2012

Trajectory Generation and Control for Quadrotors

Daniel Warren Mellinger
University of Pennsylvania, dwmellin@gmail.com

Trajectory Generation and Control for Quadrotors

Abstract

This thesis presents contributions to the state-of-the-art in quadrotor control, payload transportation with single and multiple quadrotors, and trajectory generation for single and multiple quadrotors. In Ch. 2 we describe a controller capable of handling large roll and pitch angles that enables a quadrotor to follow trajectories requiring large accelerations and also recover from extreme initial conditions. In Ch. 3 we describe a method that allows teams of quadrotors to work together to carry payloads that they could not carry individually. In Ch. 4 we discuss an online parameter estimation method for quadrotors transporting payloads which enables a quadrotor to use its dynamics in order to learn about the payload it is carrying and also adapt its control law in order to improve tracking performance. In Ch. 5 we present a trajectory generation method that enables quadrotors to fly through narrow gaps at various orientations and perch on inclined surfaces. Chapter 6 discusses a method for generating dynamically optimal trajectories through a series of predefined waypoints and safe corridors and Ch. 7 extends that method to enable heterogeneous quadrotor teams to quickly rearrange formations and avoid a small number of obstacles.

Degree Type

Dissertation

Degree Name

Doctor of Philosophy (PhD)

Graduate Group

Mechanical Engineering & Applied Mechanics

First Advisor

Vijay Kumar

Keywords

control, quadrotor, trajectory generation

Subject Categories

Mechanical Engineering | Robotics

TRAJECTORY GENERATION AND CONTROL FOR QUADROTORS

Daniel Mellinger

A DISSERTATION

in

Mechanical Engineering and Applied Mechanics

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2012

Vijay Kumar, PhD, Supervisor of Dissertation

Professor, Department of Mechanical Engineering and Applied Mechanics

Jennifer Lukes, PhD, Graduate Group Chairperson

Associate Professor, Department of Mechanical Engineering and Applied Mechanics

Dissertation Committee:

Mark Yim, PhD, Professor, Mechanical Engineering and Applied Mechanics

Vijay Kumar, PhD, Professor, Mechanical Engineering and Applied Mechanics

Ali Jadbabaie, PhD, Professor, Electrical and Systems Engineering

Raffaello D'Andrea, PhD, Professor, Dynamic Systems and Control

Bruce Kothmann, PhD, Senior Lecturer, Mechanical Engineering and Applied Mechanics

Acknowledgements

During my time at Penn I have learned a lot, worked on interesting research projects, and had fun. I am grateful to everyone who contributed to any of those things.

Specifically, I would first like to thank my advisor, Vijay Kumar, for all of his support during my time at Penn. I have enjoyed the projects we worked on together and his willingness to let me shape the direction of those projects. Vijay is a dedicated advisor, a talented engineer, and a good person and I am lucky to have had him as a mentor.

I would also like to thank my thesis committee members, Bruce Kothmann, Mark Yim, Ali Jadbabaie and Raff D'Andrea, for taking the time to serve on my committee. Bruce has been especially helpful and has always been excited to talk with me about aerodynamics, controls, or whatever math puzzle he is thinking about at the time.

My many friends and colleagues at Penn have made it a fun place to work. I thank them for all the good times inside and outside the lab.

I am indebted to my family for the love and support they have given me over the years. My older sister, Corie, gave me a head start by teaching me whatever she learned in school. My parents gave me every opportunity and allowed me to choose my own direction in life. I can't thank them enough for everything they have done for me.

Last but not least, I thank my wife Anna who always believes in me and always makes me smile.

Table of Contents

Chapter 1

Introduction

Small unmanned rotorcraft have the potential to change the world in a positive way. They can maneuver in three dimensions to collect information with onboard sensors and even physically interact with their environments using onboard grippers. Full-size helicopters are presently used for numerous tasks that their smaller brethren could perform such as traffic, crop, and weather monitoring; building, bridge, and power line inspection; general surveillance activities; as well as aerial photography and videography applications. Unmanned rotorcraft do not require an onboard pilot which means they can be made smaller, safer, and cheaper.

The small size of these vehicles enables them to operate indoors in constrained spaces. This capability will be particularly useful in dangerous situations such as searching for survivors in damaged buildings, entering and clearing buildings with armed adversaries, and collecting information in buildings with biological or nuclear contamination. In these scenarios the ability to create situational awareness without ever having to put a human in harm's way is extremely valuable.

