ABSTRACT

Title of Thesis: QUATERNION-BASED CONTROL FOR AGGRESSIVE TRAJECTORY TRACKING WITH A MICRO-QUADROTOR UAV

Andrew Kehlenbeck, Master of Science, 2014

Thesis directed by: Professor J. Sean Humbert
Department of Aerospace Engineering

With potential missions for quadrotor micro-air vehicles (MAVs) calling for smaller, more agile vehicles, it is important to implement attitude controllers that allow the vehicle to reach any desired attitude without encountering computational singularities, as is the case when using an Euler angle representation. A computationally efficient quaternion-based state estimator is presented that enables the Army Research Laboratory's (ARL) 100-gram micro-quadrotor to determine its attitude during agile maneuvers using only an on-board gyroscope and accelerometer and a low-power processor. Inner and outer loop attitude and position controllers are also discussed that use the quaternion attitude representation to control the vehicle along aggressive trajectories with the assistance of an outside motion capture system. A trajectory generation algorithm is then described that leverages the quadrotor's inherent dynamics to allow it to reach extreme attitudes for applications such as perching on walls or ceilings and flying through small openings.

QUATERNION-BASED CONTROL FOR AGGRESSIVE TRAJECTORY TRACKING WITH A MICRO-QUADROTOR UAV

by

Andrew Kehlenbeck

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2014

Advisory Committee:
Professor J. Sean Humbert, Chair/Advisor
Professor Inderjit Chopra
Professor Derek Paley

Acknowledgments

There are many people to thank who helped and supported me throughout this endeavor. I would like to start by thanking my adviser, Dr. J. Sean Humbert. Your guidance and support has made this experience more valuable than I could have ever imagined. Thank you to the Army's MAST-CTA program and UMD's Department of Aerospace Engineering for funding my graduate research and coursework. Your continuous support allowed me to stay focused on my studies and research throughout my time as a graduate student.

My graduate experience would have been a fraction of what it was without my fellow researchers at the Autonomous Vehicle Lab. Hector Escobar, Greg Gremillion, Badri Ranganathan, Dr. Joe Conroy, and Dr. Imraan Faruque - I am forever indebted to the five of you for what you have taught me over the past 2 and a half years. You have made me a better engineer and a better researcher, and for that I thank you wholeheartedly. You also, along with the rest of the lab, made the AVL an incredible place to work. Everyone provided nonstop support, advice whenever necessary, and constant entertainment, making the long days and weeks in the AVL fly by.

A great deal of thanks are in order for my family. Your support and encouragement throughout my educational career is the reason I am where I am today. Thank you to my parents Lynn and Jack Kehlenbeck and my brother Adam for never failing to be there for me to listen to my latest frustrations and provide invaluable advice and guidance. Thank you also to my grandfathers Robert Comer

and the late Harry Kehlenbeck for being the best role models I could ask for. Your stories, experiences, and advice throughout my life have led me to this moment and I owe my passion for science and knowledge to you and my parents.

Finally, thank you to my girlfriend and best friend, Sarah Ninivaggi. Your unwavering love, support, and patience was essential in getting me through the long days and frustrating nights. I appreciate everything you do for me.

Andrew Kehlenbeck

Table of Contents

List of Figures

List of Abbreviations

ARL	Army Research Lab
AVL	Autonomous Vehicle Lab, University of Maryland
BDML	Biomimetics and Dextrous Manipulation Lab, Stanford University
CG	Center of gravity
CTA	Collaborative Technology Alliance
ESC	Electronic Speed Controller
GINA	Guidance and Inertial Navigation Assistant
MAST	Micro-Autonomous Systems and Technologies
MAV	Micro Aerial Vehicle
PCB	Printed Circuit Board
PD	Proportional-Derivative Control
PID	Proportional-Integral-Derivative Control
QP	Quadratic Program
UAV	Unmanned Aerial Vehicle

Chapter 1: Introduction

1.1 Motivation and Technical Approach

The popularity of quadrotors is at an all-time high for research applications. They now serve as the standard platform for the development of new sensing, mapping, and navigation technology for small unmanned aerial vehicles (UAVs). In addition to their role in current research programs, quadrotors are also widely available commercially. Children and hobbyists alike can now purchase quadrotors that are easy to fly with little or no previous piloting experience. With the public becoming increasingly more comfortable with quadrotor UAVs, the list of potential applications, both in the military and in everyday life, continues to grow.

Typical non-aggressive quadrotor UAVs use simple controllers and state estimation algorithms that are meant only for the hover condition. However, as desired capabilities become more demanding, these controllers are insufficient for a variety of applications. Collision or obstacle avoidance during high-speed flight, for example, requires quick and aggressive maneuvering that current simplified controllers would not be able to achieve. In order to create a wider realm of potential quadrotor applications, more advanced attitude and position controllers are required.

The robotics community has made significant progress in improving the capa-

bilities of quadrotor UAVs and enabling aggressive quadrotor flight with a variety of algorithms and control methods. Numerous university research groups have focused on the advancement of many aspects of quadrotor operation over the past several years, such as dynamics and control [1], [2], [3], state estimation [4], and aggressive maneuvering [5], [6], [7]. The University of Pennsylvania's General Robotics, Automation, Sensing, and Perception (GRASP) Laboratory and the Flying Machine Arena at ETH Zurich in particular have done significant work with aggressive quadrotor flight and trajectory generation in recent years [8], [9], [10], [11] [12]. These labs have developed complex algorithms and nonlinear model-based controllers that enable agile quadrotor flight, however, these algorithms are often too computationally intensive for implementation on smaller, low-power vehicles.

The contribution of this thesis is to present a state estimation algorithm and attitude and navigation controllers designed to enable aggressive flight in small quadrotor UAVs. Using a four-dimensional quaternion attitude representation, as opposed to the much more common Euler angle method, the quadrotor can achieve extreme attitudes without a complex, trigonometric state estimator that is difficult to implement on small, low-power processors. A trajectory generation algorithm is also presented that leverages the inherent dynamics of quadrotor UAVs to create achievable trajectories that require the minimum control effort to complete. The generated trajectories are entirely compatible with the quaternion framework of the state estimator and controllers. This is demonstrated with the example application of performing an aggressive perching maneuver on a vertical wall using a specially designed dry-adhesive developed at Stanford University.

In this work, a micro-quadrotor developed jointly by the Autonomous Vehicle Laboratory (AVL) and the Army Research Laboratory (ARL) is used to demonstrate aggressive control with a quaternion attitude representation. The state estimator and attitude controller are both implemented on-board the vehicle using a GINA Mote, an avionics and communication board developed at the University of California, Berkeley. The outer-loop navigation controller is implemented off-board in Labview and uses a Vicon motion capture system to provide inertial state feedback. The on- and off-board sensors used for each part of the control scheme are outlined in Fig. 1.1. The trajectory generation algorithm uses Matlab with the convex optimization CVX Toolbox [13] to create quadrotor trajectories.

	Inner Loop			Outer Loop			
	Attitude (ϕ, θ)	Heading (ψ)	Body Rates (p, q, r)	Body Velocities (u, v, w)	Position (x, y, z)	Velocity $(\dot{x}, \dot{y}, \dot{z})$	Acceleration $(\ddot{x}, \ddot{y}, \ddot{z})$
Off-Board Feedback (Vicon)		✓		✓	✓	✓	✓
On-Board Gyroscope and Accelerometer	✓		✓				

Figure 1.1: Outline of the on- and off-board sensors used in the presented controllers.

1.2 Chapter Outline

The outline for this thesis is as follows. Chapter 2 describes the ARL MkIV micro-quadrotor, the test vehicle for the proposed aggressive flight algorithms. Chapter 3 explains the motivation for using quaternions as an attitude representation and derives the quaternion-based state estimator. Chapter 4 describes the quaternion-

based attitude controller. Chapter 5 derives the differential flatness property of a quadrotor and describes the minimum-control trajectory generation algorithm. A simulated example is also presented to demonstrate the various types of constraints that can be incorporated into the trajectory. Chapter 6 describes two trajectory-tracking controllers that leverage the quaternion attitude representation, one of which allowing for any attitude during flight. Chapter 7 presents the example application of perching on a vertical wall using Stanford's dry-adhesive perching mechanism. Chapter 8 concludes the thesis and discusses future work.

Chapter 2: Overview of ARL Microquad MkIV

The current version of the Autonomous Vehicle Laboratory's (AVL) micro-quadrotor, referred to as the ARL MkIV (Fig. 2.1), was designed jointly by AVL and the Army Research Lab (ARL) as part of a collaborative grant under the Micro Autonomous Systems and Technologies Collaborative Technology Alliance (MAST-CTA). This vehicle is the culmination of several years of development of a small-scale quadrotor with potential for autonomous flight. The ARL MkIV was designed for easy manufacturing by using the vehicle's printed circuit board (PCB) as its main structural member. The motors and battery are the only components that need to be mounted on the pre-populated PCB before it can be flown. The vehicle also includes several on-board processors for flight control and sensor integration, as well as a wireless communication board and custom electronic speed controllers (ESC).

2.1 Microquad Structure

The main structural component of the ARL MkIV micro-quadrotor is the PCB itself. This design was chosen to simplify the manufacturing process. The PCB can be ordered pre-populated with the necessary electronics and, after programming the microprocessors and mounting the motors and battery, it is ready to fly. Additional

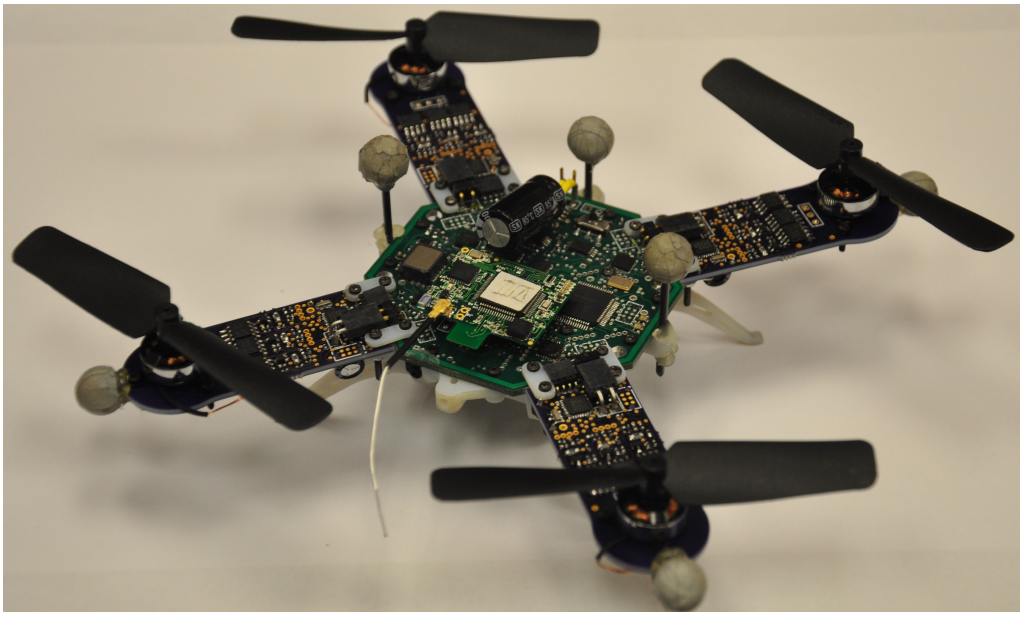


Figure 2.1: ARL MkIV.

parts are used for testing purposes, such as landing gear, sensor mounts, and Vicon motion capture markers.

In previous versions of the AVL's PCB micro-quadrotor, it was found that extensive use resulted in unpredictable motor behavior due to loose electrical connections near the motor-arm joints of the PCB. The constant flexing and vibration of the motor-arms during flight caused connections in the surface-mount components to weaken. In order to make the vehicle more structurally robust, the ESC electronics were separated onto their own motor-arm boards and the control and communication electronics were condensed to a central hub PCB. The ESC-motor PCBs are secured to the central hub PCB with plastic, rapid prototyped couplings. During crashes, these couplings snap and protect the PCBs from excessive flexing that could damage the electrical connections. The modular design of the ARL MkIV

is shown in Fig. 2.2.

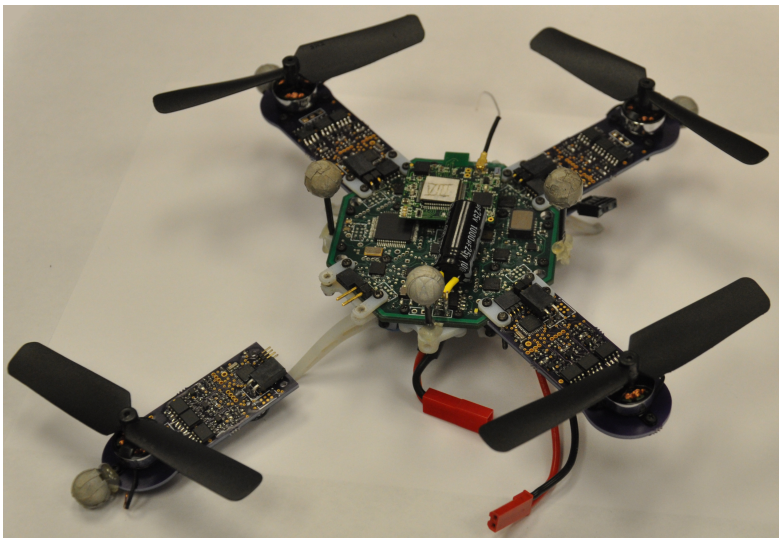


Figure 2.2: ARL MkIV with modular ESC arms.

2.2 Motors and ESCs

Custom ESCs were designed for the ARL MkIV to drive the vehicles four brushless motors (Fig. 2.3). Brushless motors were chosen over brushed motors due to their efficiency, availability, and back-EMF control capabilities. The ESCs drive the brushless motors and ensure that they are rotating at the desired speed by making control adjustments based on the back-EMF signal. These ESCs are able to update the motor speeds faster and hold steadier constant speeds than off-the-shelf ESCs of comparable size and power.

The motors used on the ARL MkIV are 4000kV micro-motors weighing 3.1 grams each. These were used with 3-inch long, injection molded rotors with a 2-inch pitch. These rotors must be balanced to reduce vehicle vibration. The balancing is

done manually with a balancing stand and thin tape is added to the rotor until it is sufficiently balanced. This motor-rotor combination produces a maximum thrust of about 40 grams.

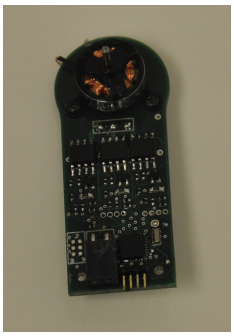
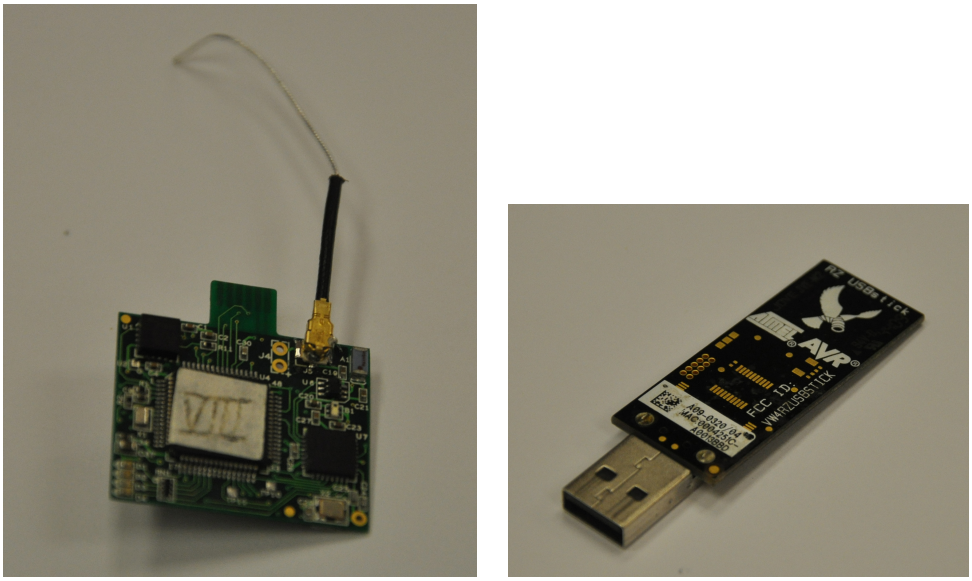


Figure 2.3: ESC arm without rotor.

2.3 Communication and Control

The ARL MkIV uses the Guidance and Inertial Navigation Assistant (GINA) mote version 2.2c developed at the University of California - Berkeley for its avionics (Fig. 2.4(a)). The mote uses an MSP430 16-bit microcontroller from Texas Instruments for processing and a 2.4 GHz Atmel AT86RF231 low-power radio for bi-directional, wireless communication to a base-station. It also has a Kionix KXSD9-1026 3-axis accelerometer, an Invensense ITG3200 3-axis gyroscope, and a Honeywell HMC5843 3-axis magnetometer on-board for inertial sensing. The mote weighs 1.8 grams and was chosen for use on the ARL MkIV due to its pre-packaged low-power wireless communication and high-quality sensor integration. Custom avionics algorithms, however, were designed and implemented in place of the standard package.

The base-station used for receiving data from and transmitting data to the ARL MkIV is a USB Atmel RZ600 base station mote (Fig. 2.4(b)).



(a) (b)

Figure 2.4: (a) UC Berkeley's GINA mote 2.2c. (b) USB Atmel basestation.

2.4 Power Supply

The ARL MkIV uses a 2-cell lithium polymer battery rated for 7.4 volts (Fig. 2.5). This voltage is regulated to 3.3 volts to power the mote and 5 volts to power the ESCs and motors. The battery weighs 24.6 grams and is mounted under the PCB body of the vehicle with a rapid prototyped bracket. With this power source, the ARL MkIV can hover with no additional payload for up to 9 minutes.

Figure 2.5: 2-cell Li-Po battery.

2.5 Coordinate System Definition

The Earth-fixed inertial coordinate system I and the body-fixed quadrotor coordinate system B are defined in Fig. 2.6. Both coordinate systems use the North-East-Down (NED) convention common to aerospace dynamics work. The vector ${}^{E}\bar{r}_{O'/O}$ is the position vector for the quadrotor, locating its center of gravity (CG) in inertial space relative to some inertial origin. For testing purposes at the Autonomous Vehicle Laboratory (AVL), the inertial origin is typically set to the center of the flight area on the floor plane.

2.6 Flight Area and Experimental Setup

A Vicon motion capture system (Fig. 2.7) is used to provide high-resolution inertial data on the current position and orientation of the ARL MkIV during flight. This data is collected at 100 Hz by a custom LabVIEW Virtual Instrument and

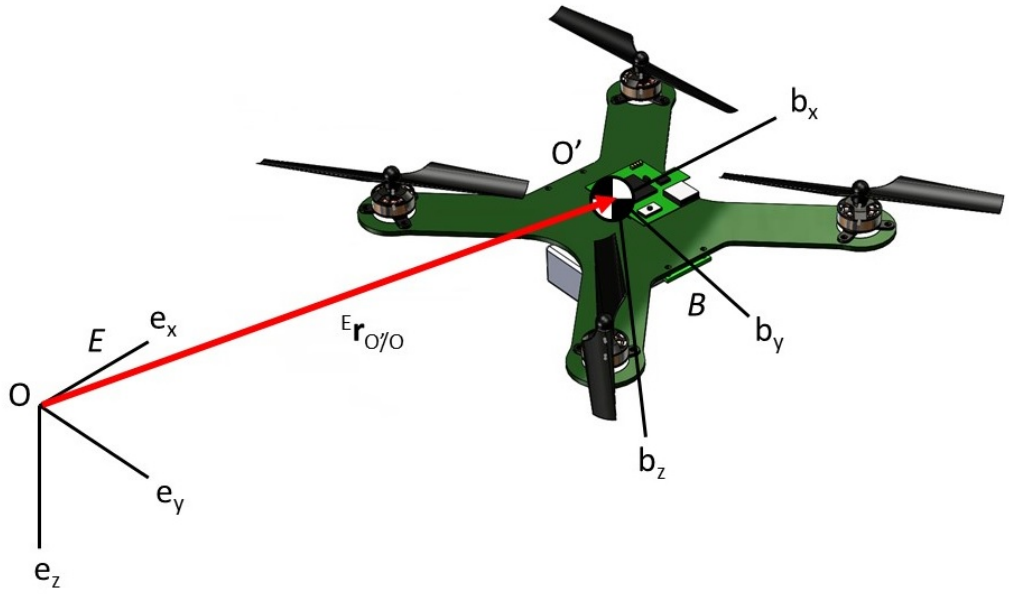


Figure 2.6: Inertial and body coordinate systems.

is then used to compute navigation-based control commands that are sent to the vehicle. LabVIEW is used for all piloted and autonomous interfacing with the ARL MkIV during flight. This allows for higher level, outer-loop navigational controllers to be designed and quickly implemented for testing. Sensor and control data is also sent back to LabVIEW from the vehicle at 100 Hz for state estimator performance analysis.

The AVL has an 18-foot square flight area for testing of the ARL MkIV (Fig. 2.8). The area is surrounded by protective netting and the floor is covered with foam padding to protect the vehicle during a crash. The Vicon cameras are set up along the perimeter of the flight area to provide inertial feedback for the entire flight space.

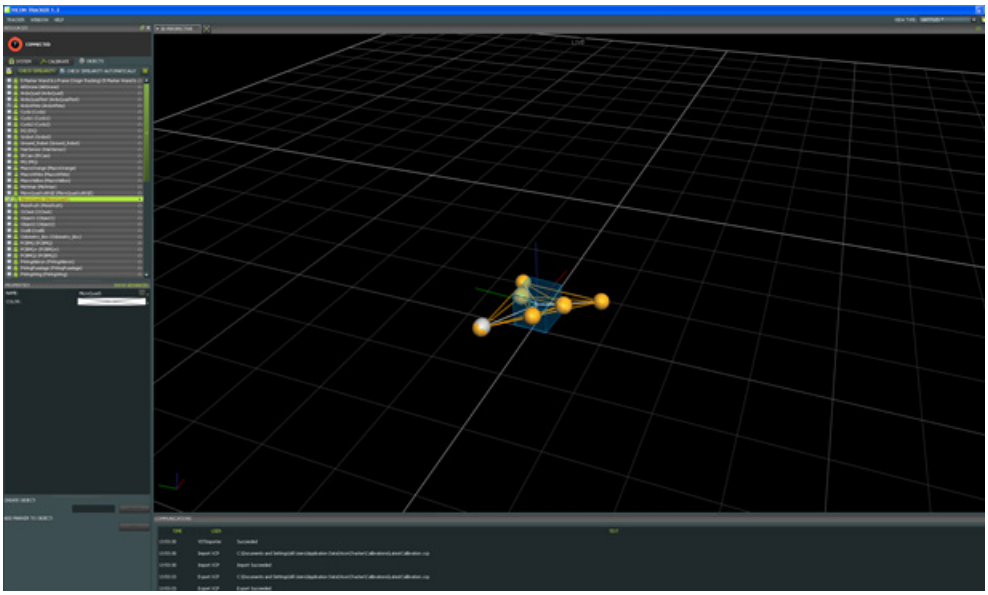


Figure 2.7: Vicon motion capture system screenshot.

