in \{-10, -9, \ldots, 10\}$.

The process of mapping U_3 is performed with Algorithm 4.2.3. We measure the thrust

Algorithm 4.2.3 $M_y^{\mathbb{P}}$ Mapping

for $U_3 = 0.05k, k \in \{-10, -9, \ldots, 10\}$ **do** ▷ Applying range of values for U_3
  for $U_1 = 0.1j, j \in \{4, 5, 6\}$ **do** ▷ Accounting for a range of U_1
    for $i \in \{1, \ldots, 4\}$ **do** ▷ Accounting for all 4 motors
      $T_{\text{rotor},ijk}(U_1, U_3) \leftarrow$ Scale Reading
      $T_{\text{rotor},ijk,\text{net}}(U_3) = T_{\text{rotor},ij}(U_1, U_3) - T_{\text{rotor},i}(U_1, 0)$
      $M_{\text{rotor},ijk}(U_3) = \text{SIGN}(i) \cdot l_y \cdot T_{\text{rotor},i,\text{net}}(U_3)$
    end for
    $M_{\text{total},jk}(U_3) = \sum_{i=1}^{4} M_{\text{rotor},ij}(U_3)$ ▷ Moment due to each rotor combined
  end for
  $M_{\text{total},k}(U_3) = \frac{1}{3} \sum_{n=1}^{3} M_{\text{total},j}(U_3)$
end for

function SIGN(i) ▷ Accounts for the direction that thrust is applied about $y^{\mathbb{P}}$
  if $i == 1$ or $i == 3$ **then**
    Return 1 ▷ Thrust of rotor creates positive moment about $y^{\mathbb{P}}$
  else
    Return -1 ▷ Thrust of rotor creates negative moment about $y^{\mathbb{P}}$
  end if
end function

generated by each motor over a series of test values given by

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} j \\ 0 \\ k \\ 0 \end{bmatrix}$$

where $j \in \{0.40, 0.50, 0.60\}$ and $k \in \{-0.5, -0.45, \ldots, 0.45, 0.5\}$. The range of j is selected as it reflects the intended throttle value at which the UAV hovers. We select a range of values for U_1 to verify that the moment created due to U_3 is largely unaffected by U_1. However, U_1 does impact the amount of thrust available to create $M_y^{\mathbb{P}}$. The range of k is selected such that it leaves power available for $M_x^{\mathbb{P}}$ and $M_z^{\mathbb{P}}$.

We conduct the experiment by applying $U_3 = 0.05k, k \in \{-10, -9, \ldots, 10\}$, fixed $U_1 =$

Algorithm 4.2.4 $M_z^{\mathbb{P}}$ Mapping

for $U_4 = 0.05k, k \in \{-8, -7, \ldots, 8\}$ **do** ▷ Applying range of values for U_4
  for $U_1 = 0.1j, j \in \{4, 5, 6\}$ **do** ▷ Accounting for a range of U_1
    for $i \in \{1, \ldots, 4\}$ **do** ▷ We assume 4 rotors
      $F_{\text{rotor},ijk}(U_1, U_4) \leftarrow$ Scale Reading
      $M_{\text{rotor},ijk}(U_1, U_4) = l_{\text{arm}} \cdot F_{\text{rotor},ijk}(U_1, U_4)$
      $M_{\text{rotor},ijk,\text{net}}(U_4) =$ SIGN$(i)(M_{\text{rotor,ijk}}(U_1, U_4) - M_{\text{rotor,ijk}}(U_1, 0))$
    end for
    $M_{\text{rotor},jk}(U_4) = \sum_{i=1}^{4} M_{\text{rotor},ijk,\text{net}}(U_4)$ ▷ Moment of each rotor combined
  end for
  $M_{\text{total},k}(U_4) = \frac{1}{3}\sum_{j=1}^{3} M_{\text{total,jk}}(U_4)$
end for

function SIGN(i) ▷ Accounts for the direction that thrust is applied about $z^{\mathbb{P}}$
  if $i == 1$ or $i == 2$ **then**
    Return 1 ▷ Thrust of rotor creates positive moment about $z^{\mathbb{P}}$
  else
    Return -1 ▷ Thrust of rotor creates negative moment about $z^{\mathbb{P}}$
  end if
end function

$0.1j, j \in \{4, 5, 6\}$, for each rotor and measuring the thrust output of the rotor, denoted by $T_{\text{rotor},ijk}(U_1, U_3)$. The thrust produced for a particular value of U_1 when U_3 is set to zero is denoted $T_{\text{rotor},ijk}(U_1, 0)$. We account for the net impact of U_3 on the measured thrust of a single rotor, denoted by $T_{\text{rotor},ijk,\text{net}}(U_3) \triangleq T_{\text{rotor},ijk}(U_1, 0) - T_{\text{rotor},ijk}(U_1, U_3)$. The contribution of a single rotor to $M_y^{\mathbb{P}}$ produced by the UAV for a particular combination of U_1 and U_3 is defined as $M_{\text{rotor},ijk}(U_3) \triangleq T_{\text{rotor},ijk,\text{net}}(U_3) l_y$ and accounting for the direction to which the rotor's thrust impacts $M_y^{\mathbb{P}}$, denoted by the SIGN(i) function. For each value U_1 and U_3, we perform a summation of each rotor's contribution to $M_y^{\mathbb{P}}$, denoted by $M_{\text{total},jk}(U_3)$. We then average the net moment about $M_{\text{total},jk}$ across $U_1 = 0.1j, j \in \{4, 5, 6\}$, such that we have a set of moments, denoted $M_{\text{total},k}(U_3)$, that correspond to the control inputs $U_3 = 0.05k, k \in \{-10, -9, \ldots, 10\}$.

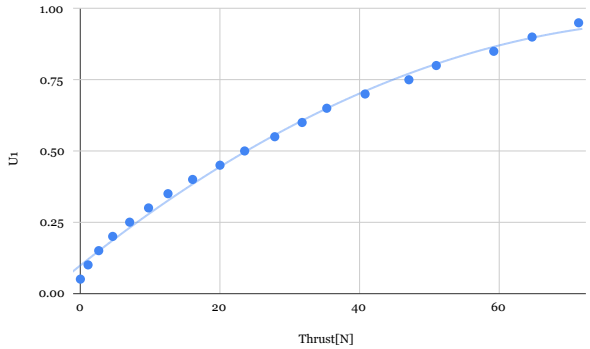
The procedure for mapping U_4 is shown in Algorithm 4.2.4. This experiment is conducted

with the setup shown in Figure 4.4b, and our sensor reading does not reflect the thrust generated, rather the reaction torque of the rotor due to air drag. We measure the moment generated by each motor over a series of test values given by

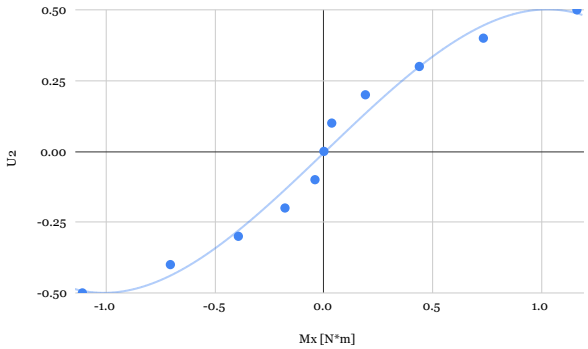
$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} j \\ 0 \\ 0 \\ k \end{bmatrix}$$

where $j \in \{0.40, 0.50, 0.60\}$ and $k \in \{-0.4, -0.35, \ldots, 0.35, 0.4\}$. The range of j is selected as it reflects the intended throttle value at which the UAV hovers. We select a range of values for U_1 to verify that the moment created due to U_4 is largely unaffected by U_1. However, U_1 does impact the amount of thrust available to create $M_z^{\mathbb{P}}$. The range of k is selected such that it leaves power available for $M_x^{\mathbb{P}}$ and $M_y^{\mathbb{P}}$.

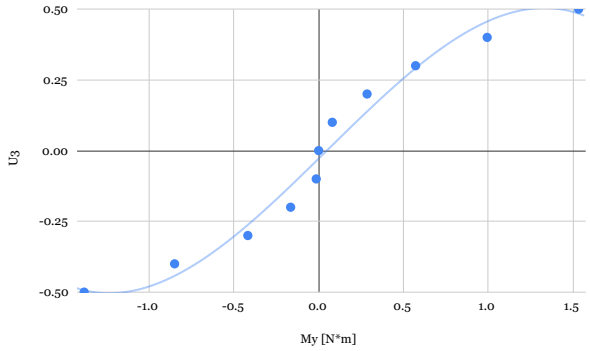
We conduct the experiment by applying $U_4 = 0.05k, j \in \{-8, -7, \ldots, 8\}$, fixed $U_1 = 0.1j, j \in \{4, 5, 6\}$, for each rotor, and measuring the resultant force, denoted by $F_{\text{rotor},ijk}(U_1, U_4)$. The resulting reaction torque of a single rotor is defined $M_{\text{rotor},ijk}(U_1, U_4) \triangleq l_{\text{arm}} F_{\text{rotor},ijk}(U_1, U_4)$. The moment produced for a particular value of U_1 when U_4 is set to zero is denoted $M_{\text{rotor,ijk}}(U_1, 0))$. We account for the net impact of of U_3 on the measured torque of a single rotor, denoted $M_{\text{total,jk}}(U_4) \triangleq M_{\text{rotor},ijk}(U_1, U_4) - M_{\text{rotor,ijk}}(U_1, 0))$ and account for the direction to which the rotor's torque impacts $M_z^{\mathbb{P}}$, denoted by the $\text{SIGN}(i)$ function. For each value U_1 and U_4, we perform a summation of the torques acting about $z^{\mathbb{P}}$, denoted by $M_{\text{total,jk}}(U_4)$. We then average the net moments of $M_{\text{total},jk}$ across $U_1 = 0.1j, j \in \{4, 5, 6\}$, such that we have a single set of moments, denoted by $M_{\text{total},k}(U_4)$, that correspond to the control inputs $U_4 = 0.05k, k \in \{-8, -7, \ldots, 8\}$.