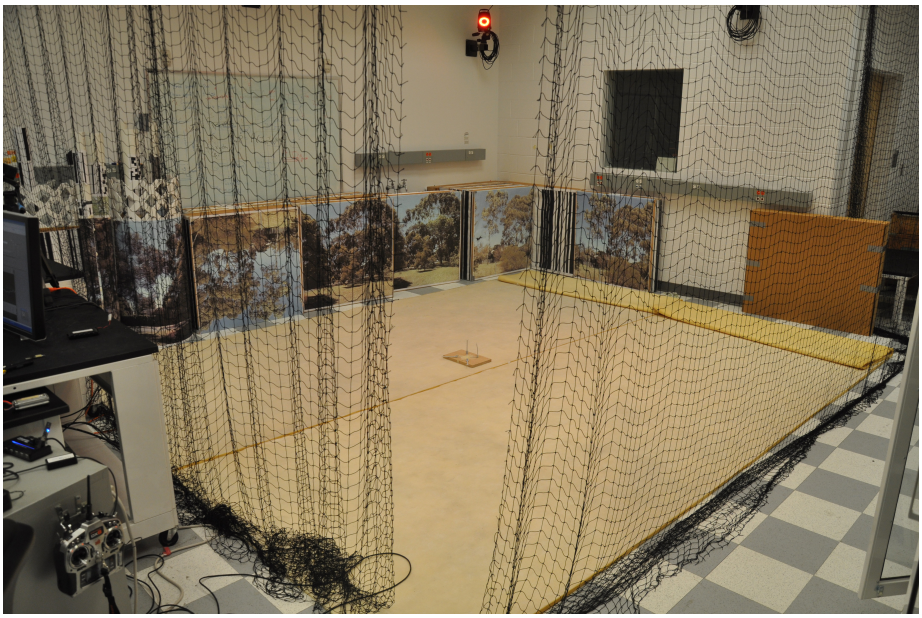


Figure 2.8: AVL's flight area.

Chapter 3: Quaternion State Estimation

In order to control the attitude of the ARL MkIV, the vehicle must have an accurate estimate of its current orientation at all times. To accomplish this, a state estimator is implemented on-board. This estimator fuses the measurements from the on-board gyroscope and accelerometer to form an estimate of the vehicle's current state. Since the ARL MkIV is typically flown in an interior room of a large building, the magnetometer is not used in the estimation scheme. Instead, Vicon is used to generate an example heading vector to correct for yaw drift.

3.1 Motivation for Quaternion Attitude Representation

Typically, aircraft define their attitude in terms of pitch, roll, and yaw Euler angles. Every set of Euler angles has inherent singularities, commonly known as gimbal lock. These singularities are a result of the trigonometric-based derivation of Euler angles and they occur at attitudes that are not defined by a unique sequence of pitch, roll, and yaw angles. For the classic 3-2-1 aviation Euler angles, singularities exist at 90 degrees pitch. For many flight regimes, this method of attitude representation is sufficient and will not result in computational singularities. However, if the vehicle is designed to have the ability to pass through these singularities, ad-

ditional computational steps must be taken or an alternative, higher-order attitude representation must be used.

The ARL MkIV is designed to operate in aggressive flight regimes that may include periods of inverted flight during certain maneuvers. Therefore, Euler angles alone are not sufficient for representing the vehicle's attitude. Instead, a quaternion attitude representation is used to provide a full description of the vehicle's orientation without the need for handling Euler angle singularities computationally.

There are several other advantages to using a quaternion attitude representation over Euler angles. For vehicles that reach attitudes greater than about 5 degrees in pitch or roll, solving numerous trigonometric functions are necessary for the use of Euler angles. Trigonometric functions are computationally expensive to solve and would potentially slow down the control loop speed on-board the vehicle. The small angle approximation can be used with Euler angles for attitudes less than 5 degrees, but this severely limits the capabilities of the quadrotor. Quaternions, however, require a single trigonometric function only when a non-zero yaw angle is included in the orientation. Otherwise, quaternion operations are solely algebraic and computationally inexpensive. It is also simpler to smoothly interpolate between two orientations using quaternions than it is with Euler angles.

The main advantage Euler angles have over quaternions is their ease of visualization. Given a pitch, roll, and yaw angle, it is much simpler to visualize the orientation of the aircraft than it is if given a quaternion. This is strictly a user disadvantage, however, and does not affect the utility of quaternions once implemented in code. Quaternions also cannot be separated by degree of freedom as Euler angles

can. Converting a quaternion orientation into a usable pitch, roll, and yaw input for a quadrotor requires an extra algebraic transformation.

3.2 Overview of Quaternion Operations

Before the quaternion-based state estimator can be presented, it is essential to understand quaternions and their associated algebra. A quaternion is a 4-dimensional, hyper-complex number. The 3 complex parts, denoted as i, j, and k, are interrelated by Eq. (3.1). The set of quaternions form a non-commutative division ring under the operations of addition and multiplication. [[14]]

$$\begin{aligned} i^2 = j^2 = k^2 = ijk = -1 \\ ij = k = -ji \\ jk = i = -kj \\ ki = j = -ik \end{aligned} \tag{3.1}$$

Quaternions can be written as a vector with 4 scalar components q_0, q_1, q_2, and q_3 (Eq. (3.2)), with components q_1, q_2, and q_3 corresponding to the distance along the quaternion basis vectors of i, j, and k. The first component, q_0, is considered the scalar part of the quaternion and q_1, q_2, and q_3 together form the vector part. A quaternion is the sum of a scalar and a vector, which is not a standard mathematical operation. Therefore, the basic quaternion operations must be defined.

$$\bar{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = q_0 + iq_1 + jq_2 + kq_3 \tag{3.2}$$

Before reviewing quaternion addition and multiplication, it is worth noting that two quaternions are equal if each of their components are equal. Quaternion addition is defined as the quaternion formed by summing the corresponding components, as shown in Eq. (3.3). It can be seen that quaternion addition is both associative and commutative, and it satisfies the field properties for addition.

$$\bar{p} + \bar{q} = \begin{bmatrix} p_0 + q_0 \\ p_1 + q_1 \\ p_2 + q_2 \\ p_3 + q_3 \end{bmatrix} = (p_0 + q_0) + i(p_1 + q_1) + j(p_2 + q_2) + k(p_3 + q_3) \tag{3.3}$$

The product of a quaternion and a scalar is a straightforward operation as shown in Eq. (3.4), where e is a scalar. The scalar distributes to each of the quaternion components to form the product. The product of two quaternions, however, is slightly more involved. The products of the basis vectors were shown in Eq. (3.1). Note that these products are not commutative. Using these properties, the product of two quaternions can be calculated with Eq. (3.5). The $\otimes$ symbol represents quaternion multiplication.

$$e\bar{q} = \begin{bmatrix} eq_0 \\ eq_1 \\ eq_2 \\ eq_3 \end{bmatrix} = eq_0 + \hat{i}eq_1 + \hat{j}eq_2 + \hat{k}eq_3 \tag{3.4}$$

$$\bar{q} \otimes \bar{p} = \begin{bmatrix} q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3 \\ q_0p_1 + q_1p_0 + q_2p_3 - q_3p_2 \\ q_0p_2 - q_1p_3 + q_2p_0 + q_3p_1 \\ q_0p_3 + q_1p_2 - q_2p_1 + q_3p_0 \end{bmatrix} \tag{3.5}$$

An important concept when using quaternions to define rotations is the complex conjugate. The complex conjugate of a quaternion is a quaternion with the same scalar component and the negative vector component. The complex conjugate of quaternion $\bar{q}$ is shown in Eq. (3.6). The conjugate is typically denoted by an asterisk.

$$\bar{q}^\star = \begin{bmatrix} q_0 \\ -q_1 \\ -q_2 \\ -q_3 \end{bmatrix} \tag{3.6}$$

Another important concept for quaternion rotations is that of a quaternion norm. The norm of a quaternion is defined by Eq. (3.7), which is the root of the sum of squares of the quaternion components. A unit quaternion will have a norm of 1. A non-unit quaternion can be normalized by dividing each component by the

quaternion's norm, resulting in a unit quaternion.

$$\| \bar{q} \| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{3.7}$$

The final quaternion concept to cover before defining quaternion rotations is the inverse of a quaternion (Eq. (3.8)). The inverse of a quaternion is the quaternion conjugate divided by the squared norm of the quaternion (Eq. (3.9)). Notice that if the quaternion is of unit length, the inverse is equivalent to the quaternion conjugate. [14]

$$\bar{q}^{-1}\bar{q} = 1 \tag{3.8}$$

$$\bar{q}^{-1} = \frac{\bar{q}^{\star}}{\| \bar{q} \|^2} \tag{3.9}$$

3.3 Quaternions as an Attitude Representation

A quadrotor's attitude, or the orientation of any rigid body for that matter, can be defined in many ways, most commonly using Euler angles that define rotations about a sequence of coordinate axes. Euler's rotation theorem states, however, that any series of rotations can be defined by a single rotation about a fixed axis, called an Euler axis, which runs through a fixed point. This is the basis of the axis-angle representation of an object's orientation in space. Quaternions can be thought of as a method of representing an axis-angle attitude with four numbers. A quaternion can be formed with the axis of rotation, $\bar{u}$, and the angle of rotation,

θ. The conversion between axis-angle attitudes and the corresponding quaternion is derived using Euler's formula. The quaternion basis axes i, j, and k represent the three Cartesian axes. The conversion equation is shown in Eq. (3.10). [14]

$$\bar{q} = e^{\frac{\theta}{2}(u_x\hat{i}+u_y\hat{j}+u_z\hat{k})} = cos\left(\frac{\theta}{2}\right) + (u_x\hat{i} + u_y\hat{j} + u_z\hat{k})sin\left(\frac{\theta}{2}\right) \tag{3.10}$$

The set of all unit quaternions provides a double covering of the 3-dimensional special orthogonal group of all rotations $SO(3)$. This means that there are two unique quaternions representing every possible rotation in 3-dimensions. Conceptually, if both the axis and the rotation about that axis are negated, the same rotation will result. Therefore, a quaternion rotation is equivalent to the rotation represented by the same quaternion with a negated scalar component. This double covering will need special consideration in the quaternion attitude controller development later.

Now that a quaternion has been defined as a representation of attitude, the next step is to define the method of rotating a point in space by the rotation a quaternion is representing. This can be done with Eq. (3.11), where $\bar{p}$ is a point in space, $\bar{q}$ is a unit quaternion representing a rotation, and $\bar{r}$ is the result of rotating point $\bar{p}$ by quaternion $\bar{q}$. Recall that the inverse of a unit quaternion is equivalent to its conjugate. This operation is known as the Hamilton product, points $\bar{p}$ and $\bar{r}$ are put into the form of a pure quaternion. A pure quaternion is a quaternion whose scalar component is zero. This allows the operation to be performed with quaternion multiplication, as described in the previous section. It is worth noting that the angle of rotation about the axis represented by the quaternion abides by the

right-hand rule, where the rotation is clockwise when looking in the same direction as $\bar{u}$.

$$\bar{r} = \bar{q} \otimes \bar{p} \otimes \bar{q}^{-1} \tag{3.11}$$

It is often useful to represent a quaternion rotation with an orthogonal matrix that, when post-multiplied by a column vector representing a point in space, results in the point rotated by the quaternion. This orthogonal rotation matrix is shown in Eq. (3.12). [14]

$$R = \begin{bmatrix} q_0^2 + q_1^2 + q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{3.12}$$

Euler angles can be converted directly to a quaternion attitude with Eq. (3.13) below. This is useful when manually piloting the vehicle, allowing the joystick input to be mapped to a desired pitch, roll, or yaw angle that is then converted to a quaternion.

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \\ \cos\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} - \sin\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} \\ \cos\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} \\ \sin\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} - \cos\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \end{bmatrix} \tag{3.13}$$

3.4 Quaternion-Based State Estimator

The on-board quaternion state estimator uses only gyroscope and accelerometer measurements to compute an estimate of the current quaternion orientation during flights. The estimator is based on an optimized gradient descent algorithm that fuses the sensor measurements. This algorithm has been shown to yield comparable performance to the much more complex Kalman filter-based state estimator. [15]

The key to state estimation is to understand the advantages and limitations of each sensor measurement available. A gyroscope measures angular rate with a high signal-to-noise ratio and it can be integrated to estimate the angle of rotation about each axis. Integrating the accompanying noise, however, results in a gradual drift in the angle estimate that cannot be corrected without additional information. An accelerometer provides an absolute reference that can be used to correct for this integration drift, but acceleration measurements typically have much lower signal-to-noise ratios. Additional information about the heading of the vehicle is still required to complete the orientation estimation. Typically a magnetometer would be used to correct the heading drift, but due to the ARL MkIV's indoor operation, the Vicon system is used to create an artificial heading correction. In the future, this can be replaced by, for example, an optic flow or other vision-based sensor to maintain the vehicles initial heading. Therefore, the gyroscope, accelerometer, and Vicon heading vector are fused in the state estimation algorithm to produce an accurate attitude estimate as the vehicle moves.

The state estimation scheme involves estimating orientation from the gyro-

scope and accelerometer separately and then optimally fusing the results together to form a final state estimate [15]. Beginning with the gyroscope-based estimate, the angular velocity measurements are put into the form of a pure quaternion at discrete time k (Eq. (3.14)). This pure quaternion can then be used to calculate the quaternion derivative (Eq. (3.15)). Integrating the quaternion derivative results in a quaternion orientation estimate at discrete time k based on the gyroscope reading (Eq. (3.16)).

$$\bar{\omega}_{gyro}[k] = \begin{bmatrix} 0 \\ \omega_{x,k} \\ \omega_{y,k} \\ \omega_{z,k} \end{bmatrix} \tag{3.14}$$

$${}^{E}\dot{\bar{q}}^{B}_{gyro}[k] = \tfrac{1}{2}\,{}^{E}\hat{q}^{B}[k-1] \otimes \bar{\omega}_{gyro}[k] \tag{3.15}$$

$${}^{E}\bar{q}^{B}_{gyro}[k] = {}^{E}\hat{q}^{B}[k-1] + {}^{E}\dot{\bar{q}}^{B}_{gyro}[k]\Delta t \tag{3.16}$$

The next step is to use the acceleration measurement to calculate an estimate of orientation. The accelerometer measures the gravity vector, plus any translational accelerations, in the vehicles body frame. For the time being, it will be assumed that the quadrotor is not accelerating along its translational axes. Therefore, the accelerometer is solely measuring the gravity vector.

Since the accelerometer does not give any information on the rotation about the gravity vector, there is not a unique orientation associated with a given mea-

surement. This can be dealt with by forming an optimization problem that finds the shortest rotation between the gravity vector measured by the accelerometer and the gravity vector of the current orientation estimate. This method avoids the use of trigonometric functions that are expensive to solve with a low-power processor. The objective of this optimization problem, shown in Eq. (3.17), involves rotating the normalized gravity vector in the inertial frame to the body frame using the current orientation estimate, and then subtracting the accelerometer measurement in the body frame. To minimize this objective, the gradient descent algorithm is used due to its simplicity for on-board implementation. The algorithm for a single step is shown below in Eq. (3.18), where μ is the step size. The direction of the step is determined by the gradient of the objective function. The gradient is a function of the objective and its Jacobian, shown in Eq. (3.19). Eq. (3.20) and Eq. (3.21) show the simplified objective and Jacobian functions that are used in the actual implementation. Note that they are only algebraic functions of the acceleration measurement and orientation estimate. Normalizing the gradient does involve a square root operation, but this is the only transcendental operation in the algorithm. Note that a_x, a_y, and a_z are the acceleration components from the accelerometer.

$$f_{grav}\left({}^{E}\hat{q}^{B}[k], \hat{a}_{accel}[k]\right) = {}^{E}\hat{q}^{B}[k]^{\star} \otimes {}^{E}\bar{g} \otimes {}^{E}\hat{q}^{B}[k] - \hat{a}_{accel}[k] \tag{3.17}$$

$${}^{E}\hat{q}^{B}_{accel}[k+1] = {}^{E}\hat{q}^{B}[k] - \mu \frac{\nabla f_{grav}\left({}^{E}\hat{q}^{B}[k], \hat{a}_{accel}[k]\right)}{\| \nabla f_{grav}\left({}^{E}\hat{q}^{B}[k], \hat{a}_{accel}[k]\right) \|} \tag{3.18}$$

$$\nabla f_{grav}\left({}^{E}\hat{q}^{B}[k], \hat{a}_{accel}[k]\right) = J\left(f_{grav}\left(\dots\right)\right)^{T} f_{grav}\left(\dots\right) \tag{3.19}$$

$$f_{grav}\left({}^{E}\hat{q}^{B}[k], \hat{a}_{accel}[k]\right) = \begin{bmatrix} 2(\hat{q}_1\hat{q}_3 - \hat{q}_0\hat{q}_2) - a_x \\ 2(\hat{q}_0\hat{q}_1 + \hat{q}_2\hat{q}_3) - a_y \\ 2(\frac{1}{2} - \hat{q}_1^2 - \hat{q}_2^2) - a_z \end{bmatrix} \tag{3.20}$$

$$J_{grav} = \begin{bmatrix} -2\hat{q}_2 & 2\hat{q}_3 & -2\hat{q}_0 & 2\hat{q}_1 \\ 2\hat{q}_1 & 2\hat{q}_0 & 2\hat{q}_3 & 2\hat{q}_2 \\ 0 & -4\hat{q}_1 & -4\hat{q}_2 & 0 \end{bmatrix} \tag{3.21}$$

The Vicon-generated heading vector can be used in the same manner as the acceleration vector to correct for the vehicle heading. The heading vector is fixed along the inertial x-axis and is given in the body frame. As before, the heading vector is rotated to the body frame using the current orientation estimate and then the Vicon-heading vector is subtracted (Eq. (3.22)). The simplified equations for the objective and Jacobian are shown in Eq. (3.23) and Eq. (3.24). Note that $x_{head,x}$, $x_{head,y}$, and $x_{head,z}$ are the components of the heading vector in the body frame.

$$f_{head}\left({}^{E}\hat{q}^{B}[k], \hat{x}_{head}[k]\right) = {}^{E}\hat{q}^{B\star}[k] \otimes {}^{E}\hat{h} \otimes {}^{E}\hat{q}^{B}[k] - \hat{x}_{head}[k] \tag{3.22}$$

$$f_{head}\left({}^{E}\hat{q}^{B}[k], \hat{x}_{head}[k]\right) = \begin{bmatrix} 2(\frac{1}{2} - \hat{q}_2^2 - \hat{q}_3^2) - x_{head,x} \\ 2(\hat{q}_1\hat{q}_2 - \hat{q}_0\hat{q}_3) - x_{head,y} \\ 2(\hat{q}_0\hat{q}_2 + \hat{q}_1\hat{q}_3) - x_{head,z} \end{bmatrix} \tag{3.23}$$

$$J_{head} = \begin{bmatrix} 0 & 0 & -4\hat{q}_2 & -4\hat{q}_3 \\ -2\hat{q}_3 & 2\hat{q}_2 & 2\hat{q}_1 & -2\hat{q}_0 \\ 2\hat{q}_2 & 2\hat{q}_3 & 2\hat{q}_0 & 2\hat{q}_1 \end{bmatrix} \tag{3.24}$$

These two objective functions, one for the acceleration vector and one for the heading vector, can be combined into a single optimization program that results in a unique orientation. The functions are combined by joining the vectors, as shown in Eq. (3.25) and Eq. (3.26), for the objective and Jacobian, respectively. This results in a single gradient descent optimization problem that uses the accelerometer measurement and heading vector to compute a quaternion orientation estimate.

$$f_{accel,head}\left({}^{E}\hat{q}^{B}[k], \hat{a}_{accel}[k], \hat{x}_{head}[k]\right) = \begin{bmatrix} f_{accel}(\dots) \\ f_{head}(\dots) \end{bmatrix} \tag{3.25}$$

$$J_{accel,head} = \begin{bmatrix} J_{accel}^{T} \\ J_{head}^{T} \end{bmatrix} \tag{3.26}$$

Now that the vehicle orientation has been estimated by the gyroscope measurement and the acceleration and heading measurements, the two estimates can be fused into a single, more accurate estimate that combines the advantages of each of the sensors. The fusion is completed using a complementary filer with weight α (Eq. (3.27)).

$${}^{E}\hat{q}^{B}[k] = \alpha\, {}^{E}\hat{q}^{B}_{accel,head}[k] + (1-\alpha)\, {}^{E}\hat{q}^{B}_{gyro}[k] \quad , \quad 0 \leq \alpha \leq 1 \tag{3.27}$$

The gradient descent optimization step size as well as the complementary

weight are chosen with the method developed by S.O.H. Madgwick [15]. The goal of the parameter tuning is to match the weighted divergence of the gyroscope-based estimate with the weighted convergence of the accelerometer and heading-based estimate, as shown in Eq. (3.28). The gyroscope weight ρ can be made very large to ensure it is greater than the convergence rate of the gyroscope-based estimate, in result making the previous orientation estimate negligible in Eq. (3.18).

$$(1-\gamma)\beta = \gamma\frac{\mu}{\Delta t} \quad , \quad \mu = \rho \parallel {}^{E}\dot{\hat{q}}^{B}_{gyro}[k] \parallel \Delta t \quad , \quad \rho > 1 \tag{3.28}$$

The state estimation equations can then be simplified to the 3 equations shown in Eq. (3.29).

$$\begin{aligned} {}^{E}\hat{q}^{B}[k] &= {}^{E}\hat{q}^{B}[k-1] + \dot{\hat{q}}[k]\Delta t \\ {}^{E}\dot{\hat{q}}^{B}[k] &= {}^{E}\dot{q}^{B}_{\omega}[k] - \beta {}^{E}\dot{q}^{B}_{\epsilon}[k] \\ {}^{E}\dot{q}^{B}_{\epsilon}[k] &= \frac{\nabla f}{\|\nabla f\|} \end{aligned} \tag{3.29}$$

3.5 On-Board Implementation

The quaternion-based state estimator was implemented on-board the GINA mote's MSP430 16-bit microprocessor. Angular rate and gravity vector measurements were obtained through the on-board Invensense ITG3200 3-axis gyroscope and the Kionix KXSD9-1026 3-axis accelerometer. In order to maintain computational efficiency and maximize the state estimator loop closure rate, all of the state estimation computations were done with scaled integers instead of floating point

values. Floating point arithmetic on the MSP430 was found to require over 10 times the number of instructions as the equivalent integer operation. Writing the entire estimator using scaled integers greatly increases the code's complexity, but it was necessary to maintain the desired loop speed while allowing time for additional processing.

Accelerometers are susceptible to high-frequency noise in general, but due to the vibrations induced on-board by the four rotors, it is necessary to run the accelerometer measurements through a low-pass filter. Therefore, a discrete-time, RC low-pass filter is used to dampen the high-frequency content of the accelerometer output before the measurements are used in the state estimator. This filter is an exponentially-weighted moving average and is shown in Eq. (3.30), where α is the smoothing factor that weighs the new raw accelerometer measurement against the previous filtered acceleration vector. The filtered acceleration vector is also normalized before being used in the state estimation. This step is computationally expensive with the required square root operation, but it is necessary to maintain the quaternion estimate's unit length.

$$\bar{a}_{filt}[i] = \alpha \bar{a}_{raw}[i] + (1-\alpha)\bar{a}_{filt}[i-1] \quad , \quad 0 \leq \alpha \leq 1 \tag{3.30}$$

The accelerometer only gives an accurate measurement of the gravity vector when the vehicle is not moving translationally. When the quadrotor is performing a maneuver, the acceleration vector will be a combination of gravity and the vehicle's inertial acceleration. Therefore, the acceleration vector is not valid for use in the

state estimator if the vehicle is in motion. Small translational motions result in slight, but acceptable, errors in the quaternion state estimate, but more aggressive motions will lead to larger errors that could then lead to instability. To protect against this, the accelerometer measurement is only used in the state estimation if its norm is close to 1 within a specified tolerance. If the norm of the acceleration measurement is much greater or less than 1, then it can be assumed that the vehicle is moving too aggressively for the acceleration vector to be useful in the state estimate. When this occurs, only the gyroscope measurement is used in the state estimation.

As explained previously, the Vicon motion tracking system is used to generate a reference yaw vector instead of using the on-board magnetometer. In the future, this yaw reference can be replaced by the magnetometer or any other inertial yaw reference available. The yaw vector reference can only be sent to the mote at 100 Hz, while the state estimation loop runs at 333 Hz. Therefore, the yaw vector is only included in the state estimation when a new measurement is available. In-between these measurements, only the accelerometer and gyroscope measurements are used. Since the drift in yaw due to angular rate integration is relatively slow, the error in yaw between yaw vector updates is negligible.

During the state estimation loops in which the norm of the acceleration vector is out of the acceptable range and there is no updated yaw vector, only the gyroscope measurement is used. In this case, the angular rate vector from the gyroscope is used to calculate the quaternion rate. The quaternion rate is then directly integrated and added to the previous quaternion estimate. The quaternion is renormalized after this operation to maintain its unit length. The full state estimation loop is shown

in the flowchart in Fig. 3.1.

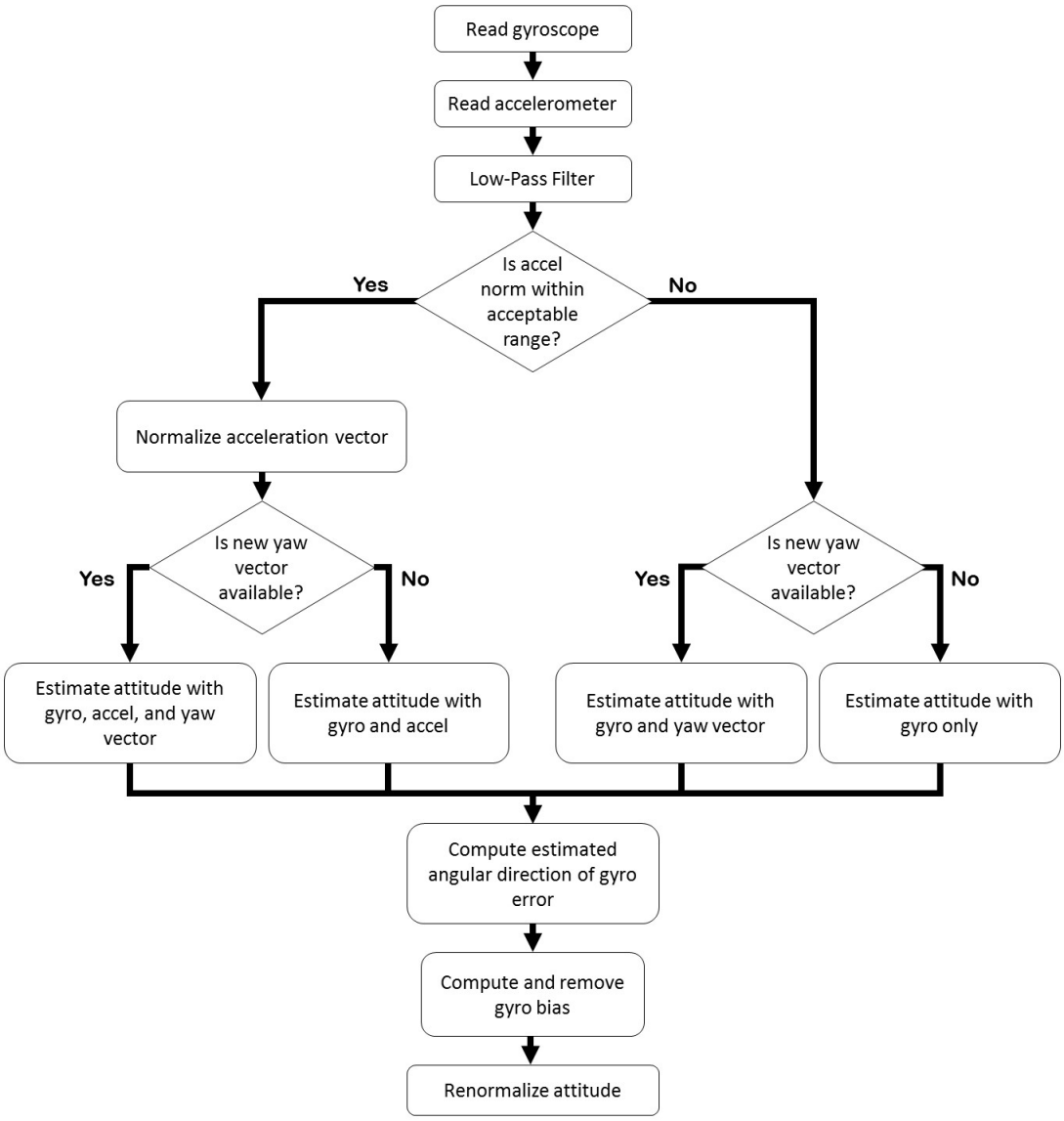


Figure 3.1: Flowchart for single iteration of quaternion state estimator.

3.6 Performance of Quaternion-Based State Estimator

Fig. 3.2 shows the typical performance of the quaternion-based state estimator, comparing the estimate to the Vicon-based attitude. The Vicon measurement is

considered the ground-truth due to its sub-millimeter precision in triangulating the position of each of the retro-reflective markers on the vehicle. For this test, the quadrotor was flown around randomly to demonstrate the estimator's ability to track orientation changes up to almost 40 degree tilt angles. A simulated heading vector calculated from Vicon data was used for the heading correction part of the state estimator. The estimate is calculated as a quaternion, as explained previously, but for easier interpretation, the corresponding Euler angles are shown.

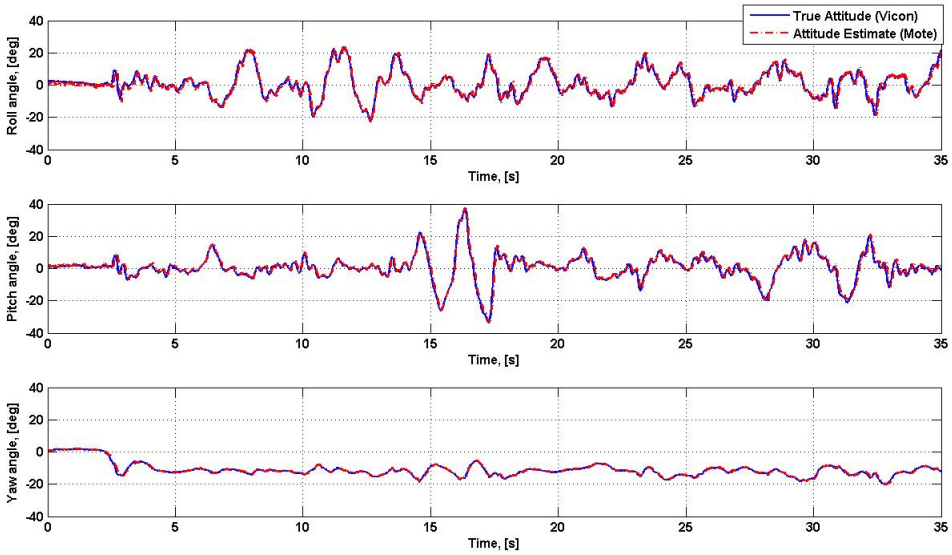


Figure 3.2: Typical results for quaternion-based state estimator in Euler angles.

The error in the attitude estimate is shown in Fig. 3.3 using Euler angles. There is a high-frequency noise component due to the effect of motor vibration on the gyroscope measurements as well as the natural high-frequency noise of the accelerometer measurements. The attitude error does not exceed 5 degrees for any of the Euler angles. The Root-Mean-Square (RMS) errors in the Euler angle estimates

are 2.2 degrees for pitch, 1.5 degrees for roll, and 0.8 degrees for yaw.

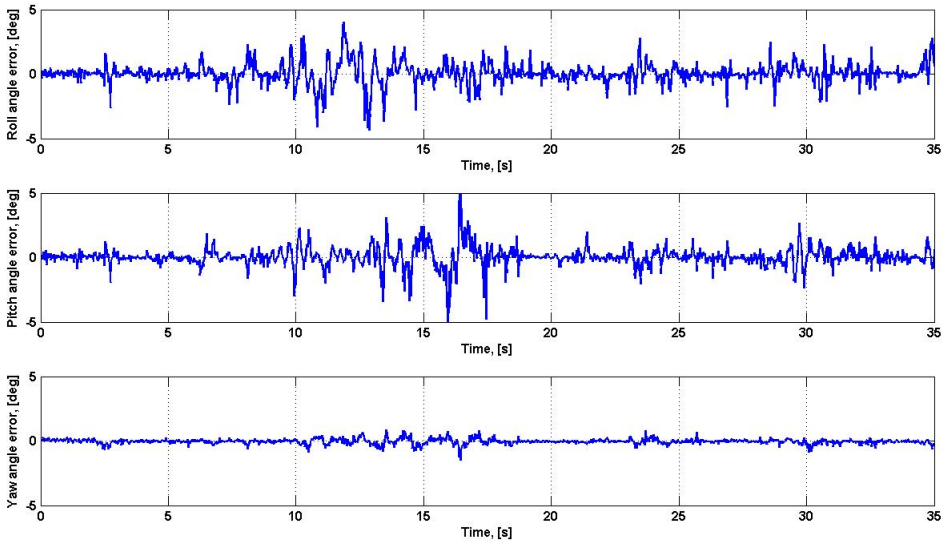


Figure 3.3: Error in Euler state from Fig. 3.2.

The test results shown in Fig. 3.4 are from the same state estimator, but without the heading correction step. It is clear that without the heading correction, the yaw angle drifts significantly over time. The error between the Vicon ground-truth attitude and the on-board state estimate is plotted in Fig. 3.5. Since the heading correction also assists the estimate along the pitch axis, the pitch angle estimate also deteriorates. This test was conducted with the motors on at a throttle similar to that of hover. The vehicle was rotated randomly by hand to achieve a larger range of attitudes than could be achieved in standard flight.

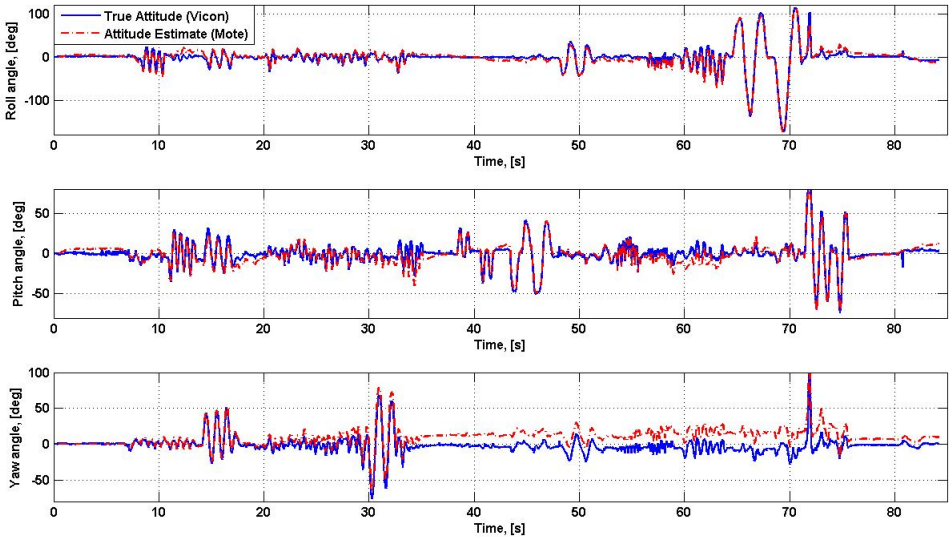


Figure 3.4: Example of state estimate without heading vector correction.

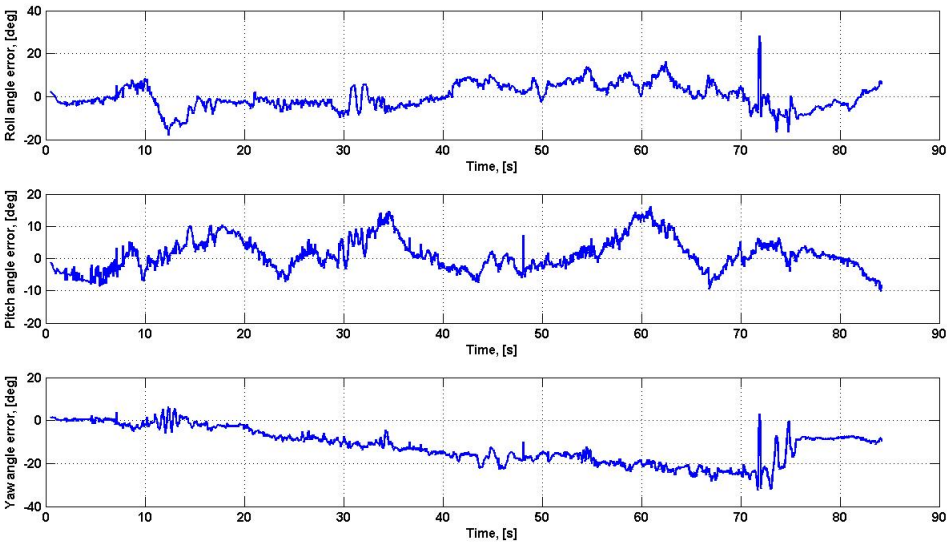


Figure 3.5: Error in Euler state from Fig. 3.4.

Chapter 4: Quaternion-Based Attitude Controller

4.1 Controller Overview

With an accurate estimate of the micro-quadrotor's attitude available, an inner-loop attitude controller can then be implemented. This attitude controller drives the vehicle's current estimated attitude to some desired attitude. This desired attitude is generated by a pilot or an outer-loop navigational controller. The use of quaternions to define attitude makes the attitude controller more complex than the typical Euler angle-based attitude controller. When using Euler angles, the attitude is already decoupled into the same axes as the motor commands necessary to rotate the vehicle in pitch, roll, and yaw. When using quaternions, however, the control needs to be decoupled into the pitch, roll, and yaw axes before the commands can be mapped to the motors.

The first step in the attitude control algorithm is to calculate the error between the current and desired quaternion attitudes. This is the quaternion corresponding to the shortest rotation required to get the vehicle to the desired attitude. The error quaternion can be calculated with Eq. (4.1). The vector part of the error quaternion is the axis of rotation expressed in the inertial frame and scaled by the sine of half the angle of rotation. Therefore, the vector can be mapped to a pitch, roll, and yaw

rotation by expressing it in the body frame of the vehicle. This operation can be done using the current quaternion attitude estimate (Eq. (4.2)). [16]

$$\begin{bmatrix} q_{0,e} \\ q_{1,e} \\ q_{2,e} \\ q_{3,e} \end{bmatrix} = \begin{bmatrix} q_{0,d} & q_{1,d} & q_{2,d} & q_{3,d} \\ -q_{1,d} & q_{0,d} & q_{3,d} & -q_{2,d} \\ -q_{2,d} & -q_{3,d} & q_{0,d} & q_{1,d} \\ -q_{3,d} & q_{2,d} & -q_{1,d} & q_{0,d} \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{4.1}$$

$$\begin{bmatrix} q_{1,e} \\ q_{2,e} \\ q_{3,e} \end{bmatrix}_B = \begin{bmatrix} 2q_0^2 + 2q_1^2 - 1 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & 2q_0^2 + 2q_2^2 - 1 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & 2q_0^2 + 2q_3^2 - 1 \end{bmatrix} \begin{bmatrix} q_{1,e} \\ q_{2,e} \\ q_{3,e} \end{bmatrix}_E \tag{4.2}$$

The pitch, roll, and yaw control commands can then be calculated using a proportional-derivative (PD) type feedback controller (Eq. (4.3)). Here, K is a diagonal gain matrix for proportional control and C is a diagonal gain matrix for derivative control, where p, q, and r are the angular rates as measured by the gyroscope. The sign of K determines whether the vehicle is commanded to rotate in the shortest direction or the longest direction to the desired quaternion. This is due to the fact that all unit quaternions double cover the $SO(3)$ group of all rotations in 3-dimensions. Therefore, the sign of K is set to be the same as the sign of the $q_{0,e}$ each iteration to ensure the vehicle is always commanded to rotate in the shortest direction. [16]

$$\begin{bmatrix} u_{pitch} \\ u_{roll} \\ u_{yaw} \end{bmatrix}_I = -K \begin{bmatrix} q_{1,e} \\ q_{2,e} \\ q_{3,e} \end{bmatrix}_B - C \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{4.3}$$

The control commands are then mapped to each motor as shown in Eq. (4.4), with the motor numbering shown in Fig. 4.1. This is the typical arrangement for an X-configuration quadrotor.

$$\begin{aligned} \Omega_1 &= \frac{T}{4} + u_{pitch} + u_{roll} + u_{yaw} \\ \Omega_2 &= \frac{T}{4} - u_{pitch} + u_{roll} - u_{yaw} \\ \Omega_3 &= \frac{T}{4} + u_{pitch} - u_{roll} - u_{yaw} \\ \Omega_4 &= \frac{T}{4} - u_{pitch} - u_{roll} + u_{yaw} \end{aligned} \tag{4.4}$$

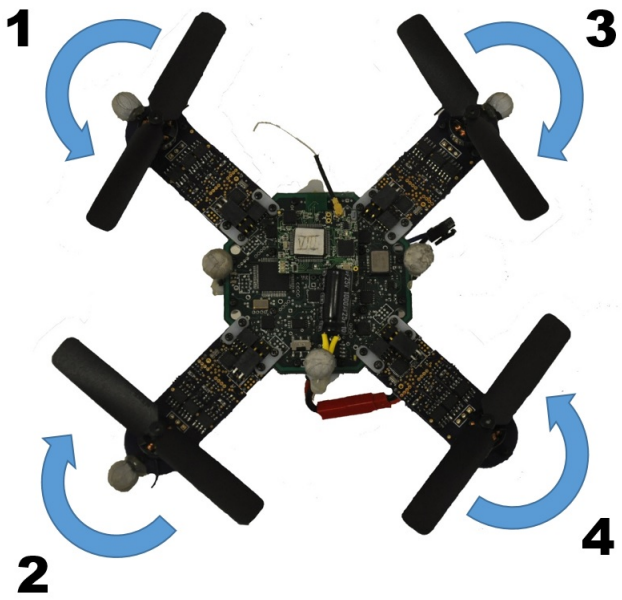


Figure 4.1: Numbering and rotation scheme for motors of ARL MkIV.

4.2 Performance of Attitude Controller

This linear state feedback quaternion attitude controller is globally asymptotically stable about the origin, which in this case corresponds to a hover attitude [17], [18]. Simulation results for this controller are shown in Fig. 4.2. In the top plot, the desired and actual quaternions are shown. The desired attitude changes in steps to illustrate the smooth quaternion transitions between various attitudes. In the bottom plot, the corresponding Euler angles of the actual attitude are shown. Note that for attitudes crossing the Euler singularities, there is a discontinuity in the Euler attitude. These discontinuities are not present in the top plot since a quaternion representation exists for all possible attitudes. This simulation demonstrates the main advantage of the quaternion attitude representation over that of Euler angles.

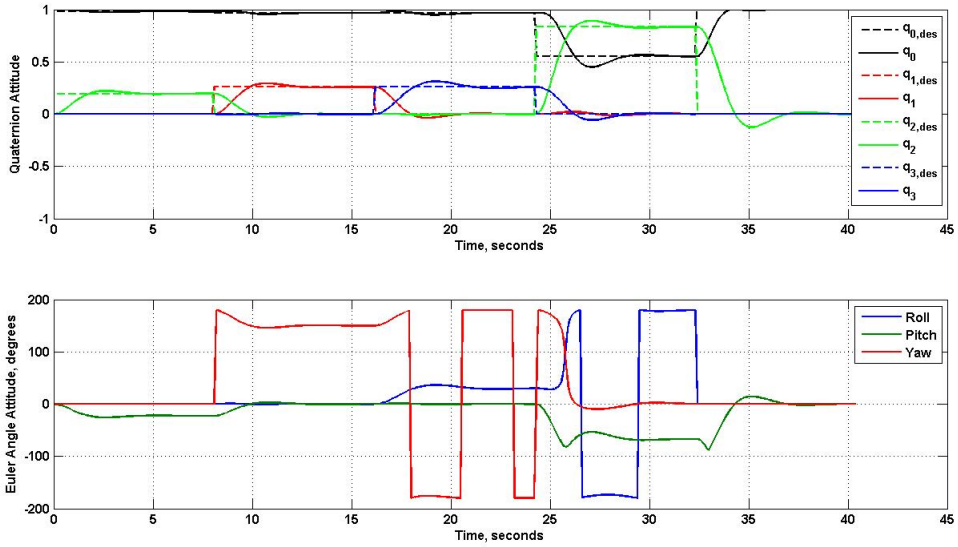


Figure 4.2: Simulation of quaternion controller compared to Euler angle representation.