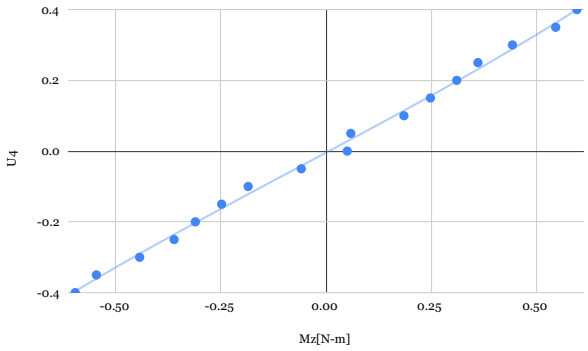
(a) Mapping of $T(t)$ to $U_1(t)$.



(b) Mapping of $M_x^{\mathbb{J}}(t)$ to $U_2(t)$.



(c) Mapping of $M_y^{\mathbb{J}}(t)$ to $U_3(t)$.



(d) Mapping of $M_x^{\mathbb{J}}(t)$ to $U_4(t)$.

Figure 4.5: The plotted results of the procedure detailed in Section 4.2.2

4.2.3 Results

The results of the experiment, following the process detailed in Section 4.2.2, are provided in this section. A second order polynomial interpolation of measured thrust and corresponding throttle input, U_1, as detailed by Algorithm 4.2.1 and shown in Figure 4.5a, yields

$$U_1(T) = 0.0982 + 0.0195T - 0.000111T^2, \qquad T \in [0, 71]. \tag{4.3}$$

A third order polynomial interpolation of $M_{\text{total},k}(U_2)$ and corresponding U_2 values, as detailed by Algorithm 4.2.2 and displayed in Figure 4.5b, yields

$$U_2(M_x^{\mathbb{P}}) = -0.00515 + 0.737M_x^{\mathbb{P}} + 0.00581(M_x^{\mathbb{P}})^2 - 0.237(M_x^{\mathbb{P}})^3, \qquad M_x^{\mathbb{P}} \in [-1.5, 1.5]. \tag{4.4}$$

A third order polynomial interpolation of $M_{\text{total},k}(U_3)$ and corresponding U_3 values, as detailed by Algorithm 4.2.3 and displayed in Figure 4.5c, yields

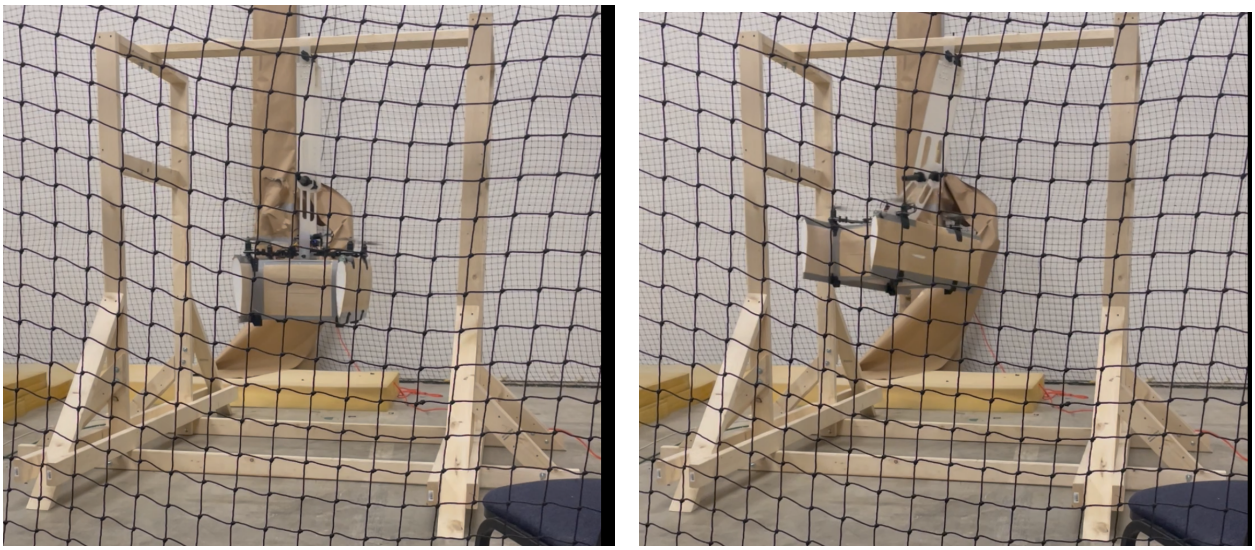
$$U_3(M_y^{\mathbb{P}}) = -0.0284 + 0.588M_y^{\mathbb{P}} + 0.0173(M_y^{\mathbb{P}})^2 - 0.119(M_y^{\mathbb{P}})^3, \qquad M_y^{\mathbb{P}} \in [-1.5, 1.5]. \tag{4.5}$$

A third order polynomial interpolation of $M_{\text{total},k}(U_4)$ and corresponding U_4 values, as detailed by Algorithm 4.2.4 and displayed in Figure 4.5d, yields

$$U_4(M_z^{\mathbb{P}}) = 6.7110^{-3} + 1.56M_z^{\mathbb{P}} - 0.0624(M_z^{\mathbb{P}})^2 - 0.435(M_z^{\mathbb{P}})^3, \qquad M_z^{\mathbb{P}} \in [-0.6, 0.6]. \tag{4.6}$$

4.2.4 Conclusion

In this section, we presented the motivation, setup, implementation, and results of an experiment designed to accurately control the QRBP discussed in this thesis. While initially intended for the QRBP, this process is applicable to any other quadcopter or X8 configuration coaxial UAV. Improvements to this experiment include automation such that the force of more complex combinations of control signals may be examined, changes to the physical layout such that multiple motors may be tested at once, and further refinements to eliminate disturbances of sensor readings. The characterization and mapping of $[T, M_x^{\mathbb{P}}, M_y^{\mathbb{P}}, M_z^{\mathbb{P}}]$ to $[U_1, U_2, U_3, U_4]^{\mathrm{T}}$ can be enhanced with an adaptive component in future work.



(a) (b)

Figure 4.6: Displayed here is a test of the QRBP's longitudinal control in which a small hover is performed. In this configuration, the polyethelene bracket provides torsion against the QRBP's lateral states of movement. Obstructing the view of the experiment is safety netting that drapes the VICON motion capture system used in these experiments.

4.3 Static Fixture Experimentation

As part of testing the control architecture presented in Chapter 3, we incorporate physical apparatus which allow us to examine the response of the controller in very specific scenarios, provide an added layer of safety and improve the consistency of a given flight test. This section details design considerations, development, and resulting impact of employing such experimental set-ups.

This section is structured as follows. Subsection 4.3.1 provides a brief description of the motivation and design considerations for a static structure for refining control algorithms. Subsection 4.3.2 details an experimental set-up that suspends a UAV in the air to allow for testing of either lateral or longitudinal states. Subsection 4.3.3 details the development of a static structure for use in testing the attitude control of the UAV. Subsection 4.3.4 presents the results of experiments performed on static structures. Subsection 4.3.5 presents

conclusions on the subject matter of this section.

4.3.1 A Brief Overview

When designing an experimental set up to streamline the process of refining the control architecture proposed in Chapter 3, we specifically consider which degrees of freedom we wish to allow the vehicle to move in while also restricting it from hitting things such as the VICON motion capture system, equipment, and operators. As a consideration for improving the repeatability of experiments and the efficiency with which they are performed, the following apparatus are designed such that the UAV starts and ends each flight at the same or very similar position. The following structures are built from materials that are easy to obtain and work with. For this reason, initial test stands are constructed from standard framing lumber, steel carriage bolts, polyethelene sheets, and decking screws. For more complex parts, a 3-D printer with Polyethylene terephthalate glycol (PETG) filament is used. Later experimental set-ups are constructed from welded steel and make use of commercially available machine parts for moving parts.

4.3.2 Isolated Controller Testing Stand

The first testing apparatus employed is built to evaluate the isolated response of the control algorithms presented in Sub-section 3.5.1 and Sub-section 3.5.2. This testing stand is constructed with the intention of suspending the QRBP in the air in such a way that only the states whose controller is being tested are free.

In one configuration, a bracket made from a bent sheet of polyethelene is used to constrain the motion of the QRBP, by adding torsion against movement of the lateral states, so that it can only move in the longitudinal plane. Shown in Figure 4.6, is an experiment in which