Chapter 5: Optimal Trajectory Generation Algorithms

5.1 Differential Flatness of a Quadrotor

The concept of differential flatness in a non-linear dynamic system was first introduced by Fleiss et al [19]. It is a commonly leveraged property of quadrotor UAVs due to the simplified manner in which all of the vehicle states can be represented. In a differentially flat system, all of the system states can be represented by a set of outputs, called flat outputs, equal in number to the number of inputs, and their derivatives. If the system has states $x \in R^n$ and inputs $u \in R^m$, then the system is flat with outputs $y \in R^m$ if the states and inputs can be written as a function of the flat outputs and the flat output derivatives (Eq. (5.1)). [20]

$$\begin{aligned} y &= y(x, u, \dot{u}, ..., y^{(q)}) \\ where\ x &= x(y, \dot{y}, ..., y^{(q)}) \\ u &= u(y, \dot{y}, ..., y^{(q)}) \end{aligned} \tag{5.1}$$

Quadrotors have been shown to be a differentially flat system with 4 flat outputs [21]. These flat outputs are the inertial position of the vehicle, x, y, and

z, and the yaw angle, ψ. The rest of the vehicle states as well as the inputs can be written as functions of these flat outputs and their derivatives. Typically, the quadrotor attitude states are expressed as Euler angles, but in this case quaternions will be used to define the vehicle's orientation. Before the vehicle states can be written in terms of the flat outputs, the translational and rotational dynamics of the quadrotor system must be defined.

5.1.1 Quadrotor Equations of Motion Using Quaternions

The translational dynamics for a typical quadrotor can be written using Newton's Second Law of Motion (Eq. (5.2)). The acceleration term consists of the vehicle's acceleration vector in the inertial frame including gravity. The force term consists of the force produced by the rotors in the inertial frame. The ARL MkIV quadrotor is unable to alter its thrust vector, as is the case with most quadrotors, so the force vector in the body frame has a constant direction along the vehicle's body z-axis. This body frame force is then rotated to the inertial frame using the vehicle's current quaternion orientation. This equation of motion is shown in Eq. (5.3). Note that the acceleration and force vectors must be written in pure quaternion form.

$$\boldsymbol{F} = m\boldsymbol{a} \tag{5.2}$$

$$\boldsymbol{q}^{\star} \otimes \begin{bmatrix} 0 \\ {}^{B}\boldsymbol{F} \end{bmatrix} \otimes \boldsymbol{q} = m \begin{bmatrix} 0 \\ {}^{I}\ddot{\boldsymbol{r}} + {}^{I}\boldsymbol{g} \end{bmatrix} \tag{5.3}$$

The rotational dynamics of a quadrotor can be derived from Euler's rotational

equations of motion, shown in Eq. (5.4). The inertia matrix in denoted $\mathbb{I}$, while ${}^{B}\boldsymbol{M}$ and ${}^{I}\boldsymbol{\omega}^{B}$ are the moment and angular velocity vectors, respectively, expressed in the body frame.

$$ {}^{B}\boldsymbol{M} = \mathbb{I}^{I}\boldsymbol{\omega}^{B} \times \mathbb{I}^{I}\boldsymbol{\omega}^{B} \tag{5.4} $$

The inputs to the quadrotor system can be modeled as forces at each of the four motors along the body z-axis. Therefore, the total force on the vehicle is the sum of the motor forces. The moments about the body x- and y-axes are functions of the motor forces and the normal distance between the axis and the motors, d. The moment about the body z-axis is a function of the motor forces and some drag coefficient, c. These equations are shown as a single matrix equation in Eq. (5.5). The vehicle inputs can be further improved by defining the relation between the rotor RPM and the resulting force, since the commanded throttle level of each motor maps linearly to a PWM signal, which then maps linearly to a rotor speed. The simplified rotor speed to force relationship is typically modeled as a quadratic, as shown in Eq. (5.6), where k is some gain and ω_i is the i^{th} rotor RPM [22].

$$ \begin{bmatrix} f_{tot} \\ {}^{B}M_x \\ {}^{B}M_y \\ {}^{B}M_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ d & d & -d & -d \\ d & -d & d & -d \\ c & -c & -c & c \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{5.5} $$

$$ f_i = k\omega_i^2 \tag{5.6} $$

5.1.2 State and Input Equations from Flat Outputs

As stated previously, the flat outputs for a standard quadrotor are the inertial position and the yaw angle. These outputs will be denoted by the vector $\boldsymbol{\chi}$ (Eq. (5.7)). All of the quadrotor states can be written as a function of these four flat outputs and the flat output derivatives. These states include the position, velocity, and acceleration of the vehicle's center of mass, as well as the orientation, rotational velocity, and rotational acceleration of the vehicle.

$$\boldsymbol{\chi} = \begin{bmatrix} x \\ y \\ z \\ \phi \end{bmatrix} \tag{5.7}$$

Position and Orientation from Flat Outputs

The mapping from the flat outputs to the position, velocity, and acceleration of the quadrotors center of mass expressed in the inertial coordinate frame is trivial, as shown in Eq. (5.8).

$$\begin{aligned} [x, y, z]^T &= [\chi_1, \chi_2, \chi_3]^T \\ [\dot{x}, \dot{y}, \dot{z}]^T &= [\dot{\chi}_1, \dot{\chi}_2, \dot{\chi}_3]^T \\ [\ddot{x}, \ddot{y}, \ddot{z}]^T &= [\ddot{\chi}_1, \ddot{\chi}_2, \ddot{\chi}_3]^T \end{aligned} \tag{5.8}$$

The mapping from the flat outputs to the quadrotors quaternion orientation can be derived using fundamental quaternion principles and the inherent properties of a standard quadrotor. A quaternion orientation can be formulated as a rotation θ about some axis $\hat{\boldsymbol{n}}$, as shown in Eq. (5.9). This representation will be useful in defining the vehicle orientation in terms of the flat outputs.

$$\boldsymbol{q} = \begin{bmatrix} \cos(\frac{\theta}{2}) \\ \hat{\boldsymbol{n}} \sin(\frac{\theta}{2}) \end{bmatrix} \tag{5.9}$$

Since a typical quadrotor can only produce a thrust force along its body z-axis, this axis will always be aligned with the inertial acceleration vector with an arbitrary yaw angle. The final orientation is then described by a yaw rotation about the inertial acceleration vector, which is collinear with the body z-axis. For the derivation of the orientation equations from the flat outputs, the normalized thrust vector in the body frame and the inertial frame will be defined as ${}^{B}\boldsymbol{F}$ and ${}^{I}\boldsymbol{F}$, respectively. The normalized body frame thrust vector is always $[0\,,0\,,-1]^T$ and the normalized inertial frame thrust vector is defined in Eq. (5.10).

$${}^{I}\hat{\boldsymbol{F}} = \frac{1}{\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} - g)^2}} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} - g \end{bmatrix} \tag{5.10}$$

The quadrotor's orientation can be found by calculating the quaternion rotation required to match the direction of the body frame thrust vector to that of the inertial frame thrust vector and correcting for the yaw angle. To define this rotation as a quaternion, a rotation vector normal to the plane of the body and

inertial thrust vectors is needed as well as the sine and cosine of half the rotation angle about that vector. The cosine and sine of the rotation angle can be calculated with an inner product and cross product, respectively, as shown in Eq. (5.11). The rotation vector $\hat{\boldsymbol{n}}$ is solved for in Eq. (5.12). [23]

$$\begin{aligned} {}^{B}\hat{\boldsymbol{F}} \cdot {}^{I}\hat{\boldsymbol{F}} &= \| \, {}^{I}\hat{\boldsymbol{F}} \, \| \| \, {}^{B}\hat{\boldsymbol{F}} \, \| \cos\theta = \cos\theta \\ {}^{B}\hat{\boldsymbol{F}} \times {}^{I}\hat{\boldsymbol{F}} &= \| \, {}^{I}\hat{\boldsymbol{F}} \, \| \| \, {}^{B}\hat{\boldsymbol{F}} \, \| \sin\theta\hat{\boldsymbol{n}} = \sin\theta\hat{\boldsymbol{n}} \end{aligned} \tag{5.11}$$

$$\hat{\boldsymbol{n}} = \frac{{}^{I}\hat{\boldsymbol{F}} \times {}^{B}\hat{\boldsymbol{F}}}{\sin\theta} = \frac{{}^{I}\hat{\boldsymbol{F}} \times {}^{B}\hat{\boldsymbol{F}}}{\sqrt{1-\cos^2\theta}} = \frac{{}^{I}\hat{\boldsymbol{F}} \times {}^{B}\hat{\boldsymbol{F}}}{\sqrt{1-\left({}^{B}\hat{\boldsymbol{F}}^{T}\,{}^{I}\hat{\boldsymbol{F}}\right)^2}} \tag{5.12}$$

Equations for the cosine and sine of the half-angle of rotation can be derived from the cosine and sine values found with the inner and cross products. This derivation uses the half-angle trigonometric identities. The equations for the half-angle cosine and sine are shown in Eq. (5.13) and Eq. (5.14), respectively.

$$\cos\left(\frac{\theta}{2}\right) = \sqrt{\frac{1}{2}\left(1+\cos\theta\right)} = \sqrt{\frac{1}{2}\left(1+{}^{B}\hat{\boldsymbol{F}}^{T}\,{}^{I}\hat{\boldsymbol{F}}\right)} \tag{5.13}$$

$$\sin\left(\frac{\theta}{2}\right) = \sqrt{\frac{1}{2}\left(1-\cos\theta\right)} = \sqrt{\frac{1}{2}\left(1-{}^{B}\hat{\boldsymbol{F}}^{T}\,{}^{I}\hat{\boldsymbol{F}}\right)} \tag{5.14}$$

The rotation vector and half-angle sine and cosine can now be substituted into Eq. (5.9), resulting in the quaternion rotation without a yaw correction. This quaternion will be denoted $\tilde{\boldsymbol{q}}$ and is shown simplified in Eq. (5.15). The quaternion

orientation $\tilde{\boldsymbol{q}}$ can be corrected for yaw with Eq. (5.16), resulting in the final vehicle orientation $\boldsymbol{q}$ as a function of the flat outputs and their first 2 derivatives.

$$\tilde{\boldsymbol{q}} = \frac{1}{\sqrt{2\left(1 + {}^{B}\hat{\boldsymbol{F}}^{T}\,{}^{I}\hat{\boldsymbol{F}}\right)}} \begin{bmatrix} 1 + {}^{B}\hat{\boldsymbol{F}}^{T}\,{}^{I}\hat{\boldsymbol{F}} \\ {}^{B}\hat{\boldsymbol{F}} \times {}^{I}\hat{\boldsymbol{F}} \end{bmatrix} \tag{5.15}$$

$$\boldsymbol{q} = \tilde{\boldsymbol{q}} \otimes \begin{bmatrix} \cos\left(\frac{\psi}{2}\right) \\ 0 \\ 0 \\ \sin\left(\frac{\psi}{2}\right) \end{bmatrix} \tag{5.16}$$

Angular Velocity from Flat Outputs

The quadrotor's angular velocity in the body coordinate frame can be expressed as a function of the flat output derivatives. First, the derivative of the normalized applied force in the inertial frame (Eq. (5.17)) is calculated using the Transport Theorem (Eq. (5.18)) [24]. The body frame derivative of the normalized force vector is always zero because the quadrotor can only produce a force along its body z-axis and its normalized magnitude is unity by definition. Simplifying and solving this equation for the angular velocity ω results in Eq. (5.19). The derivative of the normalized force in the inertial frame can also be calculated by differentiating Eq. (5.17). Using the quotient rule for differentiation, the second equation for the normalized inertial force derivative is shown in Eq. (5.20).

$$ {}^{I}\hat{\boldsymbol{F}} = \frac{{}^{I}\boldsymbol{F}}{\| {}^{I}\boldsymbol{F} \|} \tag{5.17} $$

$$ \begin{gathered} \frac{{}^{I}d}{dt}\begin{bmatrix} 0 \\ {}^{B}\hat{\boldsymbol{F}} \end{bmatrix} = \frac{{}^{B}d}{dt}\begin{bmatrix} 0 \\ {}^{B}\hat{\boldsymbol{F}} \end{bmatrix} + \begin{bmatrix} 0 \\ {}^{I}\boldsymbol{\omega}^{B} \times {}^{I}\hat{\boldsymbol{F}} \end{bmatrix} \\ {}^{B}\hat{\boldsymbol{F}} = \boldsymbol{q} \otimes \begin{bmatrix} 0 \\ {}^{I}\hat{\boldsymbol{F}} \end{bmatrix} \otimes \boldsymbol{q}^{\star} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \end{gathered} \tag{5.18} $$

$$ {}^{I}\boldsymbol{\omega}^{B} = {}^{I}\hat{\boldsymbol{F}} \times {}^{I}\dot{\hat{\boldsymbol{F}}} \tag{5.19} $$

$$ {}^{I}\dot{\hat{\boldsymbol{F}}} = \frac{{}^{I}\dot{\boldsymbol{F}}}{\| {}^{I}\boldsymbol{F} \|} - \frac{{}^{I}\boldsymbol{F}({}^{I}\boldsymbol{F}^{T}{}^{I}\dot{\boldsymbol{F}})}{\| {}^{I}\boldsymbol{F} \|^{3}} \tag{5.20} $$

The inertial frame force vector is a function of mass and inertial acceleration, which is the second derivative of the flat outputs x, y, and z (Eq. (5.21)). The equation for the angular velocity can be written in terms of just the inertial acceleration vector and its derivative (Eq. (5.22)). Since this equation includes an arbitrary yaw rotation about the body z-axis, it is only used to calculate the angular velocities about the body x- and y-axes. The z-axis angular velocity is equal to the first derivative of the flat outputs ψ (Eq. (5.23)). The equations for the body angular velocities of the quadrotor are functions purely of the first, second, and third flat output derivatives.

$$ {}^{I}\boldsymbol{F} = m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} - g \end{bmatrix} = m(\boldsymbol{a} - \boldsymbol{g}) \tag{5.21} $$

$$ {}^{I}\boldsymbol{\omega}_{p,q}^{B} = \left(\frac{\hat{\boldsymbol{F}} - \boldsymbol{g}}{\parallel \hat{\boldsymbol{F}} - \boldsymbol{g} \parallel} \right) \times \left(\frac{\dot{\hat{\boldsymbol{F}}}}{\parallel \hat{\boldsymbol{F}} - \boldsymbol{g} \parallel} - \frac{(\hat{\boldsymbol{F}} - \boldsymbol{g})((\hat{\boldsymbol{F}} - \boldsymbol{g})^{T} \dot{\hat{\boldsymbol{F}}})}{\parallel \hat{\boldsymbol{F}} - \boldsymbol{g} \parallel^{3}} \right) \tag{5.22} $$

$$ {}^{I}\omega_{r}^{B} = \dot{\psi} \tag{5.23} $$

Angular Acceleration from Flat Outputs

The quadrotor's angular acceleration in the body coordinate frame can be expressed as a function of the flat output derivatives in the same manner as the body angular velocity. The Transport Theorem is used again to calculate the second derivative of the normalized applied force in the inertial frame (Eq. (5.24). This equation is further simplified in Eq. (5.25). The equation is then rearranged to solve for the angular acceleration about the body x- and y-axes ${}^{I}\boldsymbol{\alpha}_{\dot{p},\dot{q}}^{B}$ in Eq. (5.26). The second derivative of the normalized applied force ${}^{I}\ddot{\hat{\boldsymbol{F}}}$ can be found by differentiating the first derivative of the normalized applied force in Eq. (5.20) (Eq. (5.27)).

$$ \begin{gathered} \frac{{}^{B}d}{dt} {}^{I}\dot{\hat{\boldsymbol{F}}} = \frac{{}^{B}d}{dt} {}^{I}\dot{\hat{\boldsymbol{F}}} + {}^{I}\boldsymbol{\omega}^{B} \times {}^{I}\dot{\hat{\boldsymbol{F}}} \\ \text{where } {}^{I}\dot{\hat{\boldsymbol{F}}} = {}^{I}\boldsymbol{\omega} \times {}^{I}\hat{\boldsymbol{F}} \end{gathered} \tag{5.24} $$

$$ \frac{{}^{I}d}{dt} {}^{I}\dot{\hat{\boldsymbol{F}}} = {}^{I}\boldsymbol{\alpha}^{B} \times {}^{I}\hat{\boldsymbol{F}} + {}^{I}\boldsymbol{\omega}^{B} \times ({}^{I}\boldsymbol{\omega}^{B} \times {}^{I}\boldsymbol{F}) \tag{5.25} $$

$$ {}^{I}\boldsymbol{\alpha}^{B}_{\dot{p},\dot{q}} = {}^{I}\hat{\boldsymbol{F}} \times ({}^{I}\ddot{\hat{\boldsymbol{F}}} - {}^{I}\boldsymbol{\omega}^{B} \times ({}^{I}\boldsymbol{\omega}^{B} \times {}^{I}\boldsymbol{F})) \tag{5.26} $$

$$ {}^{I}\ddot{\hat{\boldsymbol{F}}} = \frac{{}^{I}d}{dt}\left(\frac{{}^{I}\dot{\boldsymbol{F}}}{\| {}^{I}\boldsymbol{F} \|} - \frac{{}^{I}\boldsymbol{F}({}^{I}\boldsymbol{F}^{T}{}^{I}\dot{\boldsymbol{F}})}{\| {}^{I}\boldsymbol{F} \|^{3}} \right) \tag{5.27} $$

As was the case with the angular velocity about the body z-axis, the angular acceleration about the body z-axis is calculated separately with Eq. (5.28). Again, this is due to the arbitrary yaw rotation involved with the definition of the inertial acceleration vector.

$$ {}^{I}\boldsymbol{\alpha}^{B}_{\dot{r}} = \ddot{\psi} \tag{5.28} $$

It can be seen from these equations that the angular acceleration is solely a function of the 1^{st} through 4^{th} derivatives of the flat outputs. Overall, a quadrotor's 12 states can be expressed as functions of the 4 flat outputs and the first 4 flat output derivatives.

Inputs from Flat Outputs

The final step in demonstrating the differentially flat dynamics of a quadrotor is to express the control inputs as functions of the flat outputs and flat output derivatives. The control inputs for a quadrotor consist of the rotation speed of each of the four rotors. These rotor speeds can be mapped to a rotor force, as previously shown in Eq. (5.6), and then to a total thrust and moments in the body frame, as previously shown in Eq. (5.5).

The total thrust commanded to the vehicle can be calculated with Eq. (5.29). This equation includes the magnitude of the inertial acceleration with gravity, making it a function of the second derivatives of the flat outputs.

$$f_{tot} = m\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} - g)^2} \tag{5.29}$$

The applied moments to the vehicle can be calculated using Euler's equations for rotation of a rigid body. These equations were stated previously in Eq. (5.4) and are shown in Eq. (5.30) with the angular velocity substitutions of p, q, and r. The angular velocity and acceleration were both shown to be functions of the flat outputs and flat output derivatives in the previous section.

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \mathbb{I} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \mathbb{I} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{5.30}$$

It has been shown that the 12 states and 4 inputs of a quadrotor can be expressed as a sole function of the 4 flat outputs and the first 4 flat output derivatives. This special differential flatness property of the typical quadrotor system can be leveraged for control purposes, as will be explained in the following section.

5.2 Trajectory Generation Using Differential Flatness

With the rapidly increasing potential for quadrotor-based applications, it is often desirable for a quadrotor to perform complex maneuvers that would be very difficult, if not impossible, for a pilot to command directly. Common examples of

such maneuvers include quick flight through narrow openings or aggressive attitude changes to perch on a vertical wall. These types of maneuvers can be performed by leveraging the property of differential flatness inherent to most quadrotors.

5.2.1 Discrete-Time Trajectory Generation

The previous section showed how the 12 states and 4 inputs of a quadrotor could be written as functions of 4 flat outputs and the first 4 derivatives of the flat outputs. These 4 flat outputs were the inertial position of the quadrotor, x, y, and z, and the yaw angle, ψ. Therefore, the sequence of incremental changes in the vehicle state can be represented by a sequence of flat output states.

Discrete Trajectory Definition

A quadrotor trajectory can be defined as a sequence of positions in the inertial frame and a corresponding yaw angle at discrete time steps $n = 0, ..., N$, as shown in Fig. 5.1. The position of the vehicle at each of the time steps is defined by Eq. (5.31). Also defined at each of the discrete positions of the trajectory are the first four derivatives of the flat outputs (Eq. (5.32)). A trajectory of this type fully defines the quadrotor's state for all steps of the maneuver.

$$p[n] = (x[n], y[n], z[n], \psi[n]) \in R^4 \tag{5.31}$$

Figure 5.1: Discrete positions to describe a quadrotor trajectory.

$$
\begin{aligned}
v[n] &= \left(\dot{x}[n], \dot{y}[n], \dot{z}[n], \dot{\psi}[n]\right) \in R^4 \\
a[n] &= \left(\ddot{x}[n], \ddot{y}[n], \ddot{z}[n], \ddot{\psi}[n]\right) \in R^4 \\
j[n] &= \left(x^{(3)}[n], y^{(3)}[n], z^{(3)}[n], \psi^{(3)}[n]\right) \in R^4 \\
s[n] &= \left(x^{(4)}[n], y^{(4)}[n], z^{(4)}[n], \psi^{(4)}[n]\right) \in R^4
\end{aligned}
\tag{5.32}
$$

Since the vehicle inputs and states are functions of the fourth derivative of the inertial position and the second derivative of the yaw angle, a trajectory that requires the minimum amount of control effort in terms of total thrust and total moments can be generated by minimizing these derivatives. To begin the setup of the optimization problem, the second derivatives of the flat outputs are chosen as the optimization variable, χ (Eq. (5.33) and Eq. (5.34)). The inertial flat outputs are separated from the yaw angle flat output to simplify the setup. They will be joined into a single optimization problem later.

$$\chi_{x,y,z} = \begin{bmatrix} a_x[1] \\ \vdots \\ a_x[n] \\ a_y[1] \\ \vdots \\ a_y[n] \\ a_z[1] \\ \vdots \\ a_z[n] \end{bmatrix} \tag{5.33}$$

$$\chi_{\psi} = \begin{bmatrix} a_{\psi}[1] \\ \vdots \\ a_{\psi}[n] \end{bmatrix} \tag{5.34}$$

Each of the other flat output derivatives must be put in terms of the second derivative and the initial values of the flat outputs. Since the trajectory is bounded and continuous along each dimension, Lebesgue's integrability condition is satisfied and Riemann sum integration can be used. The equations for the first derivatives and flat outputs are shown in Eq. (5.35) and Eq. (5.36) as functions of the second flat output derivatives. The equations are written in a general form that can be applied to each flat output dimension independently. Note that h is the fixed step size of the discrete trajectory.

$$\begin{aligned} v[n] &= v[n-1] + ha[n-1] \\ &= v[1] + h(a[1] + a[2] + ... + a[n-1]) \end{aligned} \tag{5.35}$$

$$\begin{aligned} p[n] &= p[n-1] + hv[n-1] + \frac{h^2}{2}a[n-1] \\ &= p[1] + h(n-1)v[1] \\ &+ \frac{h^2}{2}((2n-3)a[1] + (2n-5)a[2] + ... + a[n-1]) \end{aligned} \tag{5.36}$$

Finite difference differentiation is used to form an equation for the third and fourth flat output derivatives in terms of the second derivatives, as shown in Eq. (5.37) and Eq. (5.38).

$$j[n] = \frac{1}{h}(a[n] - a[n-1]) \tag{5.37}$$

$$s[n] = \frac{1}{h^2}(a[n] - 2a[n-1] + a[n-2]) \tag{5.38}$$

Objective Function Formulation

The goal of the trajectory optimization is to find a trajectory that minimizes the fourth derivative of the inertial position and the second derivative of the yaw angle. The resulting trajectory was shown previously to require the minimum control effort in terms of total thrust and applied moments on the quadrotor. The objective of the optimization problem is chosen to be the sum of the square norm of the fourth inertial derivative and second yaw derivative at each discrete time step.

For simplicity, the optimization problem for the inertial dimensions, x, y, and z will be shown first and the yaw dimension will be included later. The objective function for the inertial optimization problem is shown in Eq. (5.39). This equation can be rearranged into a quadratic function with the optimization variable $\chi_{x,y,z}$ (Eq. (5.40)). This results in a quadratic program (QP), which, when constrained by affine functions of the optimization variable, can be solved quickly and efficiently.

$$f_{x,y,z} = \sum_{n=1}^{N} ||s_{x,y,z}[n]||_2^2 \tag{5.39}$$

$$f_{x,y,z} = \chi_{x,y,z}^T P_{x,y,z} \chi_{x,y,z} + q^T \chi_{x,y,z} + r \tag{5.40}$$

The matrix $P_{x,y,z}$ of the quadratic objective can be derived by first putting the fourth derivative calculation from Eq. (5.38) into matrix form (Eq. (5.41)). Here, a_i is an array of either x, y, or z acceleration components.

$$\begin{aligned} s_i &= A_{s,blk} a_i \\ &= \begin{bmatrix} 1/h^2 & 0 & 0 & 0 & 0 & ... & 0 \\ -2/h^2 & 1/h^2 & 0 & 0 & 0 & ... & 0 \\ 1/h^2 & -2/h^2 & 1/h^2 & 0 & 0 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1/h^2 & -2/h^2 & 1/h^2 \end{bmatrix} \begin{bmatrix} a_i[0] \\ a_i[1] \\ a_i[2] \\ \vdots \\ a_i[N] \end{bmatrix} \\ &for\ i \,\epsilon\, \{x, y, z\} \end{aligned} \tag{5.41}$$

The block matrix for the fourth derivative calculation $A_{s,blk}$ can be used to

define $P_{x,y,z}$, as in Eq. (5.42). Note that the vector q and the scalar r in the typical quadratic program form are both null.

$$
\begin{aligned}
f_{x,y,z} &= \sum_{n=1}^{N} ||s_{x,y,z}[n]||_2^2 \\
&= \chi_{x,y,z}^T \begin{bmatrix} A_{s,blk}^T & 0 & 0 \\ 0 & A_{s,blk}^T & 0 \\ 0 & 0 & A_{s,blk}^T \end{bmatrix} \begin{bmatrix} A_{s,blk} & 0 & 0 \\ 0 & A_{s,blk} & 0 \\ 0 & 0 & A_{s,blk} \end{bmatrix} \chi_{x,y,z} \\
&= \chi_{x,y,z}^T P_{x,y,z} \chi_{x,y,z}
\end{aligned}
\tag{5.42}
$$

$$
where\ P_{x,y,z} = \begin{bmatrix} A_{s,blk}^T * A_{s,blk} & 0 & 0 \\ 0 & A_{s,blk}^T * A_{s,blk} & 0 \\ 0 & 0 & A_{s,blk}^T * A_{s,blk} \end{bmatrix}
$$

Now that the inertial position optimization problem has been derived, the minimization of the second derivative of the yaw angle can be set up. The optimization variable χ_ψ was defined earlier in Eq. (5.34) as an array of the yaw angular acceleration at each discrete time step. The objective function for the second yaw derivative minimization is shown in Eq. (5.43). As before, this objective function can be put into the form of a QP for efficient solving (Eq. (5.44)).

$$
f_\psi = \sum_{n=1}^{N} ||a_\psi[n]||_2^2 \tag{5.43}
$$

$$f_\psi = \chi_\psi^T P_\psi \chi_\psi + q^T \chi_\psi + r \tag{5.44}$$

The derivation of the matrix P_ψ is straightforward since the optimization variable is of the same derivative order as the term to be minimized. Therefore, the block matrix for the yaw acceleration minimization is an identity matrix (Eq. (5.45)).

$$\begin{aligned} a_\psi &= A_{a,blk} a_\psi \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & ... & 0 \\ 0 & 1 & 0 & 0 & 0 & ... & 0 \\ 0 & 0 & 1 & 0 & 0 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_\psi[0] \\ a_\psi[1] \\ a_\psi[2] \\ \vdots \\ a_\psi[N] \end{bmatrix} \end{aligned} \tag{5.45}$$

As done before, this block matrix for the second derivative of yaw angle $A_{a,blk}$ can be used to define P_ψ (Eq. (5.46)). Note that the vector q and the scalar r in the typical QP form are both null.

$$\begin{aligned} f_\psi &= \sum_{n=1}^{N} ||a_\psi[n]||_2^2 \\ &= \chi_\psi^T A_{a,blk}^T A_{a,blk} \chi_\psi \\ &= \chi_\psi^T P_\psi \chi_\psi \\ &\textit{where } P_\psi = A_{a,blk}^T A_{a,blk} \end{aligned} \tag{5.46}$$

The objective functions for the minimization of the fourth derivative of the inertial position and the second derivative of the yaw angle can be joined into a single

objective function so that only one optimization program must be solved to generate a trajectory. The joined objective function, denoted f_o, is shown in Eq. (5.47).

$$f_o = \begin{bmatrix} \chi_{x,y,z} \\ \chi_\psi \end{bmatrix}^T \begin{bmatrix} P_{x,y,z} & 0 \\ 0 & P_\psi \end{bmatrix} \begin{bmatrix} \chi_{x,y,z} \\ \chi_\psi \end{bmatrix} \tag{5.47}$$

Equality Constraints

For the inertial flat output portion of the discrete trajectory, the trajectory is constrained to an initial and final position, velocity, and acceleration. The initial position and velocity are included in the equations that calculate the final position, velocity, and acceleration, so it is not necessary to include them as separate constraints. Therefore, only the initial acceleration and final position, velocity, and acceleration are required to be constrained. These constraints can be calculated from the optimization variable $\chi_{x,y,z}$ using the equations defined in Eq. (5.35) and Eq. (5.36).

These constraint equations can be put into the form of a dot product for use in the optimization program. The initial and final acceleration constraints for time steps 1 and N are shown in Eq. (5.48). The row-vectors $m_{a1,i}$ and $m_{aN,i}$ will be used when forming the final optimization constraint matrices.

$$
\begin{aligned}
a_i[1] &= m_{a1,i} a_i \\
m_{a1,i} &= [1,\ 0,\ 0,\ ...,\ 0] \\
a_i[N] &= m_{aN,i} a_i \\
m_{aN,i} &= [0,\ 0,\ 0,\ ...,\ 0,\ 1] \\
for\ i &\epsilon \{x, y, z\}
\end{aligned}
\tag{5.48}
$$

The final position and velocity constraints are shown as dot products in Eq. (5.49). Note that these equations are functions of the initial position and velocity for the corresponding dimension. As with the acceleration constraints, the row-vectors $m_{pN,i}$ and $m_{vN,i}$ will be used when forming the final optimization constraint matrices.

$$
\begin{aligned}
p_i[N] &= m_{pN,i} a_i + p_i[1] + h v_i[1] \\
m_{pN,i} &= [\frac{(2N-3)}{2} h^2,\ \frac{(2N-5)}{2} h^2,\ ...,\ \frac{1}{2} h^2] \\
v_i[N] &= m_{vN,i} a_i + v_i[1] \\
m_{vN,i} &= [h,\ h,\ h,\ ...,\ h] \\
for\ i &\epsilon \{x, y, z\}
\end{aligned}
\tag{5.49}
$$

Position, velocity, and acceleration equality constraints can also be defined for points within the trajectory. This is useful for many situations, such as when the vehicle is required to be at a certain attitude at a certain point along the trajectory or when the vehicle must pass through a position at a certain time. Fig. 5.2 shows an arbitrary trajectory with a position constraint at steps 1, n_2, and N, velocity constraint at step N, and attitude constraints using acceleration vectors

at steps n_1 and N. The equations for mid-trajectory equality constraints are shown in Eq. (5.50).

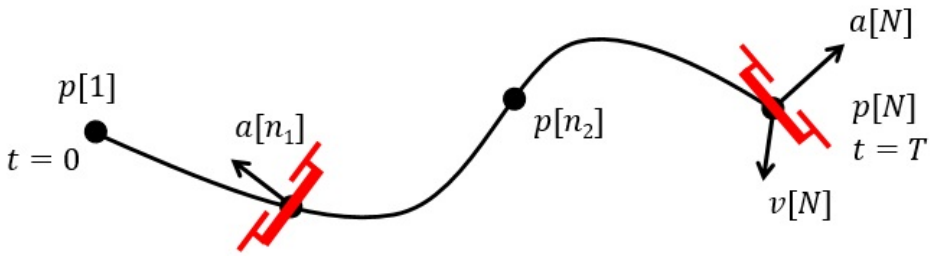


Figure 5.2: Examples of inertial equality constraints along trajectory.

$$
\begin{aligned}
&p_i[n] = m_{pn,i}a_i + p_i[1] + hv_i[1] \\
&\qquad m_{pn,i} = [\frac{(2n-3)}{2}h^2,\ \frac{(2n-5)}{2}h^2,\ ...,\ \frac{1}{2}h^2,\ 0,\ ...,\ 0] \\
&v_i[n] = m_{vn,i}a_i + v_i[1] \\
&\qquad m_{vn,i} = [h,\ h,\ h,\ ...,\ h,\ 0,\ ...,\ 0] \\
&a_i[n] = m_{an,i}a_i \\
&\qquad m_{an,i} = [0,\ ,\ ...,0,\ 1,\ 0,\ ...,\ 0] \\
&for\ i\epsilon\{x,y,z\}
\end{aligned}
\tag{5.50}
$$

These equality constraints can be combined into QP matrix equality constraint of the form $A_{eq}\chi = b_{eq}$ (Eq. (5.51) and Eq. (5.52)).

$$A_{eq,x,y,z}\chi_{x,y,z} = b_{eq,x,y,z} \tag{5.51}$$

$$A_{eq,x,y,z} = \begin{bmatrix} m_{a1,x} & 0 & 0 \\ 0 & m_{a1,y} & 0 \\ 0 & 0 & m_{a1,z} \\ m_{pN,x} & 0 & 0 \\ 0 & m_{pN,y} & 0 \\ 0 & 0 & m_{pN,z} \\ m_{vN,x} & 0 & 0 \\ 0 & m_{vN,y} & 0 \\ 0 & 0 & m_{vN,z} \\ m_{aN,x} & 0 & 0 \\ 0 & m_{aN,y} & 0 \\ 0 & 0 & m_{aN,z} \\ m_{an,x} & 0 & 0 \\ 0 & m_{an,y} & 0 \\ 0 & 0 & m_{an,z} \\ m_{pn,x} & 0 & 0 \\ 0 & m_{pn,y} & 0 \\ 0 & 0 & m_{pn,z} \\ m_{vn,x} & 0 & 0 \\ 0 & m_{vn,y} & 0 \\ 0 & 0 & m_{vn,z} \end{bmatrix}, b_{eq} = \begin{bmatrix} a_x[1] \\ a_y[1] \\ a_z[1] \\ p_x[N] - p_x[1] - hv_x[1] \\ p_y[N] - p_y[1] - hv_y[1] \\ p_z[N] - p_z[1] - hv_z[1] \\ v_x[N] - v_x[1] \\ v_y[N] - v_y[1] \\ v_z[N] - v_z[1] \\ a_x[N] \\ a_y[N] \\ a_z[N] \\ a_x[n] \\ a_y[n] \\ a_z[n] \\ p_x[n] - p_x[1] - hv_x[1] \\ p_y[n] - p_y[1] - hv_y[1] \\ p_z[n] - p_z[1] - hv_z[1] \\ v_x[n] - v_x[1] \\ v_y[n] - v_y[1] \\ v_z[n] - v_z[1] \end{bmatrix} \tag{5.52}$$

The yaw angle, yaw angular velocity, and yaw angular acceleration can be fixed to a desired value at any point during the trajectory, as illustrated in Fig. 5.3. These terms can be calculated from the optimization variable χ_ψ using Eq. (5.35) and Eq. (5.36). The constraint equation for the yaw angular acceleration are shown in dot product form in Eq. (5.53) for use in the optimization problem. The row-vectors $m_{an,\psi}$ will be used to form the final constraint matrices.

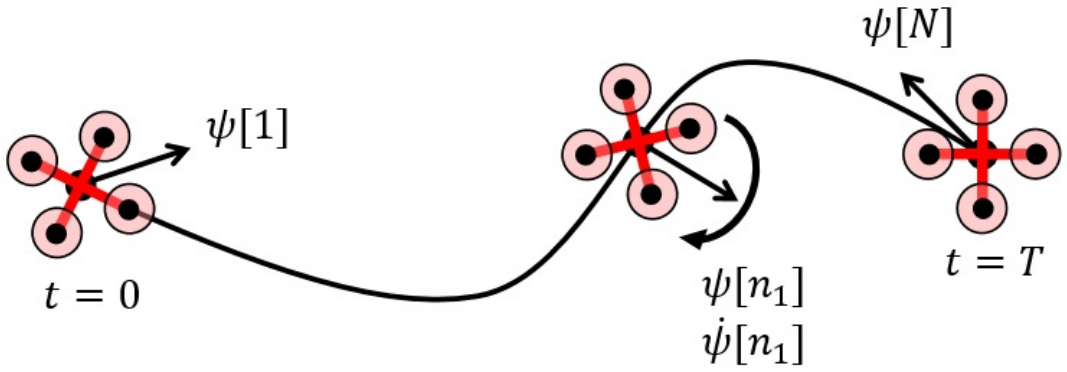


Figure 5.3: Examples of yaw equality constraints along trajectory.

$$
\begin{aligned}
a_\psi[1] &= m_{a1,\psi} a_\psi \\
m_{a1,i} &= [1,\ 0,\ 0,\ ...,\ 0] \\
a_\psi[1] &= m_{a1,\psi} a_\psi \\
m_{a1,i} &= [0,\ 0, ...,\ 1, ...,\ 0,\ 0] \\
a_\psi[n] &= m_{an,\psi} a_\psi \\
m_{aN,\psi} &= [0,\ 0,\ 0,\ ...,\ 0,\ 1]
\end{aligned}
\tag{5.53}
$$

The equations for the yaw angle and yaw angular velocity are shown in Eq. (5.54).

As with the angular acceleration constraints, the row-vectors $m_{pn,\psi}$ and $m_{vn,\psi}$ will be used to form the final constraint matrices.

$$\begin{aligned} p_\psi[n] &= m_{pn,i}a_\psi + p_\psi[1] + hv_\psi[1] \\ m_{pn,\psi} &= [\frac{(2n-3)}{2}h^2,\ \frac{(2n-5)}{2}h^2,\ ...,\ \frac{1}{2}h^2,\ 0,\ ...,\ 0] \\ v_\psi[n] &= m_{vn,i}a_\psi + v_\psi[1] \\ m_{vn,\psi} &= [h,\ h,\ h,\ ...,\ h,\ 0,\ ...,\ 0] \end{aligned} \tag{5.54}$$

These yaw equality constraints can be combined into a QP matrix inequality constraint of the form $A_{eq}\chi = b_{eq}$ (Eq. (5.55)).

$$A_{eq,\psi}\chi_\psi = b_{eq,\psi}$$

$$where\ A_{eq,\psi} = \begin{bmatrix} m_{a1,\psi} \\ \vdots \\ m_{aN,\psi} \\ m_{p2,\psi} \\ \vdots \\ m_{pN,\psi} \\ m_{v2,\psi} \\ \vdots \\ m_{vN,\psi} \end{bmatrix},\ b_{eq} = \begin{bmatrix} a_\psi[1] \\ \vdots \\ a_\psi[N] \\ p_\psi[2] - p_\psi[1] - hv_\psi[1] \\ \vdots \\ p_\psi[N] - p_\psi[1] - hv_\psi[1] \\ v_\psi[2] - v_\psi[1] \\ \vdots \\ v_\psi[N] - v_\psi[1] \end{bmatrix} \tag{5.55}$$

Inequality Constraints

It is necessary to place constraints on the minimum and maximum position, velocity, acceleration, jerk, and snap to ensure continuity in the trajectory as well as to restrict the trajectory to the dynamic limits of the quadrotor. These limits at each discrete point can be set for the trajectory by setting up matrix inequality constraints of the form $A_{ineq}\chi \leq b_{ineq}$. These inequality constraints will be derived separately in the following sections.

Position Constraints

The position at each step of the trajectory is bounded by the limits of the available space for the maneuver (Eq. (5.56). This can include corridors, known obstacles, or any other environmental constraint on position (Fig. 5.4). If there are no immediate limitations to the position, then a position constraint is not necessary for the corresponding time step.

$$p_{i,min}[n] \leq p_i[n] \leq p_{i,max}[n],\ \forall n, i \in \{x, y, z\} \tag{5.56}$$

Each position component is calculated using Eq. (5.36) in vector dot product form, shown in the assembled matrix form in Eq. (5.57). Only the rows of $A_{pos,blk}$ and $b_{pos_{blk,i,w}}$ corresponding to the time steps with position constraints are necessary to use in the final inequality constraint.

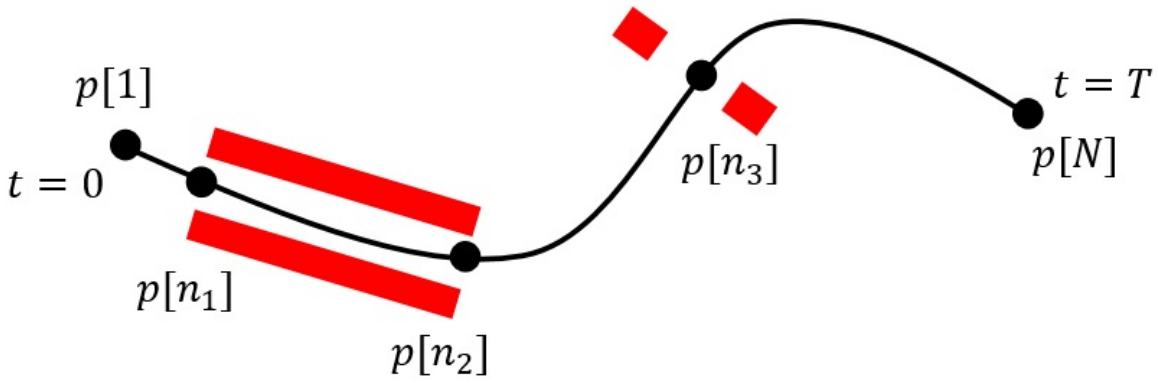


Figure 5.4: Examples of position inequality constraints along trajectory.

$A_{pos}\chi_{x,y,z} \leq b_{pos}$

where

$$A_{pos,blk} = \begin{bmatrix} \frac{1}{2}h^2 & 0 & 0 & ... & 0 \\ \frac{3}{2}h^2 & \frac{1}{2}h^2 & 0 & ... & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \frac{(2k-3)}{2}h^2 & \frac{(2k-5)}{2}h^2 & ... & ... & \frac{1}{2}h^2 \end{bmatrix}, \; b_{pos_{blk,i,w}} = \begin{bmatrix} p_{i,w}[2] - p_i[1] - hv_i[1] \\ p_{i,w}[3] - p_i[1] - hv_i[1] \\ \vdots \\ p_{i,w}[N] - p_i[1] - hv_i[1] \end{bmatrix}$$

$for\ i\epsilon\{x, y, z\}, w\epsilon\{max, min\}$

(5.57)

These position inequality constraint block matrices can be combined to form the overall position inequality matrix (Eq. (5.58)). Note that rows may be omitted if there is no position constraint at that particular time step.

$$A_{pos}\chi_{x,y,z} \leq b_{pos}$$

$where$

$$A_{pos} = \begin{bmatrix} -A_{pos,blk} & 0 & 0 \\ 0 & -A_{pos,blk} & 0 \\ 0 & 0 & -A_{pos,blk} \\ A_{pos,blk} & 0 & 0 \\ 0 & A_{pos,blk} & 0 \\ 0 & 0 & A_{pos,blk} \end{bmatrix},\ b_{pos} = \begin{bmatrix} -b_{pos,blk,x,min} \\ -b_{pos,blk,y,min} \\ -b_{pos,blk,z,min} \\ b_{pos,blk,x,max} \\ b_{pos,blk,y,max} \\ b_{pos,blk,z,max} \end{bmatrix} \tag{5.58}$$

Velocity Constraints

The velocity at each time step of the trajectory is bounded by both the limits of the quadrotor and any desired velocity bound along one of the inertial axes (Eq. (5.59)). The velocity components at each step are calculated using Eq. (5.35) in vector dot product form, as shown in Eq. (5.60).

$$v_{i,min}[n] \leq v_i[n] \leq v_{i,max}[n],\ \forall n, i \in \{x, y, z\} \tag{5.59}$$

$A_{vel}\chi_{x,y,z} \leq b_{vel}$

where

$$A_{vel,blk} = \begin{bmatrix} h & 0 & 0 & ... & 0 & 0 \\ h & h & 0 & ... & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ h & h & ... & ... & h & 0 \end{bmatrix},\ b_{vel_{blk,i,w}} = \begin{bmatrix} v_{i,w}[2] - v_i[1] \\ v_{i,w}[3] - v_i[1] \\ \vdots \\ v_{i,w}[N] - v_i[1] \end{bmatrix} \tag{5.60}$$

for $i\epsilon\{x, y, z\}, w\epsilon\{max, min\}$

The velocity inequality constraint block matrices can be combined to form the overall velocity inequality matrix (Eq. (5.61)). Note that all steps should include a velocity constraint to prevent the resulting trajectory from requiring unrealistically large speeds.