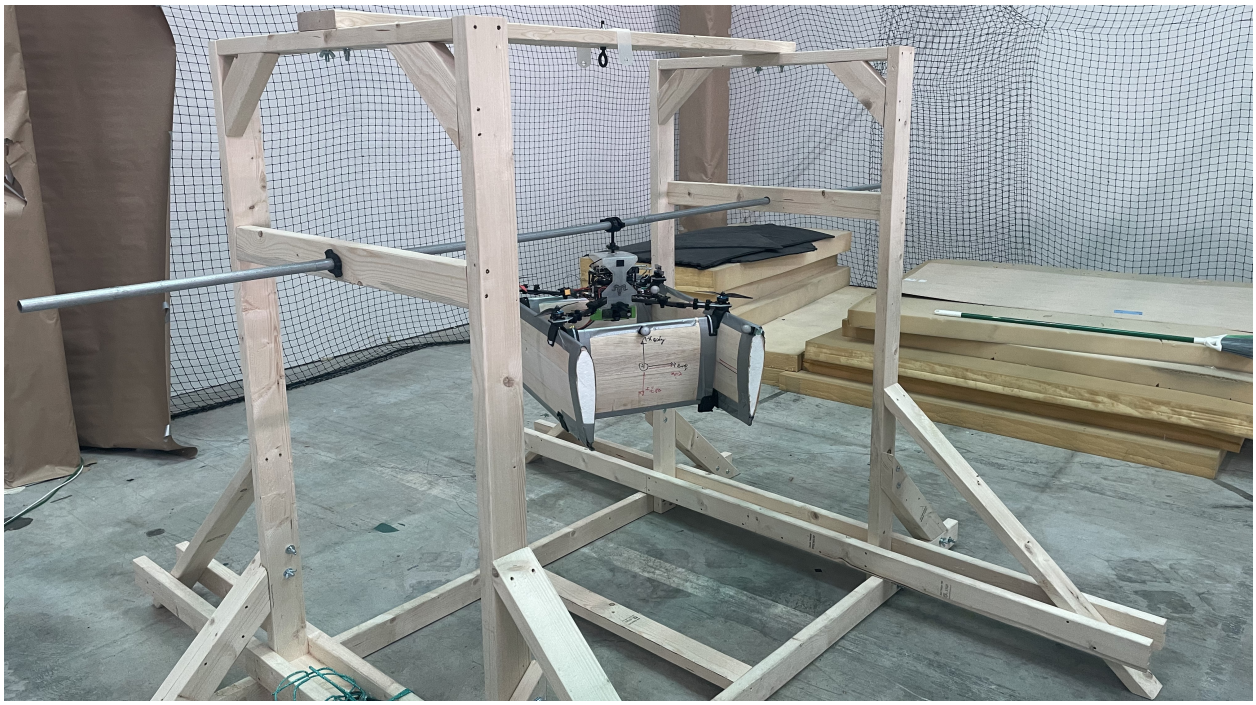
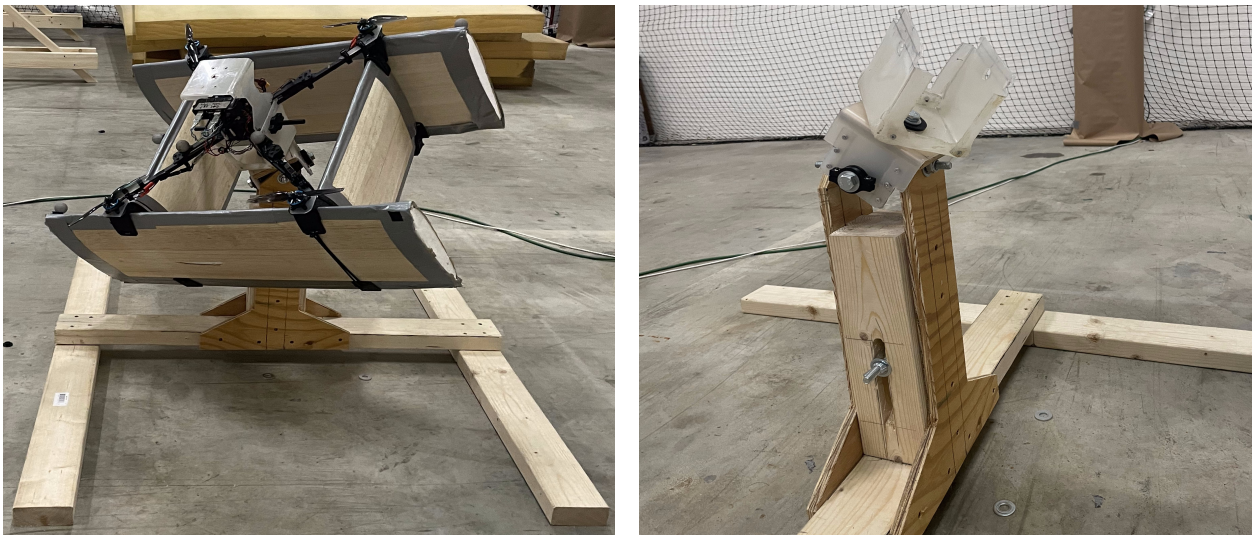


Figure 4.7: Displayed here is a configuration of the static structure that constrains all of the QRBP's motion but roll and yaw.

this configuration is used and as such, the vehicle is able to perform a rudimentary hover maneuver without need for lateral control.

In another configuration of this test stand, the longitudinal states of the QRBP are constrained to allow for testing of the lateral controller. This is done via a steel *eye-hook* mounted to the test stand with metal rod. This configuration, as shown in Figure 4.7, allows the QRBP's movement to be constrained solely to roll and yaw.

The tests performed on this structure are used to verify basic efficacy of the two control algorithms presented in Sub-Sections 3.5.1 and 3.5.2 respectively, in isolation. This structure does not replace actual flight and faces several issues. Namely, by hanging a UAV from a structure, the actual effort required on the part of the controller was far less than it would be in actual flight, and by isolating the control algorithms comprising Chapter 3, it could not be definitively proven that the two comprising algorithms worked seamlessly



(a) (b)

Figure 4.8: Displayed here is the initial Inverted-Pendulum Stand shown with (left) and without (right) the QRBP mounted to it.

together. Additionally, the test stand could obstruct the motion capture system, thereby reducing the quality of the vehicle's state data. Essentially, the experimental set-up presented in this Sub-Section does not provide a test-bed with which to effectively tune the gain values dictating the proposed controller's response, but it does allow for verification that the proposed controller's response does attempt to drive the UAV toward a commanded trajectory, thereby determining the controller to be ready for the next stage of refinement.

4.3.3 Inverted Pendulum Test Stand

To continue refining the proposed control algorthim of the QRBP and improve upon the static test fixture presented in 4.3.2, another static structure is employed, the *Inverted Pendulum Stand*, which fixes the position of the QRBP and provides a UAV with all three degrees of rotational freedom. The intention of this static structure is to verify that the comprising parts of the control algorithm presented in Chapter 3 are cohesive, thereby allowing us

(a) (b)

Figure 4.9: The redesigned Inverted-Pendulum Stand shown with (left) and without (right) the QRBP mounted to it.

to determine preliminary gain values with which to dictate the controller's response, while negating the need for position control. We then employ this structure to ensure capability of the proposed control algorithm to track user-defined attitude trajectories.

The initial Inverted Pendulum Test Stand, shown in Figure 4.8 is built from framing lumber, steel bolts, polyethelene, and 3-D printed parts. Due to part availability, 3-D printed bushings are used with limited effectiveness as they lack tight tolerances and ruggedness. From testing, it is apparent that the gains needed for a vehicle to balance atop this structure are generally 4 times higher than the gains needed for actual flight.

To address these shortcomings and to take advantage of different machining capabilities, a new Inverted Pendulum Test Stand, shown in Figure 4.9, is built from welded steel tubing. To replace the previously 3-D printed parts, a tie-rod end is utilized. From testing, this

greatly reduces friction and gains found in this system are able to be directly used in actual flight tests with only minor adjustments.

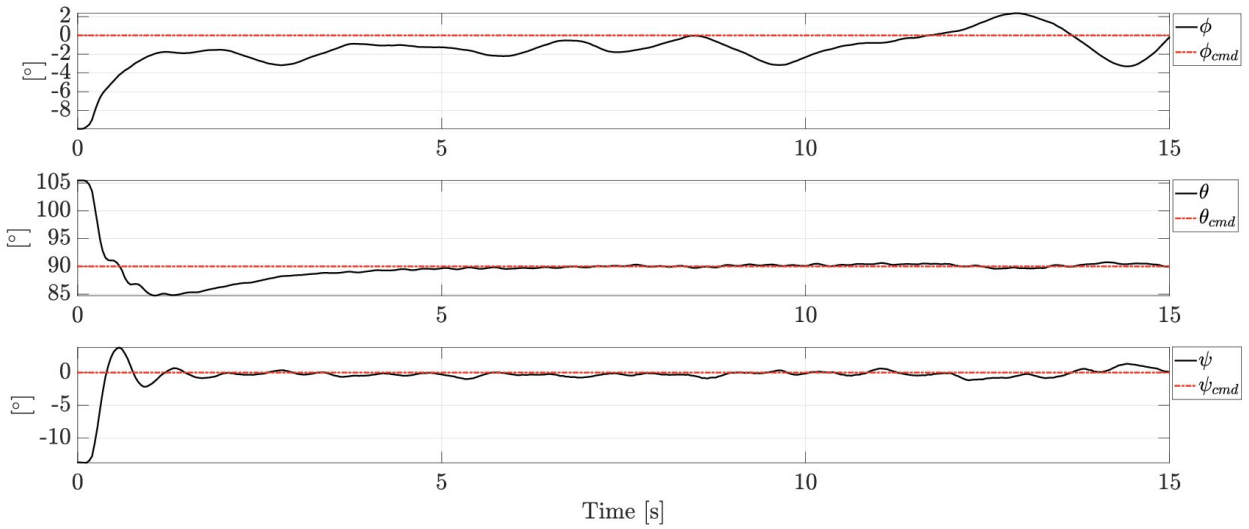


Figure 4.10: An experiment of the QRBP's attitude control while mounted to the Inverted-Pendulum Stand.

4.3.4 Results

This section presents the results of experiments performed on the static fixture presented in Sub-Section 4.3.3.

Tests of the attitude control of the QRBP require disabling the position control elements of the longitudinal and lateral controller. Whereas previously the commanded yaw and pitch of the aircraft are calculated as a function of the forces used for the QRBP to track a user-defined trajectory, we omit those forces from our commanded attitude values. We directly define the command pitch and yaw of the inner-loop of the proposed longitudinal and lateral controllers. The commanded roll trajectory of the aircraft remains user-defined.

Having determined that the proposed controller could track a commanded trajectory in the longitudinal and lateral regimes, the controller's gains are more closely tuned with the

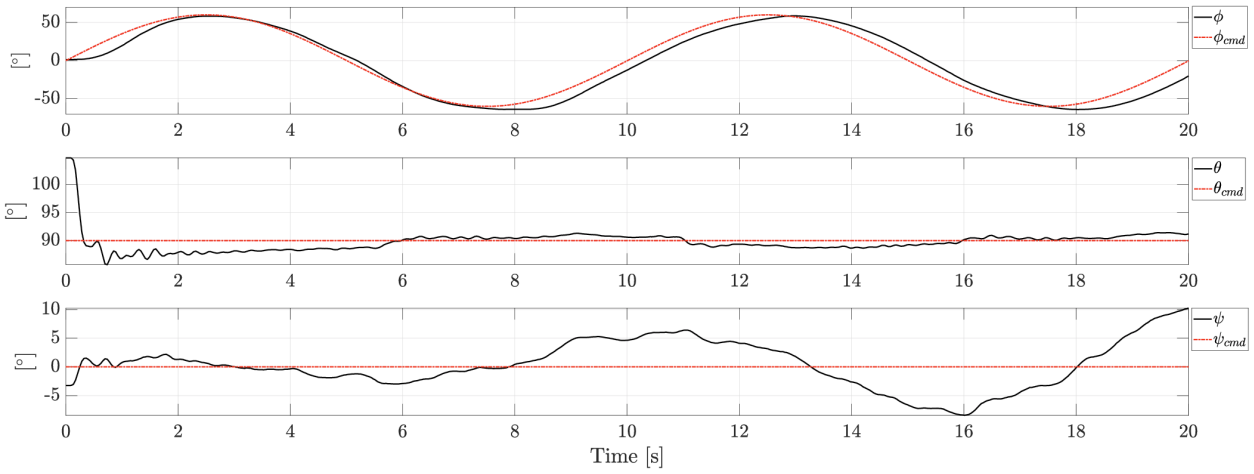


Figure 4.11: A test of the QRBP's attitude control, given a sinusoidal ϕ command, while mounted to the Inverted-Pendulum Stand.

QRBP affixed to the Inverted-Pendulum Stand. For the initial test, we tune of the gain values that impact attitude control by first setting,

$$\begin{bmatrix} \phi_{\text{cmd}} \\ \theta_{\text{cmd}} \\ \psi_{\text{cmd}} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{\pi}{2} \\ 0 \end{bmatrix}, \tag{4.7}$$

such that the vehicle must balance itself. Shown in Figure 4.10 are the results of this experiment.

Shown in Figure 4.11 are the results of a test in which we set

$$\begin{bmatrix} \phi_{\text{cmd}} \\ \theta_{\text{cmd}} \\ \psi_{\text{cmd}} \end{bmatrix} = \begin{bmatrix} A\sin(B \cdot t) \\ \frac{\pi}{2} \\ 0 \end{bmatrix}, \tag{4.8}$$

where $A = 1.0472$ and $B = 0.6283$ resulting in a sine wave with an amplitude of 60° and a period of 10s. As $\theta_{\text{cmd}} = \frac{\pi}{2}$ and $\psi_{\text{cmd}} = 0$, the QRBP in this experiment must balance itself

vertically while rolling about $x^{\mathbb{J}}$.

4.3.5 Conclusion

The experimental set-ups presented in this section are used to refine the control algorithm presented in Chapter 3. The ability to isolate the lateral and longitudinal states of the QRBP is invaluable for debugging the implementations of those controllers. Furthermore, refinement of the vehicle's attitude control before attempting hover flight saves a tremendous amount of potential crashing. Results of the testing performed on the experimental setup presented in 4.3.3 demonstrate the capability of the proposed controller in tracking a commanded attitude and provide a jumping off point. Future work to improve the functionality of a fixed jig that isolates degrees of freedom of a UAV would be well served with the incorporation of t-slot aluminum extrusion and pillow-block bearings to reduce complexity of production, offer a far more adjustable geometry and produce smoother movements.

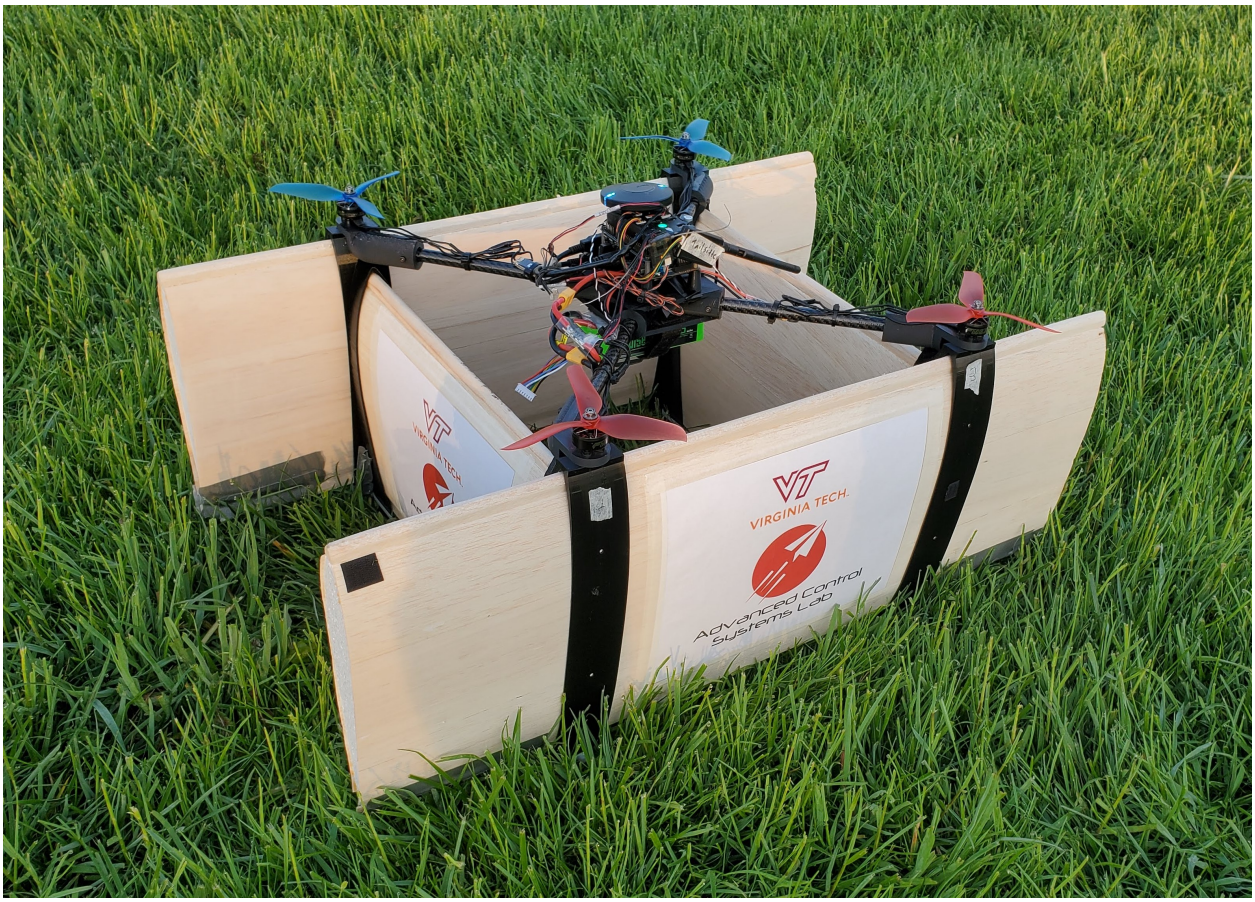


Figure 4.12: The previous configuration of the QRBP, originally presented in [4].

4.4 Vehicle Design

This section details improvements to the QRBP platform previously developed at the ACSL [4], shown in Figure 4.12, and is detailed as follows. Sub-section 4.4.1 presents an overview of the design requirements of the QRBP. Sub-section 4.4.2 presents the changes made to the vehicle's design along with justifications for said changes. In Sub-section 4.4.3 we draw conclusions about the changes made and explore possibilities for future work.

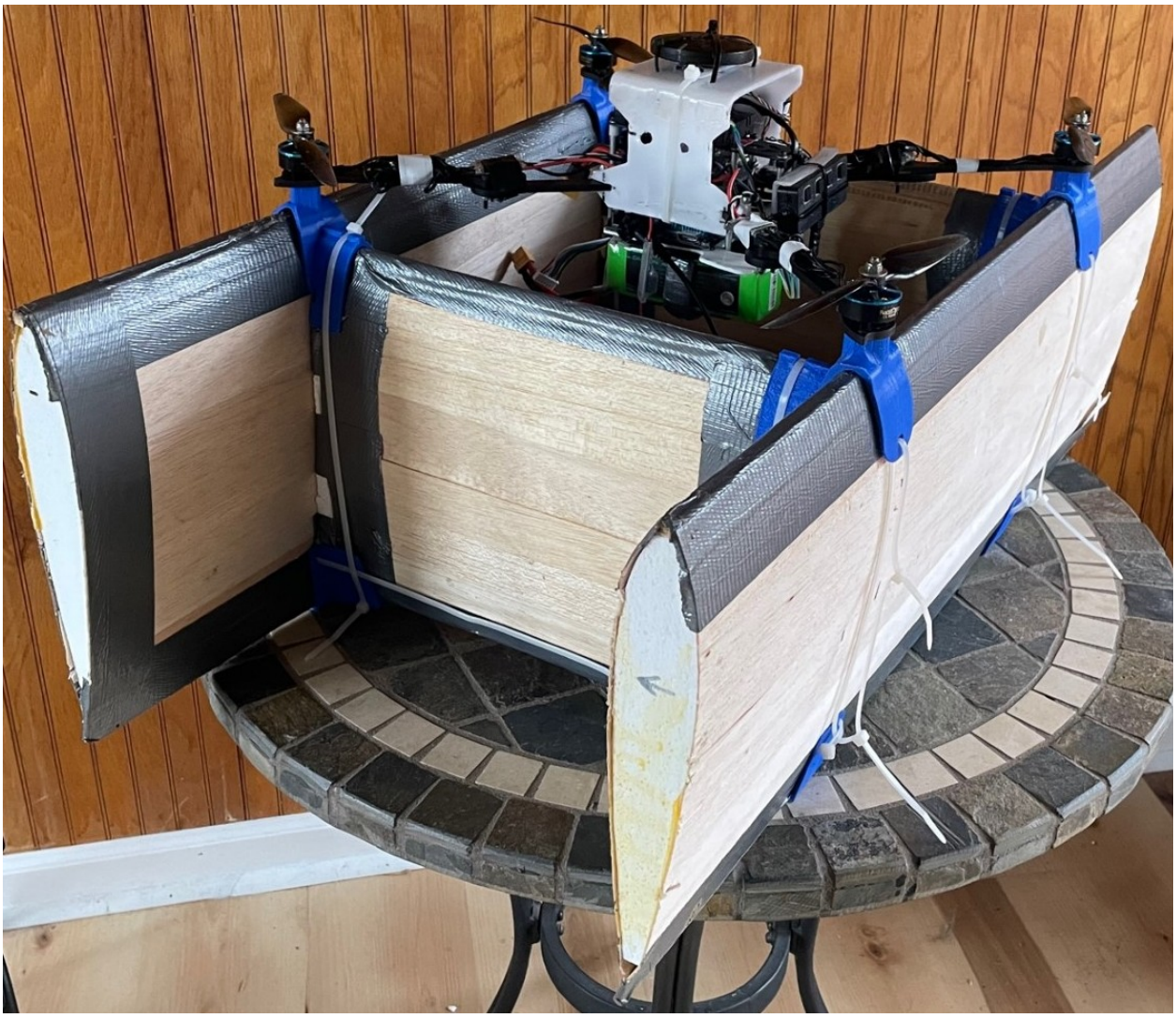


Figure 4.13: The current configuration of the QRBP.

4.4.1 Brief Overview

The theoretical QRBP is designed very simply such that it can operate in two distinct modes, quadcopter and biplane, with the only actuation being performed by its 4 rotors. Symmetric airfoils are chosen so that lift is not generated while the vehicle is in quadcopter mode. Our particular implementation of the QRBP is designed with the consideration of making flight testing as efficient as possible. This is accomplished by minimizing manufac-

turing time and relying more on COTS parts. Additionally, we incorporate intentional weak points such that we can anticipate and plan accordingly for inevitable breaks.