$A_{vel}\chi_{x,y,z} \leq b_{vel}$

where

$$A_{vel} = \begin{bmatrix} -A_{vel,blk} & 0 & 0 \\ 0 & -A_{vel,blk} & 0 \\ 0 & 0 & -A_{vel,blk} \\ A_{vel,blk} & 0 & 0 \\ 0 & A_{vel,blk} & 0 \\ 0 & 0 & A_{vel,blk} \end{bmatrix},\ b_{vel} = \begin{bmatrix} -b_{vel,blk,x,min} \\ -b_{vel,blk,y,min} \\ -b_{vel,blk,z,min} \\ b_{vel,blk,x,max} \\ b_{vel,blk,y,max} \\ b_{vel,blk,z,max} \end{bmatrix} \tag{5.61}$$

Acceleration Constraints

Similar to the velocity bounds, the acceleration at each step is also limited by the dynamic properties of the quadrotor as well as desired acceleration limitations along each inertial axis (Eq. (5.62)). Acceleration constraints can also be used to ensure the vehicle is near a desired attitude at a certain step along the trajectory, as illustrated in Fig. 5.5. Since the optimization variable consists of the acceleration components at each step, the inequality block matrix is simply an identity matrix and the inequality vector is the minimum or maximum allowable acceleration (Eq. (5.63)).

$$a_{i,min}[n] \leq a_i[n] \leq a_{i,max}[n], \ \forall n, i \in \{x, y, z\} \tag{5.62}$$

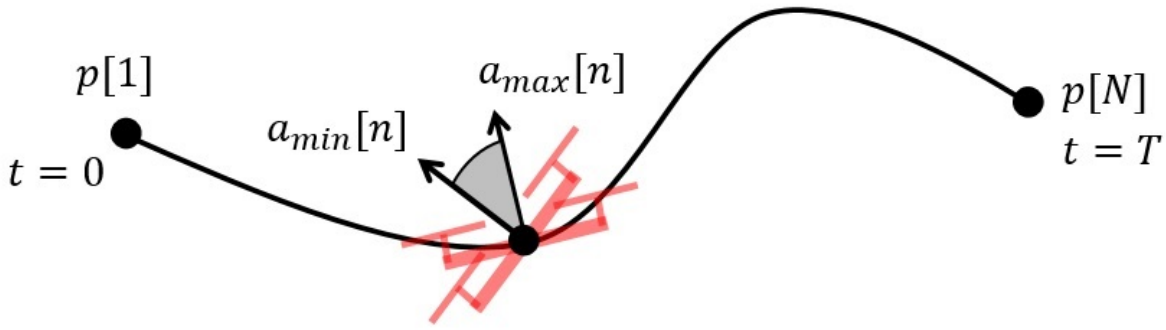


Figure 5.5: Example of acceleration inequality constraints along trajectory.

$$A_{accel}\chi_{x,y,z} \leq b_{accel}$$

$$where\ b_{accel_{blk,i,w}} = \begin{bmatrix} a_{i,w}[1] \\ a_{i,w}[2] \\ \vdots \\ a_{i,w}[N] \end{bmatrix} \tag{5.63}$$

$$for\ i\epsilon\{x,y,z\}, w\epsilon\{max,min\}$$

The acceleration inequality constraint block matrices can be combined to form the overall acceleration inequality matrix, as in Eq. (5.64). Note that all steps should include an acceleration constraint to prevent the resulting trajectory from requiring unrealistically large accelerations.

$$A_{accel}\chi_{x,y,z} \leq b_{accel}$$

$$where$$

$$A_{accel} = \begin{bmatrix} -\mathsf{I} & 0 & 0 \\ 0 & -\mathsf{I} & 0 \\ 0 & 0 & -\mathsf{I} \\ \mathsf{I} & 0 & 0 \\ 0 & \mathsf{I} & 0 \\ 0 & 0 & \mathsf{I} \end{bmatrix},\ b_{accel} = \begin{bmatrix} -b_{accel,blk,x,min} \\ -b_{accel,blk,y,min} \\ -b_{accel,blk,z,min} \\ b_{accel,blk,x,max} \\ b_{accel,blk,y,max} \\ b_{accel,blk,z,max} \end{bmatrix} \tag{5.64}$$

Jerk Constraints

As was done with the velocity and acceleration bounds, the jerk at each step

must be limited to the vehicle limits (Eq. (5.65)). It is uncommon to have known strict jerk limitations, so typically the jerk limits can be set as constant and adjusted as needed to converge to a feasible trajectory. The jerk calculation from Eq. (5.37) can be put in matrix form, as shown in Eq. (5.66).

$$j_{i,min}[n] \leq j_i[n] \leq j_{i,max}[n],\ \forall n, i \in \{x, y, z\} \tag{5.65}$$

$$A_{jerk}\chi_{x,y,z} \leq b_{jerk}$$

$$where$$

$$A_{jerk,blk} = \begin{bmatrix} -1/h & 1/h & 0 & ... & 0 & 0 \\ 0 & -1/h & 1/h & ... & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & ... & ... & -1/h & 1/h \end{bmatrix},\ b_{jerk_{blk,i,w}} = \begin{bmatrix} j_{i,w}[2] \\ j_{i,w}[3] \\ \vdots \\ j_{i,w}[N] \end{bmatrix}$$

$$for\ i\epsilon\{x, y, z\}, w\epsilon\{max, min\}$$

(5.66)

The jerk inequality constraint block matrices can be combined to form the overall jerk inequality matrix, as in Eq. (5.67). Note that all steps should include a jerk constraint to prevent the resulting trajectory from requiring unrealistically large jerk values.

$$A_{jerk}\chi_{x,y,z} \leq b_{jerk}$$

where

$$A_{jerk} = \begin{bmatrix} -A_{jerk,blk} & 0 & 0 \\ 0 & -A_{jerk,blk} & 0 \\ 0 & 0 & -A_{jerk,blk} \\ A_{jerk,blk} & 0 & 0 \\ 0 & A_{jerk,blk} & 0 \\ 0 & 0 & A_{jerk,blk} \end{bmatrix}, \; b_{jerk} = \begin{bmatrix} -b_{jerk,blk,x,min} \\ -b_{jerk,blk,y,min} \\ -b_{jerk,blk,z,min} \\ b_{jerk,blk,x,max} \\ b_{jerk,blk,y,max} \\ b_{jerk,blk,z,max} \end{bmatrix} \tag{5.67}$$

Snap Constraints

As with the jerk, the snap at each step must be bounded (Eq. (5.68)). The snap limitations of a quadrotor are not typically known, so these bounds can be adjusted as needed to converge to a feasible trajectory. The snap calculation from Eq. (5.38) can be put in matrix form, as shown in Eq. (5.69).

$$s_{i,min}[n] \leq s_i[n] \leq s_{i,max}[n], \; \forall n, i \in \{x, y, z\} \tag{5.68}$$

$A_{snap}\chi_{x,y,z} \leq b_{snap}$

$where$

$$A_{snap,blk} = \begin{bmatrix} -1/h^2 & 0 & 0 & ... & 0 & 0 & 0 \\ -2/h^2 & -1/h^2 & 0 & ... & 0 & 0 & 0 \\ 1/h^2 & -2/h^2 & 1/h^2 & ... & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & ... & ...1/h^2 & -1/h & 1/h & \end{bmatrix}, \; b_{snap_{blk,i,w}} = \begin{bmatrix} s_{i,w}[1] \\ s_{i,w}[2] \\ \vdots \\ s_{i,w}[N] \end{bmatrix}$$

$for\ i\epsilon\{x,y,z\}, w\epsilon\{max,min\}$

(5.69)

The snap inequality constraint block matrices can be combined to form the overall snap inequality matrix, as in Eq. (5.70). Note that all steps should include a snap constraint to prevent the resulting trajectory from requiring unrealistically large snap values.

$$A_{snap}\chi_{x,y,z} \leq b_{snap}$$

where

$$A_{snap} = \begin{bmatrix} -A_{snap,blk} & 0 & 0 \\ 0 & -A_{snap,blk} & 0 \\ 0 & 0 & -A_{snap,blk} \\ A_{snap,blk} & 0 & 0 \\ 0 & A_{snap,blk} & 0 \\ 0 & 0 & A_{snap,blk} \end{bmatrix}, \; b_{snap} = \begin{bmatrix} -b_{snap,blk,x,min} \\ -b_{snap,blk,y,min} \\ -b_{snap,blk,z,min} \\ b_{snap,blk,x,max} \\ b_{snap,blk,y,max} \\ b_{snap,blk,z,max} \end{bmatrix} \tag{5.70}$$

Yaw Angle Constraints

The yaw angle at each step of the trajectory can be constrained to a range as shown in Eq. (5.71). This can be useful if the vehicle has an on-board camera or other sensor that needs to remain pointed in a certain direction throughout a maneuver (Fig. 5.6). If there is no restriction on desired yaw angle, these constraints can be omitted from the optimization problem.

$$p_{\psi,min}[n] \leq p_{\psi}[n] \leq p_{\psi,max}[n], \; \forall n \tag{5.71}$$

The yaw angle can be calculated with the vector dot product shown earlier in Eq. (5.53). This equation can be put into matrix form for all of the yaw angle inequality constraints (Eq. (5.72)).

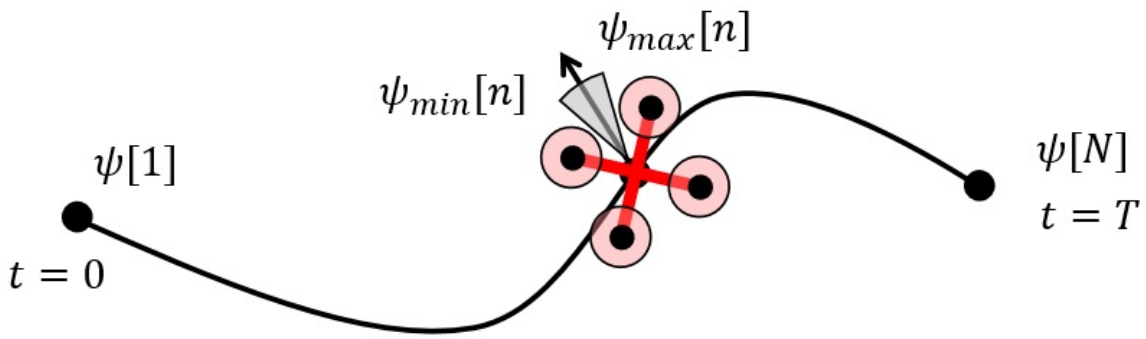


Figure 5.6: Example of yaw inequality constraints along trajectory.

$A_{pos,\psi}\chi_{\psi} \leq b_{pos,\psi}$

where

$$A_{pos,\psi,blk} = \begin{bmatrix} \frac{1}{2}h^2 & 0 & 0 & ... & 0 \\ \frac{3}{2}h^2 & \frac{1}{2}h^2 & 0 & ... & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \frac{(2k-3)}{2}h^2 & \frac{(2k-5)}{2}h^2 & ... & ... & \frac{1}{2}h^2 \end{bmatrix}, \; b_{pos_{blk,\psi,w}} = \begin{bmatrix} p_{\psi,w}[2] - p_{\psi}[1] - hv_{\psi}[1] \\ p_{\psi,w}[3] - p_{\psi}[1] - hv_{\psi}[1] \\ \vdots \\ p_{\psi,w}[N] - p_{\psi}[1] - hv_{\psi}[1] \end{bmatrix}$$

$for\ w\epsilon\{max, min\}$

(5.72)

These yaw angle inequality constraint block matrices can be combined to form the overall yaw angle inequality matrix (Eq. (5.73)).

$$
\begin{aligned}
&A_{pos,\psi}\chi_{\psi} \leq b_{pos,\psi} \\
&where \\
&A_{pos,\psi} = \begin{bmatrix} -A_{pos,\psi,blk} \\ A_{pos,\psi,blk} \end{bmatrix},\ b_{pos} = \begin{bmatrix} -b_{pos,blk,\psi,min} \\ b_{pos,blk,\psi,max} \end{bmatrix}
\end{aligned}
\tag{5.73}
$$

Yaw Angular Velocity Constraints

The yaw angular velocity at each step of the trajectory can be constrained to a range as shown in Eq. (5.74). This can be used to include the vehicle's yaw rate limitations into the trajectory to ensure the generated trajectory is physically realistic for the vehicle.

$$
v_{\psi,min}[n] \leq v_{\psi}[n] \leq v_{\psi,max}[n],\ \forall n \tag{5.74}
$$

The yaw angular velocity can be calculated with the vector dot product shown earlier in Eq. (5.53). This equation can be put into matrix form for all of the yaw angular velocity inequality constraints (Eq. (5.75)).

$$A_{vel,\psi}\chi_{\psi} \leq b_{vel,\psi}$$

$$where$$

$$A_{vel,\psi,blk} = \begin{bmatrix} h & 0 & 0 & ... & 0 & 0 \\ h & h & 0 & ... & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ h & h & ... & ... & h & 0 \end{bmatrix},\ b_{vel_{blk,\psi,w}} = \begin{bmatrix} v_{\psi,w}[2] - v_{\psi}[1] \\ v_{\psi,w}[3] - v_{\psi}[1] \\ \vdots \\ v_{\psi,w}[N] - v_{\psi}[1] \end{bmatrix} \tag{5.75}$$

$$for\ w \epsilon \{max, min\}$$

These yaw angular velocity inequality constraint block matrices can be combined to form the overall yaw angular velocity inequality matrix (Eq. (5.76)). Note that all steps of the trajectory should include an angular velocity constraint to prevent unrealistically large yaw rates in the resulting trajectory.

$$\begin{aligned} &A_{vel,\psi}\chi_{\psi} \leq b_{vel,\psi} \\ &where \\ &A_{vel,\psi} = \begin{bmatrix} -A_{vel,\psi,blk} \\ A_{vel,\psi,blk} \end{bmatrix},\ b_{vel} = \begin{bmatrix} -b_{vel,blk,\psi,min} \\ b_{vel,blk,\psi,max} \end{bmatrix} \end{aligned} \tag{5.76}$$

Yaw Angular Acceleration Constraints

The yaw angular acceleration at each step of the trajectory can be constrained to a range as shown in Eq. (5.77). As with the angular velocity constraints, this can be used to include the vehicle's yaw angular acceleration limitations into the

trajectory to ensure the generated trajectory is physically realistic for the vehicle. Since the optimization variable consists of the yaw angular acceleration components at each step, the inequality block matrix is simply an identity matrix and the inequality vector is the minimum and maximum allowable angular acceleration (Eq. (5.78)).

$$a_{\psi,min}[n] \leq a_{\psi}[n] \leq a_{\psi,max}[n], \ \forall n \tag{5.77}$$

$$\begin{aligned} & A_{accel,\psi}\chi_{\psi} \leq b_{accel,\psi} \\ & where\ b_{accel_{blk,\psi,w}} = \begin{bmatrix} a_{\psi,w}[1] \\ a_{\psi,w}[2] \\ \vdots \\ a_{\psi,w}[N] \end{bmatrix} \\ & for\ w\epsilon\{max, min\} \end{aligned} \tag{5.78}$$

These yaw angular acceleration inequality constraint block matrices can be combined to form the overall yaw angular acceleration inequality matrix (Eq. (5.79)). Note that all steps of the trajectory should include an angular acceleration constraint to prevent unrealistically large angular accelerations in the resulting trajectory.

$$
\begin{aligned}
&A_{accel,\psi}\chi_{\psi} \leq b_{accel,\psi} \\
&where \\
&A_{accel,\psi} = \begin{bmatrix} -\mathsf{I} \\ \mathsf{I} \end{bmatrix},\ b_{accel,\psi} = \begin{bmatrix} -b_{accel,blk,\psi,min} \\ b_{accel,blk,\psi,max} \end{bmatrix}
\end{aligned}
\tag{5.79}
$$

Final Optimization Program

With the objective function and constraint equations defined, the final optimization problem can now be written. As stated previously, the optimization problem is of the form of a quadratic program (QP). The objective function is stated in Eq. (5.80). This function includes the optimization variables $\chi_{x,y,z}$ and χ_{ψ} joined into a single optimization variable χ. The matrix functional Q for the entire QP includes the corresponding position and yaw angle matrices.

$$
f_o = \begin{bmatrix} \chi_{x,y,z} \\ \chi_{\psi} \end{bmatrix}^T \begin{bmatrix} P_{x,y,z} & 0 \\ 0 & P_{\psi} \end{bmatrix} \begin{bmatrix} \chi_{x,y,z} \\ \chi_{\psi} \end{bmatrix} = \chi^T Q \chi \tag{5.80}
$$

The constraint matrix equations can be combined into single equality and inequality equations in terms of the optimization variable χ. The final equality constraint equation is shown in Eq. (5.81) and the final inequality constraint equation is shown in Eq. (5.82).

$$
\begin{aligned}
&A_{eq}\chi = b_{eq} \\
&where \\
&A_{eq} = \begin{bmatrix} A_{eq,x,y,z} & 0 \\ 0 & A_{eq,\psi} \end{bmatrix},\ b_{eq} = \begin{bmatrix} b_{eq,x,y,z} \\ b_{eq,\psi} \end{bmatrix}
\end{aligned}
\tag{5.81}
$$

$$
\begin{aligned}
&A_{ineq}\chi \leq b_{ineq} \\
&where \\
&A_{ineq} = \begin{bmatrix} A_{pos} & 0 \\ 0 & A_{pos,\psi} \\ A_{vel} & 0 \\ 0 & A_{vel,\psi} \\ A_{accel} & 0 \\ 0 & A_{accel,\psi} \\ A_{jerk} & 0 \\ 0 & A_{snap} \end{bmatrix},\ b_{ineq} = \begin{bmatrix} b_{pos} \\ b_{pos,\psi} \\ b_{vel} \\ b_{vel,\psi} \\ b_{accel} \\ b_{accel,\psi} \\ b_{jerk} \\ b_{snap} \end{bmatrix}
\end{aligned}
\tag{5.82}
$$

The full optimization problem is written in Eq. (5.83). This optimization problem satisfies the requirements of a quadratic program since the objective function is quadratic with Q being a symmetric positive semi-definite matrix and the constraint functions are affine. A quadratic program is also a convex optimization problem with a single optimal solution. Many methods exist for solving quadratic programs and solutions can be obtained quickly using Matlab's Optimization Toolbox [25] or Stephen Boyd's CVX toolbox [26].

$$\begin{aligned} \text{minimize} \quad & \chi^T Q \chi \\ \text{subjectto} \quad & A_{eq}\chi = b_{eq} \\ & A_{ineq}\chi \leq b_{ineq} \end{aligned} \tag{5.83}$$

5.2.2 Example Trajectory

To demonstrate the process of generating an optimal trajectory for a quadrotor, an example situation is presented for a 10-second trajectory with various constraints. The vehicle is to begin at an arbitrary origin. At the 2 second mark, the vehicle will enter a tunnel with width and height constraints at position $(2, -1, -2)$ and reach the end of the tunnel 2 seconds later at point $(4, -1, -2)$. Note that these coordinates are in the inertial North-East-Down coordinate system (NED). At the 5 second mark, the vehicle should come to a stop for an instant at point $(5, -1.5, -2)$ with a yaw angle of $-90\,\text{deg}$. The vehicle will then continue and stop for an instant at point $(7, -6, -3)$ with a yaw angle of $+90\,\text{deg}$. The vehicle ends the trajectory at point $(10, -5, 0)$ with a yaw angle back to $0\,\text{deg}$. This example trajectory incorporates several position constraints as well as attitude and yaw constraints. The generated trajectory is shown below in Fig. 5.7.

The 4 flat outputs and their derivatives are shown in Fig. 5.8 through 5.10. The tunnel inequality constraints and the several equality constraints are included in the plots to show that the trajectory generation algorithm was successful in abiding by the user-imposed environmental restrictions.

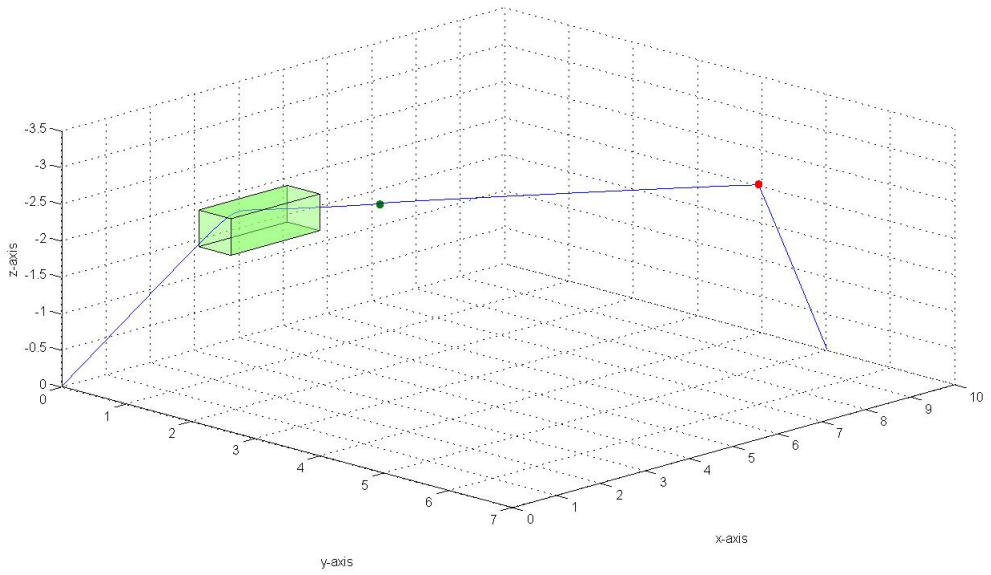


Figure 5.7: 3D view of example optimal trajectory.

The corresponding quaternion attitude at each time step can be calculated and then converted to Euler angles for easier visualization. The Euler angle time series is shown in Fig. 5.11.

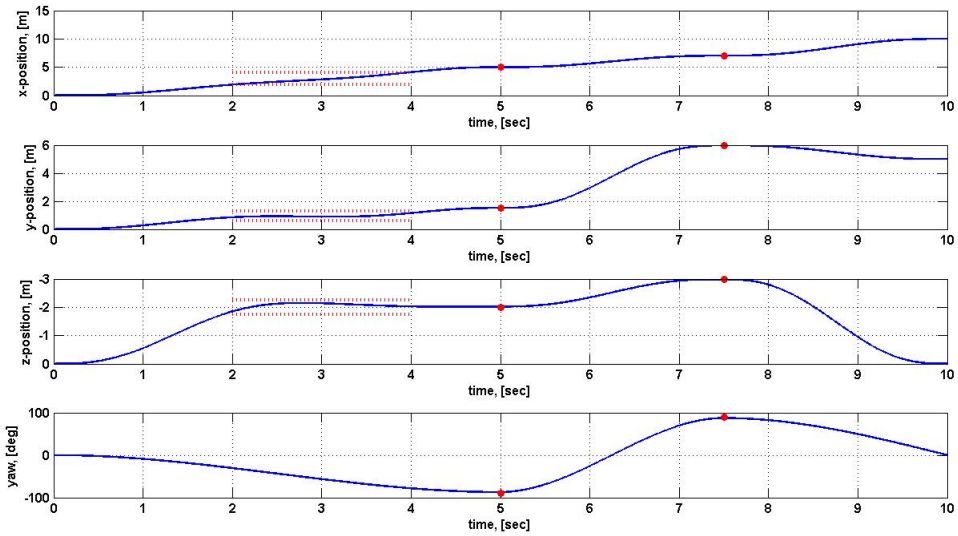


Figure 5.8: Flat outputs that define example optimal trajectory.