4.4.2 Physical Design

The version of the QRBP shown in Figure 4.12 weighs approximately 3.2kg, with a 0.75m wingspan, 0.5m of spacing between the wings, a chord-length of 0.29m, a platform area of $0.41m^2$, and a symmetric airfoil closely resembling the NACA 0012 airfoil. This configuration provided satisfactory flight performance, however it does have drawbacks, namely manufacturing time and available control bandwidth. It takes approximately 160 hours to 3-D print the parts required by this design. Additionally, with the weight of the vehicle, available thrust and the large moment necessary to conduct and maintain flight transition, it is advantageous to seek to reduce the mass of the vehicle and increase the thrust that it can produce.

To address these issues, the vehicle, as shown in Figure 4.13 is redesigned in such a way to maintain its aforementioned aerodynamic properties while reducing weight, increasing thrust and decreasing manufacturing time. The redesigned QRBP weighs approximately 2.4kg, with a 0.75m wingspan, Xm of spacing between the wings, a chord-length of 0.29m, a platform area of $0.41m^2$, and a symmetric airfoil closely resembling the NACA 0012 airfoil. For thrust, we use *Racerstar AirA 2508* 1200 KV motors paired with propellers of 7" diameter and 4.5" pitch. This combination provides us with a nearly 3:1 power-to-weight ratio as each rotor can generate up to 1.8kg of thrust. To the extent possible, 3-D printed parts are replaced with COTS parts. Parts that interface with the wings remain 3-D printed due to the complexity of their shape, however we optimize their design by further reducing mass. We do this by relying more heavily on the wings for structural integrity. This change reduces the required

3-D printing time from 160 hours to about 60 hours.

We incorporate intentional weak points in the design of the QRBP, so that in the event of a crash, more expensive and valuable components are spared. The motor mounts employed have an intentionally narrow neck to absorb the impact of of landing after flipping over. We use bent polypropylene to mount our GPS module to and shield the more valuable and fragile flight computer and micro-controller during flight. Should the vehicle pitch too far during transition to biplane mode and land on its top, the electronics shield absorbs this impact, likely breaking and thereby saving the more valuable components.

4.4.3 Conclusion

The design changes presented in this section result in a QRBP that is substantially lighter with greater control authority and faster to manufacture. The implication of these changes is the ability to pursue more aggressive trajectories with the QRBP and when the inevitable mishap occurs, it can be repaired and returned to testing much more quickly. Future improvements to the QRBP should include more robust wing construction and a stiffened frame. Larger and more powerful motors could be considered for even more ambitious trajectories and if payload delivery is part of mission requirements.

4.5 Conclusion

In this Chapter we present the considerations taken in the development of the control algorithm presented in Chapter 4. The experiment performed to characterize the QRBP's rotors allows us to more precisely actuate forces and moments during flight testing. The experimental set-ups we employ, allow us to test and analyze elements of the QRBP in a

more efficient manner while reducing risk of damage to the vehicle. To further increase the capabilities of the QRBP, the presented redesign allows for more ambitious flight tests and faster turnaround time, if and when accidents occur. Effectively, we have established tools and procedures for introducing new physical components and additions to the proposed control algorithm.

Chapter 5

Preliminary Flight Tests

In this Chapter we present the initial results of in-flight testing of the control algorithm presented in Chapter 3. Having conducted the thrust-mapping experiments presented in Section 4.2, initial tuning via the experiments presented in Section 4.3, and incorporating the design changes of Section 4.4, we now conduct tests of the proposed controller with outdoor flight tests.

This chapter is structured as follows. Section 5.1 provides an overview of the experimental process employed for flight testing. Section 5.2 provides the results of our testing. Finally, Section 5.3 draws conclusions on the material discussed in this chapter and provides observations in regards to future testing.

5.1 Brief Overview

Generally speaking, the strategy of tuning the proposed controller is to tune as few gains at a time as possible and verify that the QRBP's physical characteristics are capable of performing increasingly demanding maneuvers while considering where we would like the QRBP's available control authority to be assigned. Considering our desire for the QRBP to take advantage of the lift generated by its wings, we want it pitch aggressively so that it minimizes the time at which it is operating at a stall angle-of-attack. With these considerations, we tune the user-defined gain values connected to the x and θ states such that

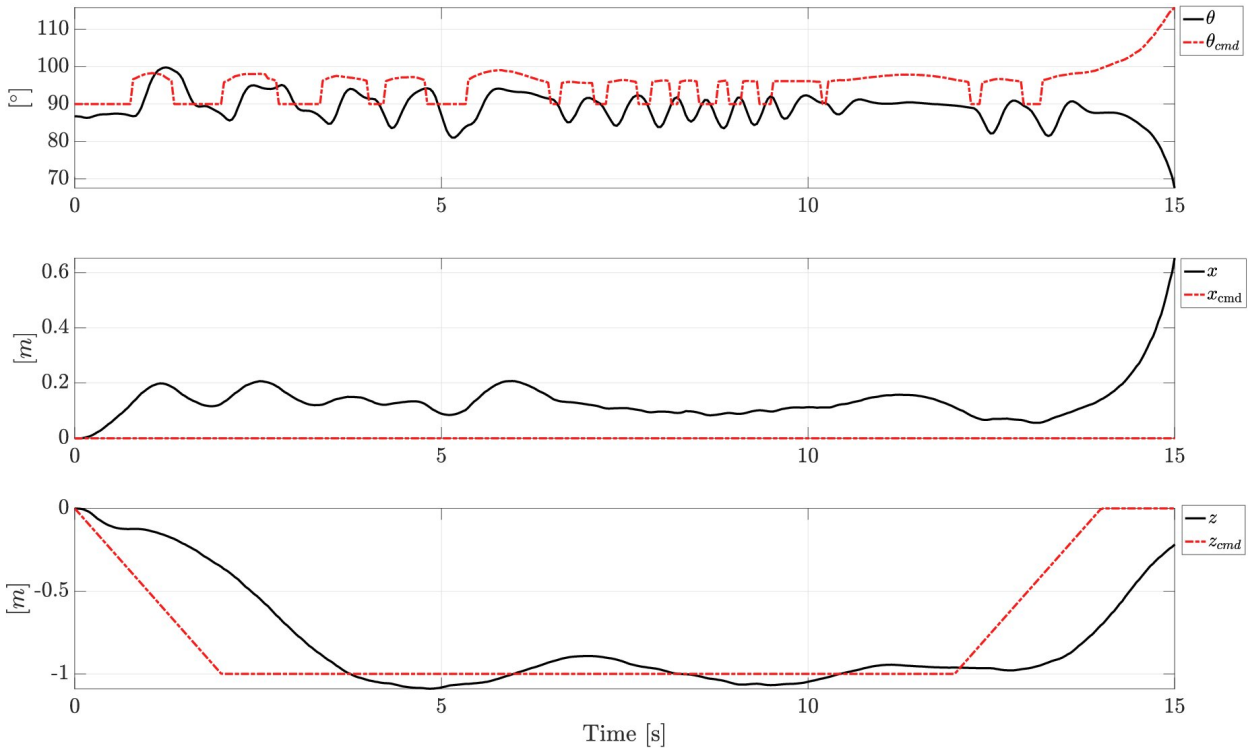


Figure 5.1: The QRBP's longitudinal states during hover flight. The actual measured state(black) and the commanded trajectory(red) are displayed. The states are sorted by the order in which they are output from the proposed controller. In this trajectory, we command the vehicle to ascend at a rate of $0.5\frac{\text{m}}{\text{s}}$ to a height of 1m, hold position for 10s, and then descend at a rate of $0.5\frac{\text{m}}{\text{s}}$.

they are more aggressive than the gains dictating the response that tracks y and ψ. The reason for this choice is to free up available control bandwidth for more demanding pitch maneuvers. We begin the tuning process by giving the baseline component of the proposed controller full control bandwidth and a simple hover trajectory. Upon identifying gain values that enables the QRBP to follow a commanded trajectory, we appropriate larger swaths of control bandwidth to the adaptive component of the proposed controller and give the vehicle increasingly demanding trajectories.

Tests are conducted in an open field, selected due to their lack of obstacles and visibility of GPS satellites. During the initial controller testing in Section 4.3, we use the ACSL's

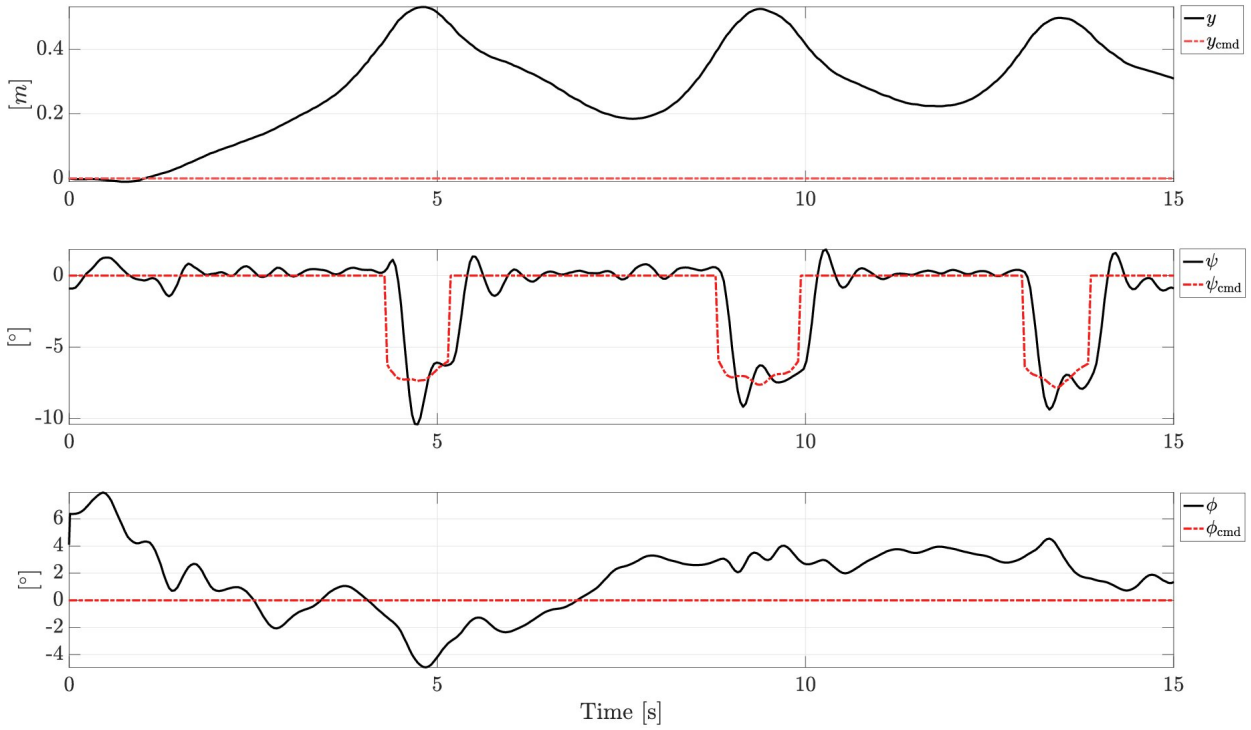


Figure 5.2: The QRBP's lateral states during hover flight. The actual measured state(black) and the commanded trajectory(red) are displayed. The states are sorted by the order in which they are output from the proposed controller. During this flight the commanded trajectory of the outer-loop lateral states is for y and ϕ to remain at zero.

VICON Motion Capture system for position data. In order to push the QRBP to perform more ambitious maneuvers, we need an area far larger than that covered by the VICON system, so we operate the QRBP in the outdoors and use an RTK GPS set-up for position data. As we are replacing a near-perfect source of position data for a less consistent solution, we select a testing area by the number of visible satellites to our GPS receiver. The area selected typically has 12-15 satellites visible.

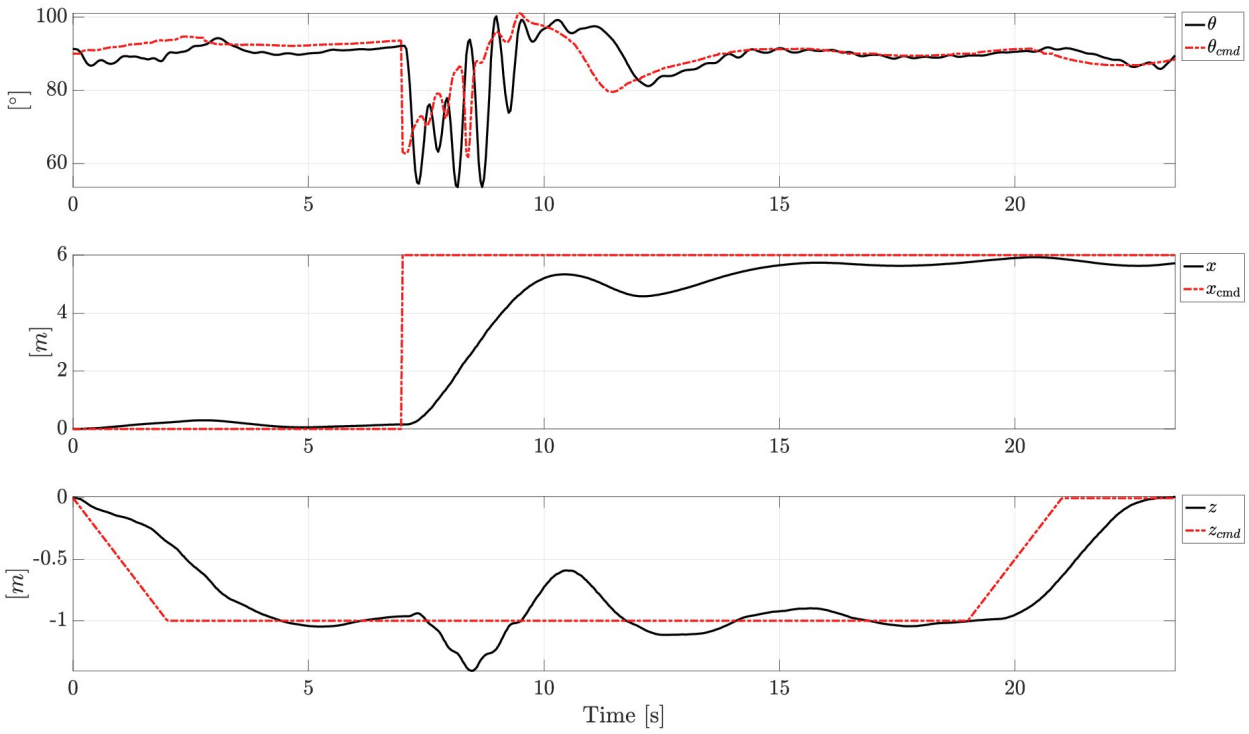


Figure 5.3: The QRBP's longitudinal states during forward flight. The actual measured state(black) and the commanded trajectory(red) are displayed. The states are sorted by the order in which they are output from the proposed controller. In this trajectory, we command the vehicle to ascend at $0.5\frac{\text{m}}{\text{s}}$ to a height of 1m, hold position for 5s, we then command a step-input trajectory to x_{cmd} of 6m, hold position for 12s, and then descend at a rate of $0.5\frac{\text{m}}{\text{s}}$.

5.2 Flight Tests

We begin flight tests by commanding the QRBP to hover 1m off the ground. Shown in Figure 5.1 are the states of the QRBP controlled by the longitudinal controller. Shown in Figure 5.2 are the states of the QRBP controlled by the lateral controller. In this experiment, we have the adaptive component of the proposed controller fully enabled. The user-defined parameters as used in this experiment are given in Appendix A.1.

The vehicle is able to track the commanded trajectory, allowing it to ascend, hover and descend in a controlled manner. In the last 1s, there is a slight loss of control of the states of

x and θ. From observation, this is due largely to ground effect which becomes particularly pronounced when attempting to slowly approach the ground.

In the next set of flight tests, we give the QRBP a commanded position that requires it to pitch significantly in order to track x_{cmd}. Shown in Figure 5.3 are the states of the QRBP controlled by the longitudinal controller. Shown in Figure 5.4 are the states of the QRBP controlled by the lateral controller.

In this flight, we push the QRBP towards a significantly more ambitious trajectory in which we give a step-input of 6m to x_{cmd}. In doing so, we push the QRBP to pitch 30°, which nears entering biplane mode. It is apparent that when the significant horizontal maneuver is performed, there are relatively large magnitude disturbances in nearly all other controlled states. This presents an impediment to more ambitious user-defined trajectories as the QRBP clearly forfeits a significant portion of its control authority in order to perform the pitch maneuver.

5.3 Conclusion

In this chapter we presented the preliminary flight tests to validate the controller presented in Chapter 3. We begin by presenting a brief overview of our gains tuning strategy and considerations of our testing environment. We present a flight test in which the QRBP, with all elements adaptive of the proposed controller enabled, successfully hovers. In the second presented flight test, we operate the QRBP with certain adaptive elements disabled, while pushing the vehicle to perform an aggressive maneuver.

From testing, we can conclude that the proposed controller, with all elements enabled, can produce a cohesive control-response that enables the vehicle to stabilize itself in mid-air.

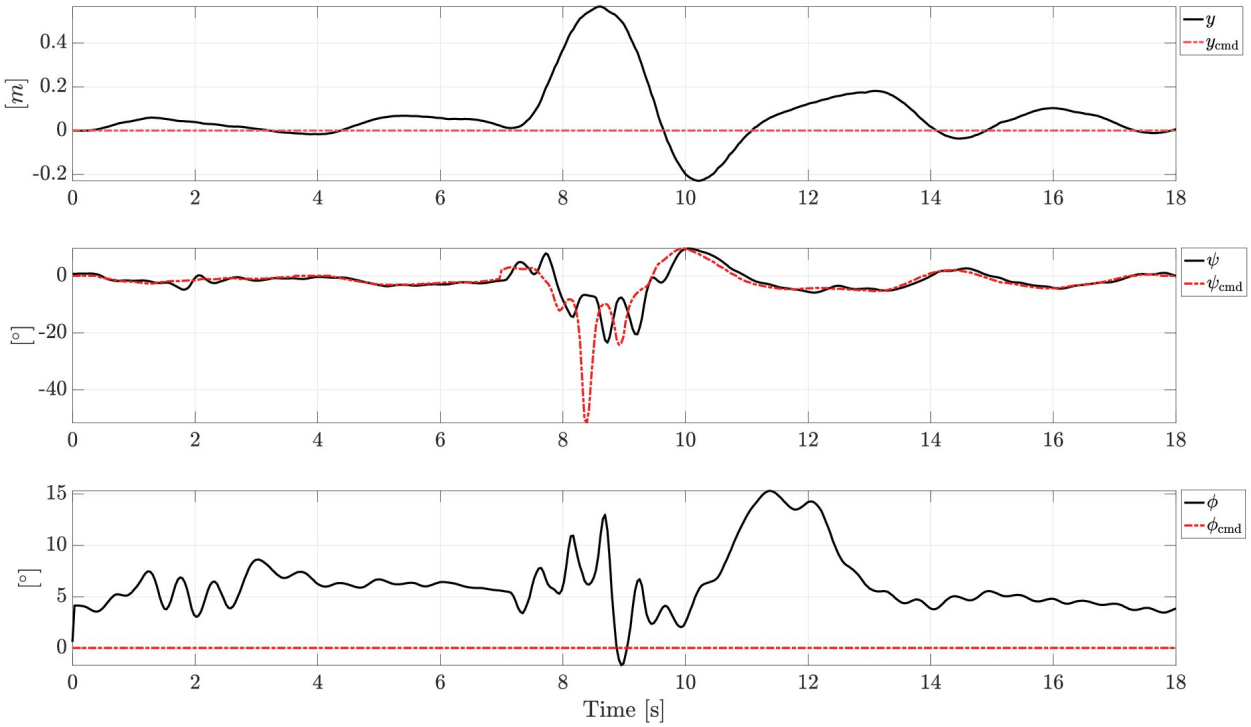


Figure 5.4: The QRBP's lateral states during forward flight. The actual measured state(black) and the commanded trajectory(red) are displayed. The states are sorted by the order in which they are output from the proposed controller. During this flight the commanded trajectory of the outer-loop lateral states is for y and ϕ to remain at zero.

We can also determine from this test that our throttle mapping is sufficiently accurate, and that the QRBP, as it currently exists, is capable of gentle maneuvering. The difficulties of maintaining control through an aggressive trajectory suggests of course that further tuning is required. However, deeper issues may be present and different trajectories should be considered in addition to further improvements to the QRBP's physical design.