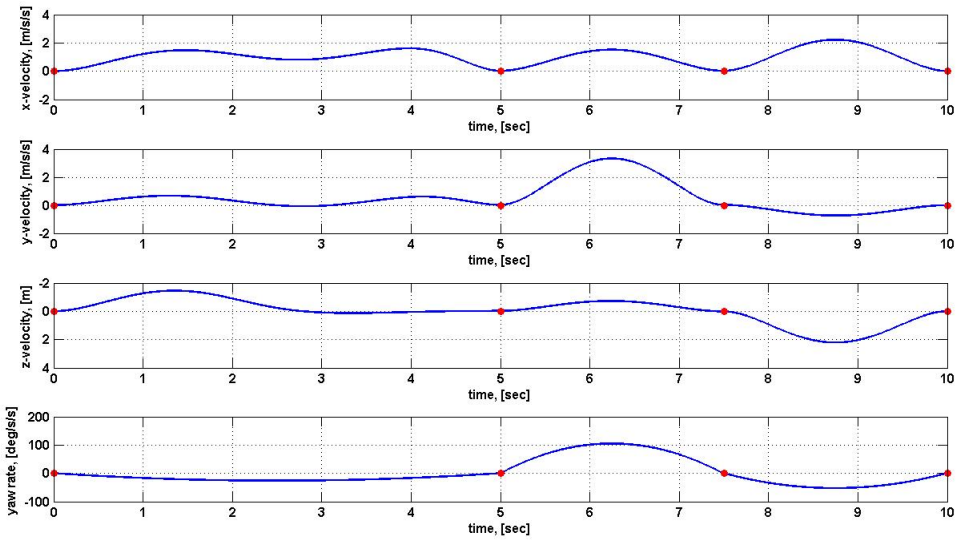


Figure 5.9: First derivatives of flat outputs that define example optimal trajectory.

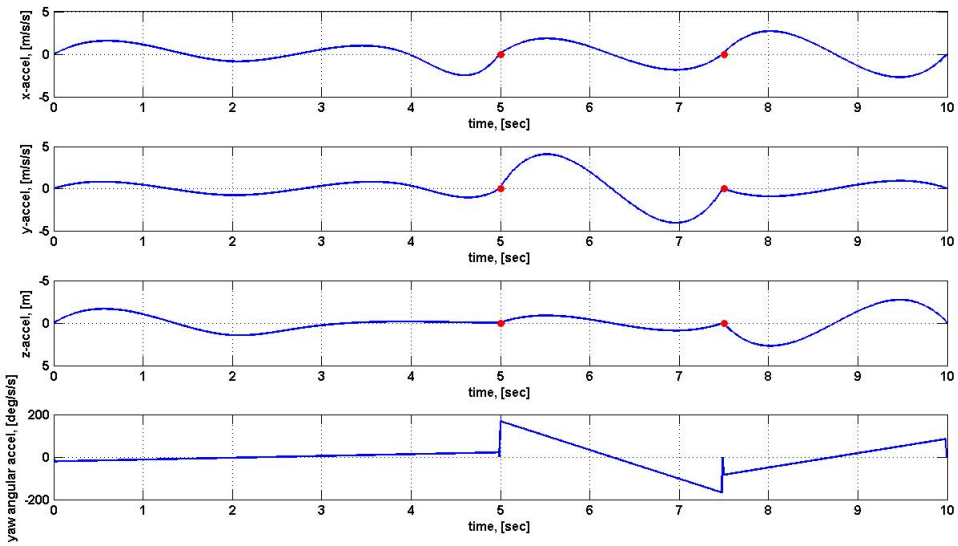


Figure 5.10: Second derivatives of flat outputs that define example optimal trajectory.

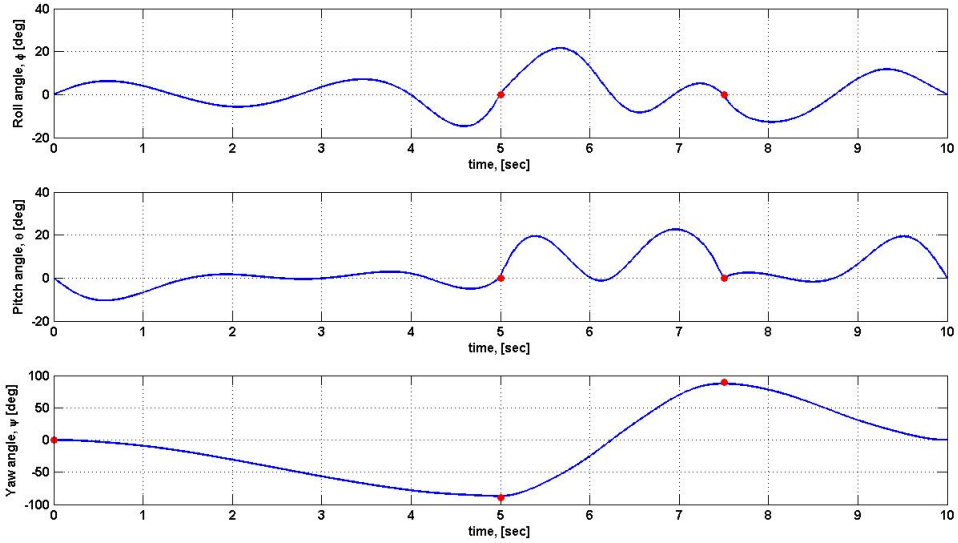


Figure 5.11: Euler angle attitudes along example trajectory.

Chapter 6: Trajectory Tracking Control

To enable the ARL MkIV to follow trajectories generated with the algorithm described in the previous chapter, an outer-loop trajectory tracking controller must be implemented. This controller acts on the slower modes of a quadrotor, which are the 3-dimensional inertial position and the heading, and therefore runs in an outer loop around the attitude controller. The tracking controller maps errors in the position and heading to inputs to the inner-loop attitude controller, taking the place of a human input device such as a joystick. This chapter defines two trajectory tracking controllers. The first is a PID position controller common to quadrotors and the second is an extension to the first, taking advantage of the quaternion attitude representation of the attitude controller.

6.1 PID Trajectory Tracking Controller

Proportional-integral-derivative (PID) feedback controllers are very common for the outer-loop control of quadrotors due to their simplicity of implementation and the fact that they don't require an accurate model of the system in order to work properly. PID controllers can be manually tuned based on the resulting performance of the controlled system, where as more complex controllers such as

LQR or H-infinity require system models.

An outer-loop PID position controller calculates the control input to the inner-loop by summing the weighted errors in position and velocity, as well as the integrated error in position and then mapping the sum to a desired attitude and throttle level. The current inertial position $\bar{r}$ of the quadrotor's center of mass is defined in Eq. (6.1) and the desired inertial position $\bar{r}_d es$ is defined in Eq. (6.2). Note that the time dependence has been omitted.

$$\bar{r} = [x,\, y,\, z]^T \tag{6.1}$$

$$\bar{r}_{des} = [x_{des},\, y_{des},\, z_{des}]^T \tag{6.2}$$

The position and velocity errors are calculated by subtracting the desired value from the current value (Eq. (6.3)). The integral of the position error is calculated numerically as shown in Eq. (6.4). The errors in yaw angle, yaw angular velocity, and integrated yaw angle are calcualted in the same manner as shown in Eq. (6.5) and Eq. (6.6).

$$\begin{aligned} \bar{e}_p &= \bar{r} - \bar{r}_{des} \\ \bar{e}_d &= \dot{\bar{r}} - \dot{\bar{r}}_{des} \end{aligned} \tag{6.3}$$

$$\bar{e}_i = \int_0^t \bar{e}_p(\tau) d\tau \rightarrow \bar{e}_i[k] = \bar{e}_i[k-1] + \bar{e}_p[k]\Delta t \tag{6.4}$$

$$\begin{aligned} \bar{e}_{p,\psi} &= \psi - \psi_{des} \\ \bar{e}_{d,\psi} &= \dot{\psi} - \dot{\psi}_{des} \end{aligned} \tag{6.5}$$

$$\bar{e}_{i,\psi} = \int_0^t \bar{e}_{p,\psi}(\tau)d\tau \rightarrow \bar{e}_{i,\psi}[k] = \bar{e}_{i,\psi}[k-1] + \bar{e}_{p,\psi}[k]\Delta t \tag{6.6}$$

The control inputs $\bar{u}_{PID}$ and $u_{PID,\psi}$ are calculated with a weighted sum of the errors (Eq. (6.7)), where K_p, K_i, and K_d are the proportional, integral, and derivative diagonal gain matrices, respectively. The inputs must then be mapped to a desired quaternion attitude for use by the inner-loop attitude controller. Since the roll and pitch axes are decoupled along the body y and x axes, the x and y components of $\bar{u}_{PID}$ can be realized as pitch and roll Euler angles. The heading control input $u_{PID,\psi}$ can be realized as the yaw Euler angle. These Euler angles can be converted to the corresponding desired quaternion attitude using Eq. (6.8). This desired quaternion is then sent to the inner-loop attitude controller that sends the necessary commands to the motors that enable the vehicle to follow the desired trajectory.

$$\begin{aligned} \bar{u}_{PID} &= K_p\bar{e}_p + K_i\bar{e}_i + K_d\bar{e}_d \\ u_{PID,\psi} &= K_{p,\psi}\bar{e}_{p,\psi} + K_{i,\psi}\bar{e}_{i,\psi} + K_{d,\psi}\bar{e}_{d,\psi} \end{aligned} \tag{6.7}$$

$$\begin{bmatrix} q_{0,des} \\ q_{1,des} \\ q_{2,des} \\ q_{3,des} \end{bmatrix} = \begin{bmatrix} \cos\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \\ \cos\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} - \sin\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} \\ \cos\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} \\ \sin\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} - \cos\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \end{bmatrix} \tag{6.8}$$

The z-axis of the control input $\bar{u}_{PID}$ is equivalent to the deviation from the hover throttle level. Defining this term as a deviation from hover is equivalent to initializing the integrated z-axis position error to the hover throttle. This gets rid of the wait time associated with the integrator wind-up when the outer-loop controller is switched on while the vehicle is already in flight.

Due to the mapping from control input to Euler angles, this outer-loop controller is subject to the limitations that go along with the use of Euler angles. It is only useful for less aggressive maneuvers that require small pitch and roll angles. In order to take advantage of the quaternion-based state estimator implemented on the ARL MkIV, a direct quaternion mapping must be used in the outer-loop, removing Euler angles completely from the formulation. The controller that accomplishes this is discussed in the next section.

6.1.1 Circular Test Trajectory

The PID trajectory tracking controller was used to fly consecutive circular trajectories as an operational example. The circles are 2 meters in diameter and the vehicle completes each circle in 3 seconds. Typical results for such a trajectory are shown in Fig. 6.1. The vehicle deviates from the trajectory by up to 20 centimeters

at some points due to the high speed requirement, as well as hardware factors that will be discussed later. Overall, however, the quadrotor is able to remain stable while traversing the circle with speeds up to 2.1 meters per second.

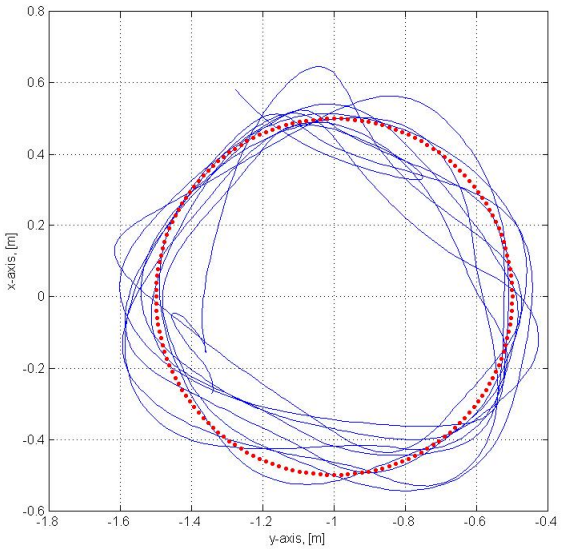


Figure 6.1: PID tracking controller used to fly circular trajectory.

6.2 Acceleration Vector-Based Trajectory Tracking Controller

In order to leverage the advantages of using quaternions as the sole method of attitude representation throughout the quadrotor control structure, the outer-loop controller can be reformulated to output a quaternion orientation directly, without the intermediate Euler conversion. The inner-loop attitude controller accepts a quaternion attitude as an input from the outer-loop controller. It was shown in a previous chapter that the attitude of a quadrotor can be entirely described by its acceleration vector and a yaw angle. Therefore, if the outer-loop controller

maps inertial state errors to a pseudo-acceleration vector, this vector can be directly converted to a quaternion orientation and sent to the inner-loop.

The state error terms used in the PID trajectory tracking controller will also be used here (Eq. (6.3)-(6.6)). The same PID structure will also be used, except here it will be mapped to an acceleration vector and not an Euler angle. The pseudo-acceleration vector ${}^{I}\bar{a}$ is calculated with Eq. (6.9). The desired acceleration vector from the trajectory generation is included in the calculation, as well as the gravity vector in the inertial frame. The total commanded force in the inertial frame can then be found with Eq. (6.10), where m is the mass of the vehicle.

$$ {}^{I}\bar{a} = K_p\bar{e}_p + K_d\bar{e}_d + \ddot{x}_{des} + K_i\bar{e}_i + {}^{I}\bar{g} \tag{6.9} $$

$$ \bar{F}^{I} = m\,{}^{I}\bar{a} \tag{6.10} $$

The heading control portion of the outer-loop is the same as that of the original PID controller. The equation for computing the yaw control input ψ_c is restated in Eq. (6.11).

$$ \psi_c = K_{p,\psi}\bar{e}_{p,\psi} + K_{i,\psi}\bar{e}_{i,\psi} + K_{d,\psi}\bar{e}_{d,\psi} \tag{6.11} $$

The pseudo-acceleration vector and yaw control input can be converted to a desired quaternion orientation using the equations derived in the earlier section on differential flatness. The normalized pseudo-acceleration vector ${}^{I}\hat{a}$ is first used to compute the corresponding quaternion attitude with arbitrary yaw rotation $\tilde{q}_{des}$

(Eq. (6.12)). The resulting quaternion represents the rotation necessary to make the body-frame acceleration vector ${}^{B}\hat{a}$ collinear with the pseudo-acceleration vector without considering the yaw angle.

$$\tilde{q}_{des} = \frac{1}{\sqrt{2\left(1 + {}^{B}\hat{a}^{TI}\hat{a}\right)}} \begin{bmatrix} 1 + {}^{B}\hat{a}^{TI}\hat{a} \\ {}^{B}\hat{a} \times {}^{I}\hat{a} \end{bmatrix} \tag{6.12}$$

The commanded yaw angle can be included in the quaternion orientation with Eq. (6.13). The resulting desired quaternion $\bar{q}_{des}$ is the quadrotor orientation that will allow the vehicle to follow the desired trajectory. This quaternion is sent to the inner-loop attitude controller.

$$\bar{q}_{des} = \tilde{q}_{des} \otimes \begin{bmatrix} \cos\left(\frac{\psi_c}{2}\right) \\ 0 \\ 0 \\ \sin\left(\frac{\psi_c}{2}\right) \end{bmatrix} \tag{6.13}$$

The magnitude of the commanded force in the inertial frame (Eq. (6.10)) is used to set the throttle command. The relationship between throttle level and applied force from the rotors was characterized, as described in the following section, and a piece-wise linear curve-fit was found. With the throttle level and desired quaternion calculations complete, the outer-loop controller enables the quadrotor to follow a desired trajectory without the limitations of an Euler angle attitude representation.

6.2.1 Motor Thrust Characterization

The acceleration vector-based controller requires a model of the mapping between throttle level and overall thrust produced by the four rotors. A thrust test was performed by mounting the ARL quadrotor to a load cell and collecting average thrust data for every throttle level in 0.05% increments. The results are shown in Fig. 6.2. [27]

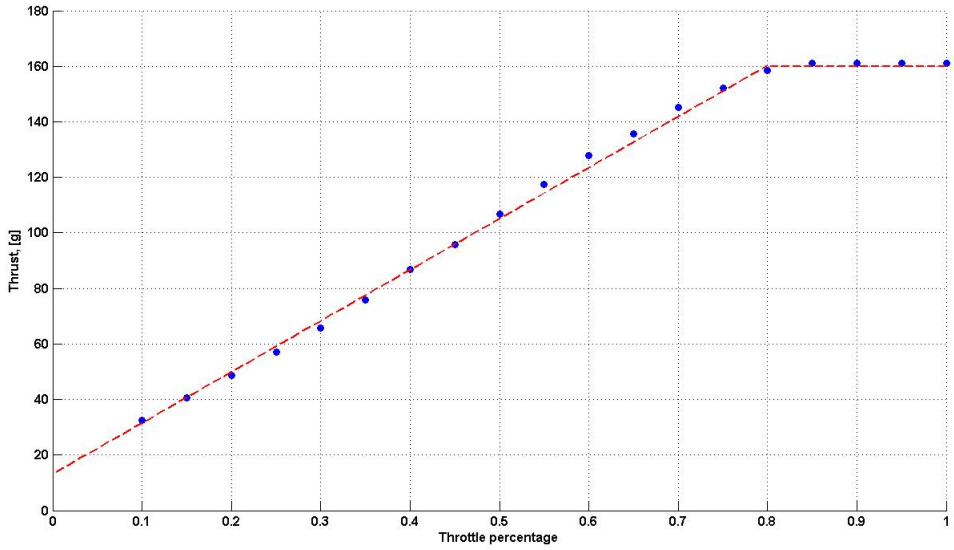


Figure 6.2: Vehicle thrust versus throttle level with best-fit linear trend line.

The controller requires a mapping equation to determine an estimate of the throttle level required to achieve a desired thrust. Therefore, the axes were flipped and a line was fit to the data (Fig. 6.3). The maximum thrust was found to be about 160 grams at a throttle level of 80%. The minimum thrust was found to be about 30 grams at a throttle level of 10%. The equation for the best-fit trend line

between 10% and 80% throttle is shown in Eq. (6.14)

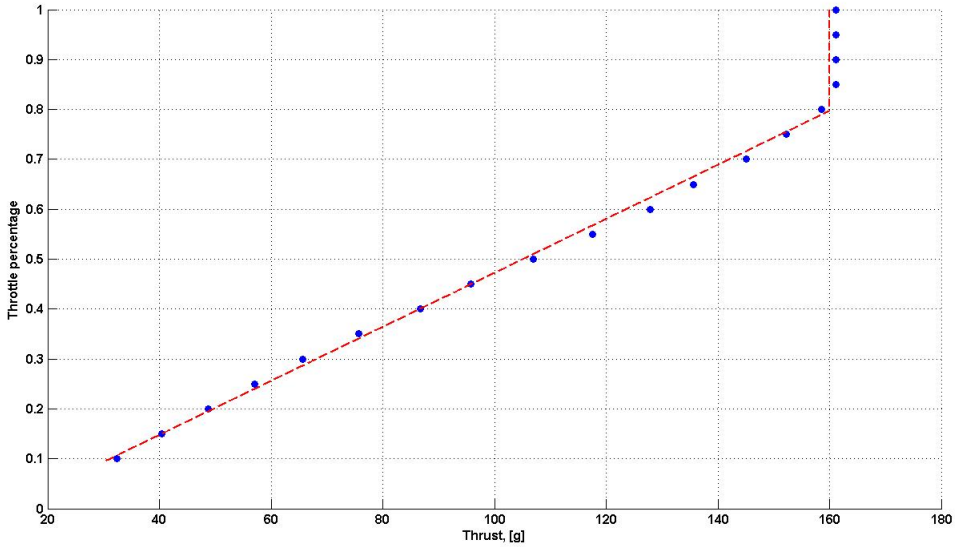


Figure 6.3: Throttle level versus vehicle thrust with best-fit linear trend line.

$$Throttle\% = 0.0054119 * Thrust - 0.068689 \tag{6.14}$$

6.2.2 Circular Test Trajectory

As was done for the PID tracking controller, the acceleration vector-based trajectory tracking controller was used to fly consecutive circular trajectories as an operational example. Again, the circles are 2 meters in diameter and the vehicle completes each circle in 3 seconds. Typical results for such a trajectory are shown in Fig. 6.4. The vehicle follows the circular trajectory slightly better than the PID controller, but still deviates by up to 20 centimeters as before. This controller, however, allows for even more aggressive maneuvers due to the full quaternion attitude

representation.

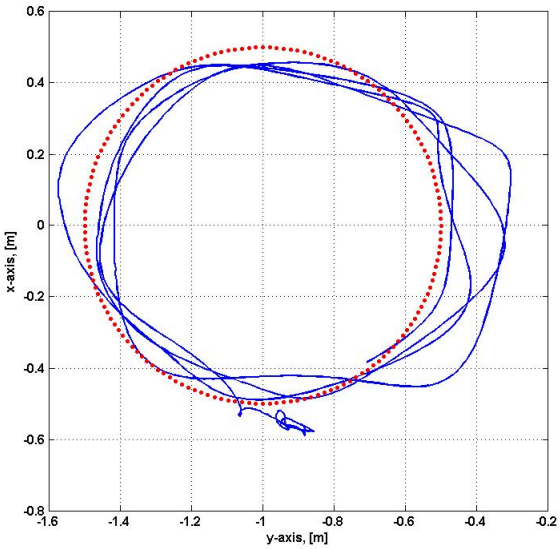


Figure 6.4: Accel vector-based tracking controller used to fly circular trajectory.

Chapter 7: Trajectory Generation for Perching Application

One potential use for the trajectory generation algorithm is to enable a quadrotor to perform an aggressive perching maneuver that ends with the vehicle at the appropriate position and orientation. This application is motivated by a collaborative research effort with Stanford University's Biomimetics and Dextrous Manipulation Laboratory (BDML) funded by the Army's Micro Autonomous Systems and Technology (MAST) program. The ability to perch on elevated or vertical surfaces greatly expands the potential uses of a quadrotor. Perching can extend the operational life of the vehicle and allow tasks to be performed without the mechanical and electrical noise associated with sustained flight. This section will describe the BDML perching mechanism, the process of generating a perching trajectory, and the simulations and flight tests of the perching maneuver.

7.1 BDML's Dry-Adhesive Perching Mechanism

The BDML of Stanford University has developed light-weight perching mechanisms for use on small-scale air vehicles, as well as crawling-based ground vehicles [28], [29]. These mechanisms involve the use of bio-inspired dry adhesive technology. This type of adhesive mimics the foot pads of a gecko, relying solely on the

inter-molecular van der Waals forces to adhere to many smooth surfaces. The dry adhesive mechanism also has the ability to attach and release from a surface with minimal force, due to the directional characteristics of the adhesive.