Chapter 6

Conclusion

6.1 Summary of Results

This thesis presented key results toward the ambitious goal of creating a unified guidance and control system for autonomous VTOL UAVs operating in contested environment. Specifically, this thesis presented an optimization-based guidance system for UAVs able to map unknown, contested environments, where the potential threats' location and capabilities are unknown. Special emphasis has been put on the problem of characterizing by means of extensive numerical simulations of the speed of each of the components of the guidance system as well as the performance of the overall system. This thesis also presented a control system for QRBPs that is based on a switched-system formulation of MRAC, showed the results of numerical simulations, and outlined preliminary experimental results collected through outdoor tests. To perform these tests, several indoor tests to characterize the performance of the propulsion system as well as the UAV's inner loop dynamics have been performed and documented in this thesis.

In detail, the contributions of this thesis have been as follows. In Chapter 2, we began by describing an implementation of the Octree Algorithm as used to partition a space in which our autonomous quadcopter is exploring. Extensive numerical simulations highlighted the effect of user-defined parameters on the speed of the algorithm as well as in the UAV's expected behavior, and allowed us casting recommendations for the user. Having characterized

the way the map is partitioned, we use this information to select goal points, which must be reached by the UAV to complete its mapping task. Fixed a goal point, the guidance system outlines a path from the UAV's current position to said goal point. Two heuristics-based path planners have been considered, namely A* and LPA*. In both algorithms, the heuristic function and the cost-to-come function were biased by means of user-defined parameters to allow for more or less tactical trajectories, that is, for trajectories more or less prone to drawing the opponent's attention on the UAV. To navigagte along this path, we then derive the equations of motion of a quadcopter within the MPC framework, demonstrating the need for a constraint generation algorithm to create separations between areas in which it is safe for a UAV to traverse and where there are obstacles. We compared four state-of-the-art algorithms to identify obstacle-free regions surrounding the UAV, namely the SFC, IRIS, MDCA and Bubble-bath algorithms. In this thesis, we found that for consistency of volumes generated, minimal amounts of unbounded sets and speed, MDCA is the preferable choice. We also find that in applications in which a user seeks to maximize the volume generated by their constraint generator, IRIS should be considered. We then presented results of numerical simulations in which the entire guidance system is tested and we confirm that when given a reckless user-defined parameter set, the simulated UAV explores the given environment faster and less consistently, whereas the simulation in which the simulated UAV is given a more tactical and greedy user-defined parameter set, resulted in a slower and more consistent approach to exploring the environment.

In Chapter 3, we provided the theory of MRAC as it allows for control of a nonlinear system while adapting to external disturbances and modeling errors. We then provided an augmented theory of MRAC with output tracking for use in the longitudinal component of the proposed control algorithm.Successively, we provided the theory of MRAC for dynamically switched systems, such that it can capture the distinct operation modes of the QRBP

for the lateral component of our proposed controller. Finally, we combined these theoretical results to create a control system for a QRBP UAV.

In Chapter 4, we present solutions to challenges that arose during validation of the control algorithm presented in Chapter 3. We begin by presenting an experiment to characterize the rotors used in our design of the QRBP with readily available parts and the micro-controller used in actual flight tests. Due to the complexity of the proposed controller, we then present the static experimental test stands used to isolate and refine the control of various degrees of freedom. As testing progresses and a testing strategy evolves, we determined what physical characteristics are necessary to refine in order to improve flight performance.

In Chapter 5, we presented the results of preliminary flight tests to validate the proposed controller presented in Chapter 3. The first set of flight tests shows a hover flight in which all adaptive components are enabled which verifies that the proposed control algorithm does work. The second set of flight tests shows a forward flight in which several adaptive components are disabled and the QRBP relies primarily on its baseline controllers. In this second set of tests, we showed a sudden and large commanded pitch maneuver which the QRBP is able to perform, however, in doing so it loses a good deal of control authority in other states, thereby limiting this configuration of the QRBP from more aggressive trajectories until that issue is resolved.

6.2 Recommendations for Future Work

By the numerical simulations presented in the end of Chapter 2, the proposed guidance system is validated and therefore, it should be employed in actual flight tests. Furthermore, the trajectory planner could be augmented to include the additional forces and moments that comprise the difference between a regular quadcopter and the QRBP. In addition, object

detection could be added to provide more detailed information about what is found during exploration. This feature could enable a user to send a quadcopter into an environment completely autonomously and upon return, provide a list of items that are potentially dangerous or of note.

The next step in improving the control algorithm presented in Chapter 3 is to implement a longitudinal control algorithm that makes use of MRAC with output tracking and prescribed performance as detailed in [20]. In the long term, more accurate modeling of the QRBP would allow for more aggressive maneuvers as it becomes more predictable as to where power needs to be delivered. This is particularly relevant for applications in which the QRBP is used to deliver a slung payload. Similarly, a trajectory planner integrated with the control algorithm would allow for optimized trajectories to be generated that particularly suit the control architecture.

The next steps in improving the rotor characterization presented in Section 4.2 should be to employ steady state load cells and automate data collection. The 3D printed bushings originally used for moving parts in the thrust stand wore out quickly and bound up often. Logically, replacing them with smoother operating bearings would decrease the impact of friction but removing moving parts and instead using a solid state load cell would completely negate the issue. Careful geometry of the load cells would also allow for thrust and reaction torque to be measured simultaneously. Adding an automated measurement aspect to this experiment would allow not only for more accurate data and time average data to be collected, but more complicated thrust commands and a feedback mechanism. Further along in the development of this work, a series of rotors should be able to be tested at once, thereby allowing for the testing process to be done in batches. Even further refinements include designing the mounting pattern of the load cells in such a way that a UAV may be mounted to it. With the various load cells providing thrust and torque data, that data

could be combined into what is essentially a very large combination of an accelerometer and gyroscope. Having emulated this sensor, a proposed control algorithm could be tested on a UAV without the chance of it crashing.

The static experimental set-ups presented in Section 4.3 can generally be improved by either making the dynamics smoother and more isolating, or by integrating these style of set-ups with an automated thrust and moment measuring stand. Regardless, there is an enormous value in being able to test just a small part of a control algorithm at a time and the main issue with the presented experimental set-ups is that they do not perfectly align with the dynamics of real flight or the proposed control algorithm. By developing a stand that physically constrains a UAV while simultaneously measuring the output forces and moments, and providing state data, would alleviate those issues.

The material presented in Section 4.4 will progress in a predictable manner as flight tests continue. The problem with the QRBP configuration is that one rotor ends up doing most of the work, which can work when the margins of the vehicle's thrust-to-weight are high enough. The design of the vehicle will continue to shed weight while increasing power. Additionally, it is an inevitability with the QRBP and the manner in which we want it to fly that on occasion, it will flip over in testing. For this reason, future iterations may have the wings extended out further along the vertical inertial axis and have the propellers closer to the center of mass. This would protect the more fragile rotors and leave the brunt of the damage to the leading of the wings which can be more easily repaired.

The flight results presented in Chapter 5 indicate that the proposed control system works but has room for improvement before the QRBP will be able to fly in biplane mode for extended periods of time. Future improvements include pushing the QRBP to perform more ambitious maneuvers, such as maintaining a pitched angle for longer periods of time, and tuning the various gains accordingly. If the vehicle in its present configuration demonstrates

a consistent saturation of the control effort available by one of the rotors, it is a sign that the vehicle needs to be lightened and possibly requires larger rotors added. The bandwidth allotted to the baseline control should also be decreased and the adaptive gains increased as permitted by flight performance.

Bibliography

[1] Andrea L'Afflitto, Gokhan Inalhan, and Hyo-Sang Shin, editors. *Control of Autonomous Aerial Vehicles: Advances in Autopilot Design for Civilian UAVs*. Advances in Industrial Control. Springer Cham, 1 edition, 2023.

[2] Sikang Liu, Michael Watterson, Kartik Mohta, Ke Sun, Subhrajit Bhattacharya, Camillo J. Taylor, and Vijay Kumar. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments. *IEEE Robotics and Automation Letters*, 2(3):1688–1695, 2017.

[3] Robin Deits and Russ Tedrake. Computing large convex regions of obstacle-free space through semidefinite programming. In *Workshop on the Algorithmic Foundations of Robotics*, 2014.

[4] Grant I. Carter. Adaptive control of the transition from vertical to horizontal flight regime of a quad-tailsitter uav. Master's thesis, Virginia Polytechnic Institute and State University, 2021.

[5] Mark A. Moser. Unmanned aircraft systems - is the commander getting what is needed? *Army War College Strategic Studies Institute of Carlisle Barracks PA*, 2011.

[6] Samya Kullab. Ukraine is building an advanced army of drones. for now, pilots improvise with duct tape and bombs. 2023. https://apnews.com/article/drones-ukraine-war-russia-innovation-technology-589f1fc0e0db007ea6d344b197207212.

[7] Scott Simon. How the use of drones in ukraine has changed war as

we know it. 2023. https://www.npr.org/2023/08/05/1192343968/how-the-use-of-drones-in-ukraine-has-changed-war-as-we-know-it.

[8] How are 'kamikaze' drones being used by Russia and Ukraine? 2023. https://www.bbc.com/news/world-62225830.

[9] Iranian Uavs in Ukraine: A visual comparison. 2023. https://www.dia.mil/Portals/110/Documents/News/Military_Power_Publications/UAV_Book.pdf.

[10] Julius A Marshall, Paul Binder, and Andrea L'Afflitto. An optimization-based guidance system for tactical multi-rotor uavs mapping contested environments. In *Conference on Control Technology and Applications (CCTA)*, pages 249–254. IEEE, 2023.

[11] Julius A. Marshall, Robert B. Anderson, Wen-Yu Chien, Eric N. Johnson, and Andrea L'Afflitto. A guidance system for tactical autonomous unmanned aerial vehicles. *J. Intell. Robotics Syst.*, 103(4), dec 2021.

[12] Sean E. Eagen. Robust optimal control of a tailsitter UAV. Master's thesis, Virginia Polytechnic Institute and State University, 2021.

[13] Tyto Robotics. Series 1585 Thrust Stand thrust stand for drone propulsion systems. 2023. https://www.tytorobotics.com/pages/series-1580-1585.

[14] R. Mehrotra and D.M. Krause. Obstacle-free path planning for mobile robots. In *Third International Conference on Image Processing and its Applications, 1989.*, pages 431–435, 1989.

[15] Lazaros Moysis, Eleftherios Petavratzis, Christos Volos, Hector Nistazakis, Ioannis Stouboulos, and Kimon Valavanis. A chaotic path planning method for 3-D area coverage using modified logistic map and a modulo tactic. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 220–227, 2020.

[16] Alex Wallar, Erion Plaku, and Donald A. Sofge. A planner for autonomous risk-sensitive coverage (PARCov) by a team of unmanned aerial vehicles. In *IEEE Symposium on Swarm Intelligence*, pages 1–7, 2014.

[17] François Charles Joseph Allaire, Gilles Labonté, Mohammed Tarbouchi, and Vincent Roberge. Recent advances in unmanned aerial vehicles real-time trajectory planning. *Journal of Unmanned Vehicle Systems*, 7(4):259–295, 2019.

[18] Daniel Ioan, Sorin Olaru, Ionela Prodan, Florin Stoican, and Silviu-Iulian Niculescu. From obstacle-based space partitioning to corridors and path planning. A convex lifting approach. *IEEE Control Systems Letters*, 4(1):79–84, 2020.

[19] Sergei Savin. An algorithm for generating convex obstacle-free regions based on stereographic projection. In *International Siberian Conference on Control and Communications*, pages 1–6, 2017.

[20] Julius A. Marshall, Grant I. Carter, and Andrea L'Afflitto. *Model Reference Adaptive Control for Prescribed Performance and Longitudinal Control of a Tail-Sitter UAV.*

[21] Nicolas Michel, Peng Wei, Zhaodan Kong, Anish Kumar Sinha, and Xinfan Lin. Modeling and validation of electric multirotor unmanned aerial vehicle system energy dynamics. *eTransportation*, 12:100173, 2022.

[22] Ahmed Aboelezz, David Wetz, Jane Lehr, Pedram Roghanchi, and Mostafa Hassanalian. Intrinsically safe drone propulsion system for underground coal mining applications: Computational and experimental studies. *Drones*, 7(1), 2023.

[23] Drone test stand: A test stand to safely test vertical lift vehicles, measure thrust and record test conditionsn. 2015. https://hackaday.io/project/4389-drone-test-stand.

[24] Swati Swarnkar, Hardik Parwana, Mangal Kothari, and Abhishek Abhishek. Biplane-quadrotor tail-sitter UAV: Flight dynamics and control. *Journal of Guidance, Control, and Dynamics*, 41(5):1049–1067, 2018.

[25] Rajneesh Singh, Vikram Hrishikeshavan, and Jayant Sirohi. Common research configuration for collaborative advancement of scalable VTOL UAS technologies. pages 1–11. Vertical Flight Society 75th Annual Forum and Technology Display, 05 2019.

[26] Zheng Qiao, Dong Wang, Jiahui Xu, Xinbiao Pei, Wei Su, Dong Wang, and Yue Bai. A comprehensive design and experiment of a biplane quadrotor tail-sitter UAV. *Drones*, 7(5), 2023.

[27] K. Yamaguchi, T. L. Kunii, K. Fujimura, and H. Toriya. Octree-related data structures and algorithms. *IEEE Computer Graphics and Applications*, 4(1):53–59, 1984.

[28] Donald Meagher. *Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer*. Rensselaer Polytechnic Institute. Image Processing Laboratory, 10 1980.

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

[30] Sven Koenig, Maxim Likhachev, and David Furcy. Lifelong planning A*. *Artificial Intelligence*, 155(1):93–146, 2004.

[31] Jean-Claude Latombe. *Robot Motion Planning*. Springer, 1991.

[32] Robert Anderson. *Routing and Control of Unmanned Aerial Vehicles for Performing Contact-Based Tasks*. PhD thesis, Virginia Polytechnic Institute and State University, 2021.

[33] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[34] Luís Martins, Carlos Cardeira, and Paulo Oliveira. Feedback linearization with zero dynamics stabilization for quadrotor control. *Journal of Intelligent & Robotic Systems*, 101(1):7, Dec 2020.

[35] A. Isidori. *Nonlinear Control Systems*. Springer, New York, NY, 1995.

[36] Jason Sheng-Hong Tsai, Chih-Cheng Huang, Shu-Mei Guo, and Leang-San Shieh. Continuous to discrete model conversion for the system with a singular system matrix based on matrix sign function. *Applied Mathematical Modelling*, 35(8):3893–3904, 2011.

[37] Ladislav Kavan, Ivana Kolingerova, and Jiri Zara. Fast approximation of convex hull. In *International Conference on Advances in Computer Science and Technology*, pages 101–104, Anaheim, CA, 2006.

[38] Nuria Ortigosa, Samuel Morillas, and Guillermo Peris-Fajarnés. Obstacle-free pathway detection by means of depth maps. *Journal of Intelligent and Robotic Systems*, 63:115–129, 07 2011.

[39] Markus Ortlieb, Florian-Michael Adolf, and Florian Holzapfel. Protected online path planning for UAVs over congested areas within convex regions of obstacle-free space. pages 1–9, 2021.

[40] Dilip Kumar Pratihar, Kalyanmoy Deb, and Amitabha Ghosh. Fuzzy-genetic algorithms and time-optimal obstacle-free path generation for mobile robots. *Engineering Optimization*, 32(1):117–142, 1999.

[41] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley select coursepack. Wiley, 2005.

[42] Julius Marshall. *Guidance and Control of Autonomous Unmanned Aerial Systems for Maritime Operations*. PhD thesis, Virginia Polytechnic Institute and State University, 2022.

[43] Andrea L'Afflitto. *A mathematical perspective on flight dynamics and control*, pages 1–122. Springer, London, UK, 01 2017.

[44] Eugene Lavretsky and Kevin A. Wise. *Robust and Adaptive Control with Aerospace Applications*. Springer, London, UK, 2013.

[45] Robert B. Anderson, Julius A. Marshall, Andrea L'Afflitto, and James M. Dotterweich. Model reference adaptive control of switched dynamical systems with applications to aerial robotics. *Journal of Intelligent and Robotic Systems*, 100:1265–1281, 2020.

[46] Jorge Cortes. Discontinuous dynamical systems. *IEEE Control Systems Magazine*, 28(3):36–73, 2008.

[47] Christopher Heil. *Introduction to Real Analysis*. Graduate Texts in Mathematics. Springer, 2010.

Appendices

Appendix A

First Appendix

A.1 Section one

The following tables list the parameters used for indicated fight. We list the user defined longitudinal and lateral controller's parameters, respectively.

Parameter	Hover	Forward
ϵ_x	0.15	0.15
ϵ_z	30	30
$\tilde{\Gamma}_\theta$	$1 \cdot 10^{-1}$	0
$\tilde{\Gamma}_x$	$3 \cdot 10^{-3}$	0
$\tilde{\Gamma}_z$	$3 \cdot 10^{-3}$	0
$\tilde{\Gamma}_{\Phi,1}$	$2 \cdot 10^{-3}$	0
$\tilde{\Gamma}_{\Phi,2}$	$2 \cdot 10^{-3}$	0
$\tilde{\Gamma}_{\Phi,3}$	$2 \cdot 10^{-3}$	0
$\tilde{\Gamma}_{\Phi,4}$	$2 \cdot 10^{-3}$	0
$\tilde{\Gamma}_{\Phi,5}$	$2 \cdot 10^{-3}$	0
$\tilde{\Gamma}_{\Phi,6}$	$2 \cdot 10^{-3}$	0
$\tilde{\Gamma}_{\Phi,7}$	$2 \cdot 10^{-3}$	0

Parameter	Hover	Forward
ϵ_y	0.15	0.15
ϵ_z	30	30
$k_{p,A}$	5	5
$k_{d,A}$	1.5	1.5
$k_{p,\mathrm{kin},\psi}$	2	2
$k_{p,\mathrm{kin},\phi}$	2	2
$K_{\mathrm{p}\phi,1}$	0.06	0.06
$K_{\mathrm{d}\phi,1}$	-0.10	-0.10
$K_{\mathrm{p}\psi,1}$	1.12	1.12
$K_{\mathrm{d}\psi,1}$	0.15	0.15
$K_{\mathrm{p}\phi,2}$	0.80	0.80
$K_{\mathrm{d}\phi,2}$	-0.10	-0.10
$K_{\mathrm{p}\psi,2}$	1.12	1.12
$K_{\mathrm{d}\psi,2}$	0.25	0.25
$\tilde{\Gamma}_{\mathrm{lat},1}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
$\tilde{\Gamma}_{\mathrm{lat},2}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
$\tilde{\Gamma}_{\mathrm{lat},3}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
$\tilde{\Gamma}_{\mathrm{Dyn,p}}$	$1 \cdot 10^{-1}$	$1 \cdot 10^{-1}$
$\tilde{\Gamma}_{\mathrm{Dyn,r}}$	$1 \cdot 10^{-1}$	$1 \cdot 10^{-1}$
$\tilde{\Gamma}_{\mathrm{Dyn},\rho_{\mathrm{cmd}},\mathrm{p}}$	$1 \cdot 10^{-2}$	$1 \cdot 10^{-2}$
$\tilde{\Gamma}_{\mathrm{Dyn},\rho_{\mathrm{cmd}},\mathrm{r}}$	$1 \cdot 10^{-2}$	$1 \cdot 10^{-2}$

For the hover flight, we set

$$K_x = \begin{bmatrix} 2.0234 & 0 & 0 & 1.1096 & 0.2895 & 0 & 0 & 0 & 0 \\ 0 & 4.4721 & 0 & 0 & 0 & 7.5969 & 0 & 5.3345 & 0 \\ 0 & 0 & 77.4597 & 0 & 0 & 0 & 49.8308 & 0 & 16.0271 \end{bmatrix}, \tag{A.1}$$

and for the forward flight, we manually select the values of K_x such that we set

$$K_x = \begin{bmatrix} 0.70 & 0 & 0 & 1.0 & 0.25 & 0 & 0 & 0 & 0 \\ 0 & 6.0 & 0 & 0 & 0 & 5.0 & 0 & 4.0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 & 24 & 0 & 15 \end{bmatrix}. \tag{A.2}$$

Appendix B

Third Appendix

The equations comprising the regressor vector:

$$\Phi^{\mathrm{T}}(t, x_{\mathrm{p}} = \begin{bmatrix} ||v^2|| \\ ||v^2||\alpha \\ ||v^2|| \cos(\gamma) \\ ||v^2|| \cos(\gamma)\alpha \\ ||v^2|| \sin^2(\gamma) \\ ||v^2|| \sin(\gamma)\alpha \\ g \end{bmatrix} \tag{B.1}$$