The perching mechanism used on the ARL MkIV quadrotor weighs 9.6 grams and attaches to the top or bottom of the vehicle. The mechanism absorbs excess energy from the initial contact with the surface to prevent the vehicle from bouncing back before the adhesive is engaged. Once both dry adhesive pads are in contact with the surface, the mechanism collapses and applies opposing shear forces to each pad, securing it to the surface. Releasing the applied shear force instantly releases the vehicle, allowing it to return to flight. The perching mechanism will adhere to most smooth surfaces, such as glass, plastic, and metal. It is sensitive to dirt and other foreign particles, however, since even a small particle between the adhesive pad and the surface will result in drastically less surface contact. One version of the BDML perching mechanism is shown in Fig. 7.1, mounted on a previous micro-quadrotor iteration and perched on a Plexiglas surface.

7.2 Perching Trajectory Generation

Perching trajectories were generated with the algorithm described previously. The starting constraints were chosen so that the quadrotor begins in hover 1 meter in front of and 1 meter below the perching target. The ending constraints were chosen so that the final attitude of the vehicle is such that the x-y plane of the body frame is parallel to the perching surface and the vehicle has a normal velocity

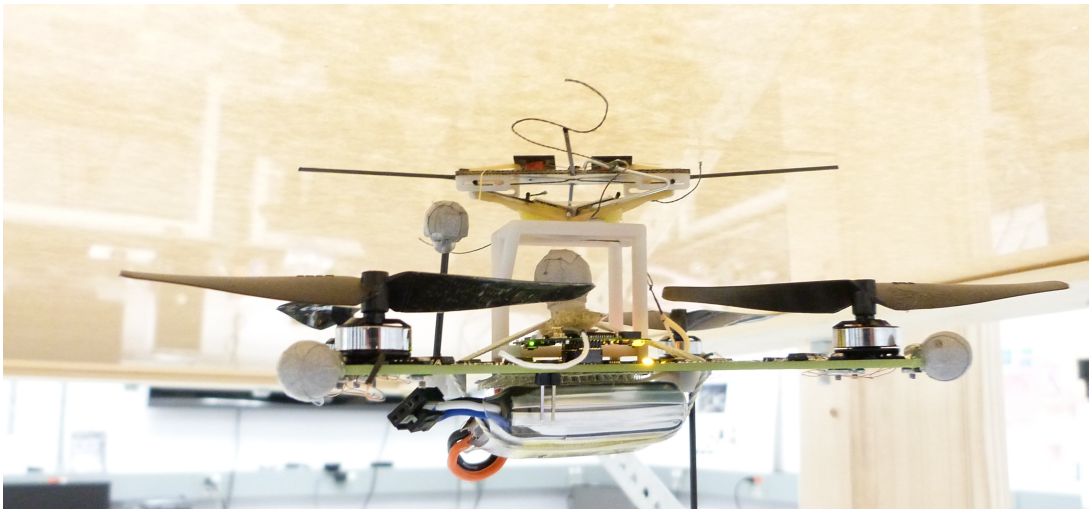


Figure 7.1: Old version of micro-quadrotor with the BDML perching mechanism, perched on Plexiglas.

between 0.5 and 1.5 meters per second with zero tangential velocity. The total time for the maneuver is set to 2.5 seconds with a time step of 0.01 seconds.

A 3D view of the resulting perching trajectory is shown in Fig. 7.2. The vehicle starts at the origin and ends at the red dot, indicating the perching target on the wall. The vectors plotted along the 3D trajectory indicated the acceleration direction with gravity included. The quadrotor's attitude is such that the x-y plane of the quadrotor is perpendicular to these acceleration vectors. The positions along each axis during the trajectory are plotted in Fig. 7.3. Note that the ending points are consistent with the constraints previously described.

The velocities and accelerations are plotted in Fig. 7.4 and Fig. 7.5, respectively. Note that the ending velocity along the x-axis is within the 0.5 to 1.5 meter per second constraint for normal perching velocity. Also, the tangential velocity is

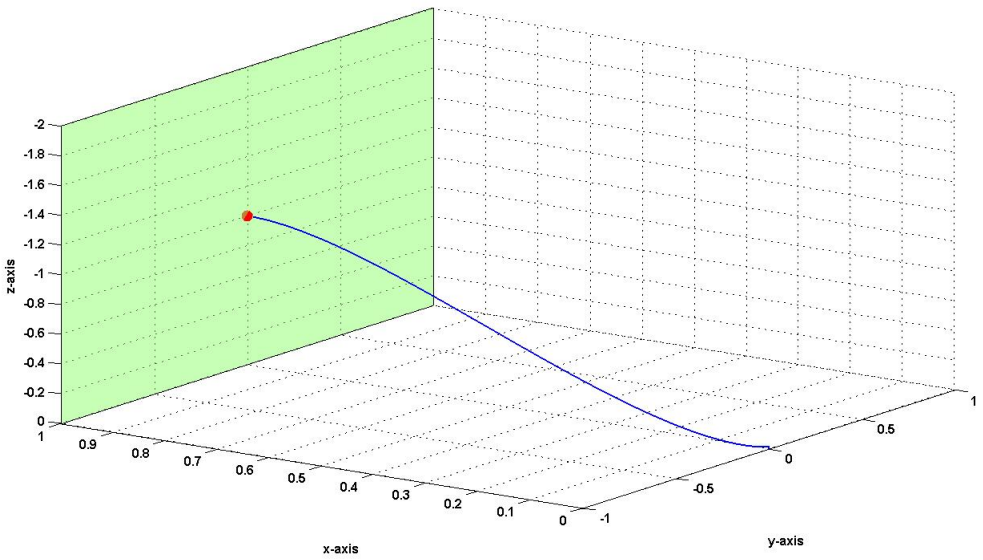


Figure 7.2: 3D view of perching trajectory.

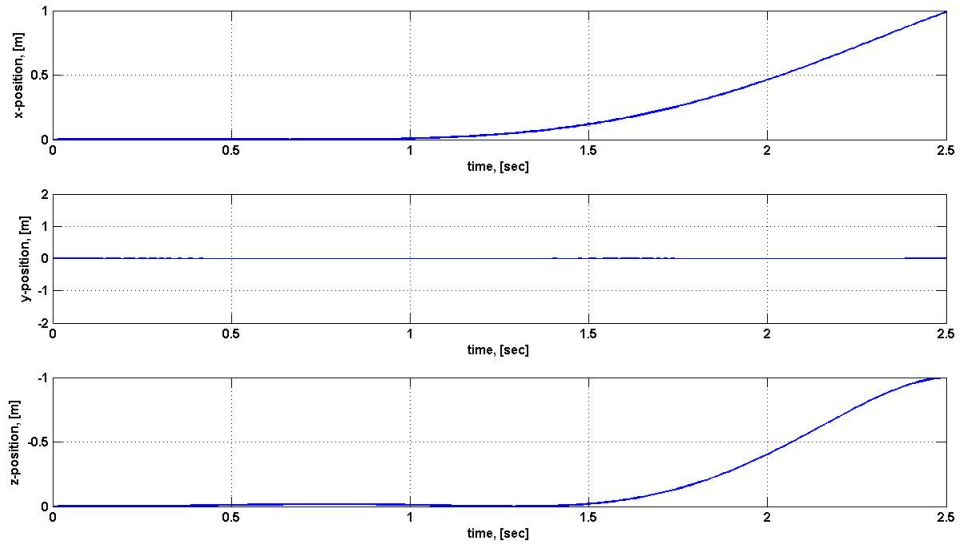


Figure 7.3: Positions along perching trajectory.

zero at the end of the trajectory. The ending acceleration is -1 meters per squared second in the x-direction and -9.81 meters per squared second in the z-direction.

When gravity is included, this results in a pitch-up perching attitude, as desired.

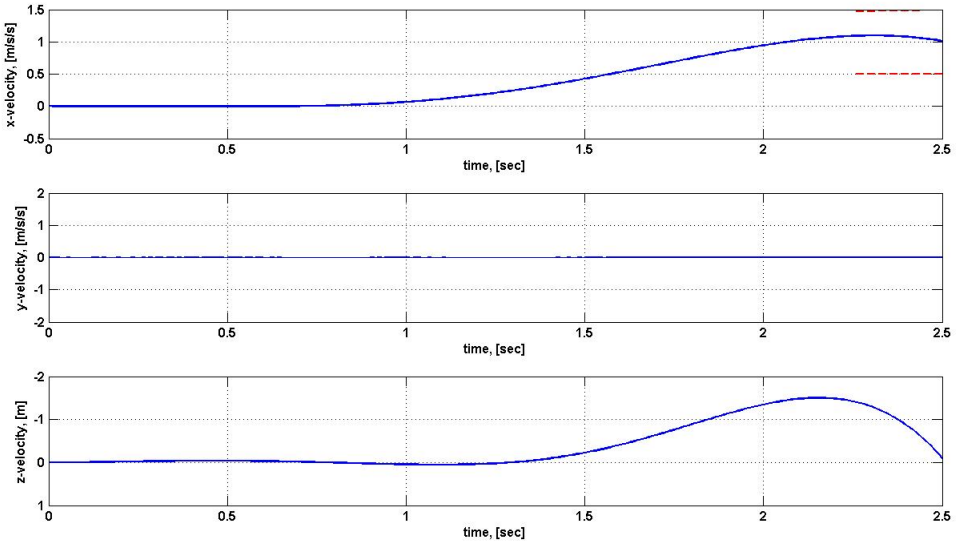


Figure 7.4: Velocities along perching trajectory.

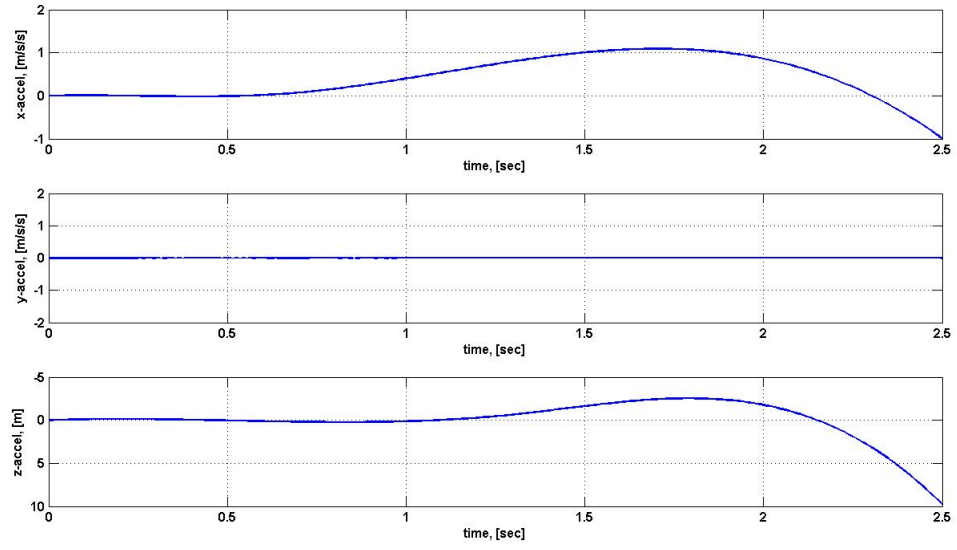


Figure 7.5: Accelerations along perching trajectory.

The quaternion attitudes at each point along the trajectory are also calculated and plotted as Euler angles for easier visualization in Fig. 7.6. Note that the pitch angle begins at zero and goes slightly negative during the forward acceleration portion of the maneuver. It then goes to positive 90 degrees at the end of the trajectory, indicating a wall-perch attitude. The yaw angle was kept at zero for the duration of the trajectory. Also, since there is no lateral component of the trajectory, the roll angle is zero.

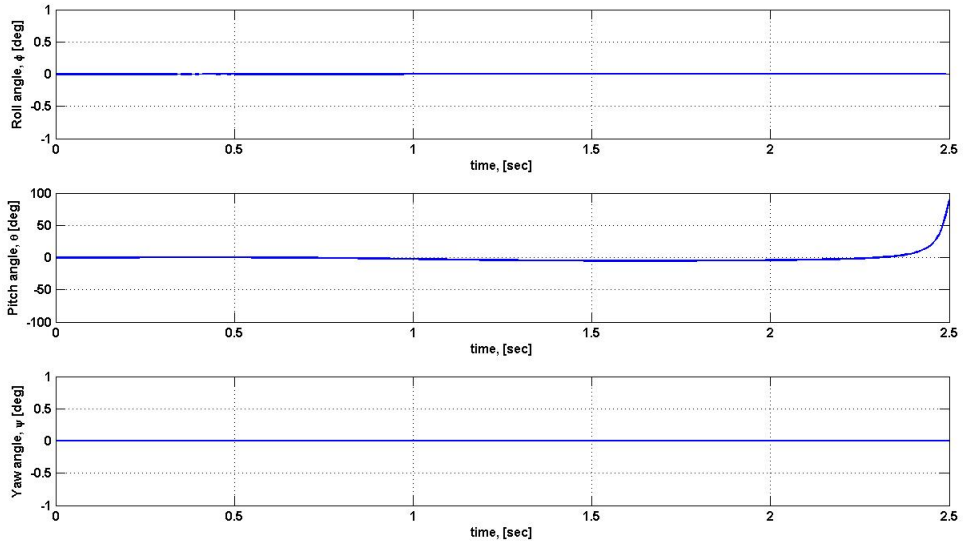


Figure 7.6: Euler angle attitude along perching trajectory.

7.3 Perching Test Flights

The acceleration vector-based trajectory tracking controller was used along with the vertical wall perching trajectory generated with the previously described algorithm to perform perching tests with the ARL MkIV micro-quadrotor. The tests

were completed using the Vicon motion capture system for inertial data feedback. A Plexiglas wall was constructed and served as the vertical perching surface.

After initial testing with the ARL MkIV vehicle, it was determined that inconsistencies in the motor speed controllers along with other vehicle limitations prevented successful perching with the current algorithm and trajectory. The vehicle was unable to track the changing acceleration vector closely enough to result in a proper attitude at the perching surface. To assist the vehicle in reaching the proper states, the trajectory was adjusted based on the state error of the previous test. This process is essentially that of an iterative learning controller, in which the controls for the next test are adjusted based on the previous test's state errors. An example of a successful perch using this method is shown in Fig. 7.7.

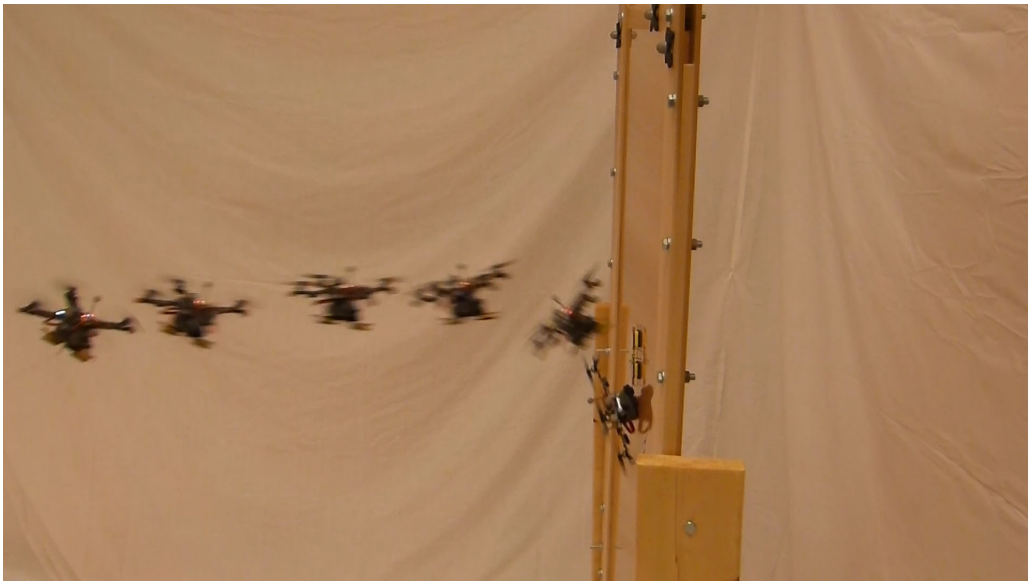


Figure 7.7: Image sequence from vertical wall perching test.

Chapter 8: Conclusions and Future Work

The work in this thesis demonstrates the utility of expanding the typical Euler angle paradigm to the higher-dimensional quaternion method of attitude representation. A trajectory generation algorithm was presented that creates minimum-control effort trajectories that enable a quadrotor to reach desired velocities and attitudes at certain positions. Outer loop controllers were also presented that allow a quadrotor to follow these trajectories and perform aggressive maneuvers without special numerical handling of extreme attitudes. These algorithms were implemented on a 100-gram micro-quadrotor and their capabilities were demonstrated with an example situation involving perching on a vertical glass wall. A specially designed gecko-inspired dry adhesive was used to attach to the glass wall once the vehicle reached the proper attitude and speed. An example trajectory was also generated that required quick flight through a tunnel, as well as fixed yaw angles and velocities at certain positions, simulating a potential application for such algorithms.

Vehicle limitations and frequent electronics issues limited the MkIV quadrotor's ability to fully demonstrate the potential of these algorithms. Nonetheless, the framework has been presented that gives any quadrotor UAV the ability to perform aggressive maneuvers during standard free-flight, as well as with pre-planned tra-

jectories. Expanding the current control structure to a nonlinear controller would further improve the quadrotor's ability to closely follow the fast-changing states of an aggressive trajectory.

The next step for such research, as is the case for many quadrotor-related endeavors, is to enable the vehicle to better sense its environment to the point that a Vicon system is unnecessary. With an accurate measurement of its position relative to a window, for example, a quadrotor could perform an agile perching maneuver to attach itself to the glass. Once perched, the vehicle could perform a number of static tasks, such as surveillance or charging, without the power requirements of remaining in flight. A quadrotor could also leverage its own dynamic properties, as was shown in the trajectory generation section, to navigate quickly around obstacles or pass through small openings that require a certain attitude in order to fit. A GPS alone is not enough to enable these types of applications. Work must continue in the area of advanced environment perception, using methods such as optical flow or simultaneous localization and mapping (SLAM), to expand the realm of possible uses for quadrotors.

Bibliography

[1] S. Bouabdallah. Design and Control of Quadrotors with Application to Autonomous Flying. *PhD thesis, Ecole Polytechnique Federale de Lausanne*, 2007.

[2] A. Kushleyev, D. Mellinger, and V. Kumar. Towards a swarm of agile micro quadrotors. *Proc. of Robotics: Science and Systems*, 2012.

[3] T. Lee, M. Leok, and N. McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). *Proc. of the IEEE Conf. on Decision and Control*, 2010.

[4] S. Shen, N. Michael, and V. Kumar. 3D estimation and control for autonomous flight with constrained computation. *Proc. of the IEEE Int. Conf. on Robotics and Automation*, May 2011.

[5] J.H. Gillula, H. Huang, M.P. Vitus, and C.J. Tomlin. Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice. *Proc. of the IEEE Int. Conf. on Robotics and Automation*, May 2010.

[6] G. Hoffmann, S. Waslander, and C. Tomlin. Quadrotor helicopter trajectory tracking control. *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

[7] Daniel Mellinger, Alex Kushleyev, and Vijay Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. *2012 IEEE International Conference on Robotics and Automation*, pages 477–483, May 2012.

[8] M.W. Mueller, M. Hehn, and R. D'Andrea. A computationally efficient algorithm for state-to-state quadrocopter trajectoy generation and feasability verification. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3480–3486, 2013.

[9] M. Hehn and R. D'Andrea. A Frequency Domain Iterative Feed-Forward Learning Scheme for High Performance Periodic Quadrocopter Maneuvers. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2445–2451, 2013.

[10] M. Hehn and R. D'Andrea. Real-time trajectory generation for interception maneuvers with quadrocopters. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4979–4984, 2012.

[11] K. Sreenath, N. Michael, and V. Kumar. Trajectory generation and control of a quadrotor with a cable-suspended load – a differentially-flat hybrid system. In *IEEE International Conference on Robotics and Automation (ICRA)*, to appear, 2013.

[12] K. Sreenath, T. Lee, and V. Kumar. Geometric control and differential flatness of a quadrotor UAV with a cable-suspended load. In *IEEE Conference on Decision and Control (CDC)*, pages 2269–2274, December 2013.

[13] M Grant and S Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[14] J.B. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*. Princeton University Press, Princeton, New Jersey, 2002.

[15] S.O.H. Madgwick, A.J.L. Harrison, and R. Vaidyanathan. Estimation of IMU and MARG orientation using gradient descent algorithm. *IEEE Int. Conf. on Rehabilitation Robotics*, 2011.

[16] B. Wie. *Space Vehicle Dynamics and Control*. AIAA, Reston, Virginia, 1998.

[17] B. Wie, H. Weiss, and A. Arapostathis. Quaternion Feedback Regulator for Spacecraft Eigenaxis Rotations. *AIAA Journal of Guidance, Navigation, and Control*, May 1989.

[18] B. Wie and P.M. Barba. Quaternion Feedback for Spacecraft Large Angle Maneuvers. *AIAA Journal of Guidance, Navigation, and Control*, May 1984.

[19] M. Fliess, J. Levine, and P. Rouchon. Flatness and defect of nonlinear systems: Introductory theory and examples. *International Journal of Controls*, 1995.

[20] R. Murray, M. Rathinam, and W. Sluis. Differential Flatness of Mechanical Control Systems: A Catalog of Prototype Systems. *ASME Int. Mech. Eng. Congress and Expo.*, November 1995.

[21] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, May 2011.

[22] D. Miller. Open Loop System Identification of a Micro Quadrotor Helicopter From Closed Loop Data. Master's thesis, University of Maryland, College Park.

[23] Mark Cutler and Jonathan How. Actuator Constrained Trajectory Generation and Control for Variable-Pitch Quadrotors. *AIAA Guidance, Navigation, and Control Conference*, pages 1–15, August 2012.

[24] N.J. Kasdin and D. Paley. *Engineering Dynamics: A Comprehensive Introduction*. Princeton University Press, Princeton, New Jersey, 2011.

[25] MATLAB. *Optimization Toolbox*. The MathWorks Inc., Natick, Massachusetts, 2014.

[26] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, New York, 2004.

[27] J Conroy, A. Kehlenbeck, J.S. Humbert, and W. Nothwang. Characterization and Enhancement of Micro Brushless DC Motor Response. *SPIE Defense + Security, Micro- and Nanotechnology Sensors, Systems, and Applications VI Conference*, 2014.

[28] A.L. Desbiens, M. Pope, F. Berg, Z.E. Teoh, J. Lee, and M. Cutkosky. Efficient Jumpgliding: Theory and Design Considerations. *IEEE ICRA*, 2013.

[29] E.W. Hawkes, D.L. Christensen, E.V. Eason, M.A. Estrada, M. Heverly, E. Hilgemann, H. Jiang, M.T. Pope, A. Parness, and M.R. Cutkosky. Dynamic Surface Grasping with Directional Adhesion. *IEEE IROS*, 2013.