It is true that a ground robot could accomplish some of these described tasks. Ground robots do have a considerable advantage over aerial vehicles in some situations because they can carry larger payloads, are generally more robust to collisions with the environment, do not require active sensing in order to stay in one place, and do not require the

continuous use of energy to stay aloft. However, the ability to fly gives a rotorcraft access to locations that are impossible for a ground robot to reach. Additionally, navigating rough terrain and climbing stairs are difficult tasks for ground vehicles but in most environments the air space is relatively free so the navigation problem for aerial vehicles is considerably easier.

Another alternative to rotorcrafts are fixed-wing vehicles. For navigating long distances the efficiency of a fixed-wing vehicle cannot be beat by a rotorcraft. However, compared with fixed-wing vehicles rotorcraft have the distinct advantage of being able to hover in place. Fixed-wing vehicles, in comparison, must be constantly moving forward to produce lift. The ability to hover is valuable for navigating constrained spaces as well as precisely picking up and dropping off payloads.

Quadrotor helicopters (or quadrotors) are a type of small unmanned rotorcraft. They have four fixed-pitch propellers attached to motors typically mounted in a cross configuration. An image of a quadrotor is shown in Fig. 2.1. They are available from several companies as research and commercial vehicles as well as toys [1, 2, 4, 5]. The long moment arms on which the propellers lie enable them to produce large control moments perform aggressive maneuvers. One of the biggest advantages of quadrotors is their mechanical simplicity. In contrast, small-scale standard helicopters and coaxial helicopters require mechanisms to change the pitch of the propeller in order to produce control forces and moments.

1.1 Related Work

Quadrotors are an excellent robotics platform for many of the reasons mentioned previously. The robotics community has seen a proliferation of research using quadrotors. There have been advances in design as several groups have built and flown quadrotors in the 50 cm range [21, 34, 40, 67] and in the 10 cm range [38, 47]. There has also been work on dynamics and control [21, 33, 40, 51], planning [11, 35, 36], trajectory

generation [26, 32, 37, 54], learning [54], payload transportation [28, 31, 53], and state estimation with onboard sensors [17, 39, 73]. Related work will be discussed in more detail throughout the thesis.

1.2 Motivation and Contributions

While there is no doubt great progress has been made on all aspects of quadrotors, they are not yet used for many of the tasks described at the beginning of this chapter and have not yet reached their full potential. To reach this goal, much work needs to be done to advance the state-of-the-art in all areas of quadrotor technology. The motivation for the work presented in this thesis is to push the limits of the capabilities of quadrotors in certain areas. Specifically, the work described here represents contributions to the state-of-the-art in terms of control, payload transportation with single and multiple vehicles, and trajectory generation for single and multiple vehicles.

First, we describe our approach to modeling and control of a single quadrotor in Ch. 2. We discuss methods that work well for near-hover flight and also methods that enable flight requiring large roll or pitch angles. This *Large-Angle* controller exploits the quadrotor's ability to produce large control moments and enables the vehicle to follow trajectories requiring large accelerations and also recover from extreme initial conditions.

In Ch. 3 we extend these control methods to systems with mutliple quadrotors attached to the same rigid body. This method enables teams of quadrotors to work together to carry payloads that they could not carry individually. In Ch. 4 we discuss online parameter estimation for quadrotors transporting payloads. This work enable a quadrotors to use its dynamics in order to learn something about the payload it is carrying and also adapt its control law in order to improve tracking performance.

The next three chapters (5-7) discuss trajectory generation methods for quadrotor helicopters. In Ch. 5 we present a trajectory generation method where complex trajectories are designed as a sequence of simpler ones. This method enables flight through narrow gaps

at various orientations and perching on inclined surfaces. A second trajectory generation method is presented in Ch. 6 which uses piecewise polynomials to generate dynamically optimal trajectories through a series of predefined waypoints and safe corridors. The last chapter, Ch. 7, discusses a method for generating piecewise polynomial trajectories for teams of heterogeneous quadrotors that enables them to quickly rearrange formations and avoid a small number of obstacles.

It should be noted that the dynamics and control content is included in Ch. 2. Otherwise, each chapter is self-contained and may be read independently from the others. Each chapter contains a discussion of related work, a explanation of the developed method, and a presentation of experimental results.

Chapter 2

Modeling and Control

Many quadrotor controllers operate near hover and rely on small angle assumptions for roll and pitch. Several groups have pushed model rotorcrafts beyond these small angles and created exciting aerobatic flights [9, 33, 54, 56]. However, during the large angle portion of these trajectories there is no position control [33, 54, 56] or position control is not precise enough for obstacle avoidance [9]. Here we describe a control law that is sufficient for small angle flight and a control law for large pitch and roll angles for the purpose of controlling precisely along aggressive trajectories.

2.1 Modeling

2.1.1 Dynamic Model

The coordinate systems and free body diagram for the quadrotor are shown in Fig. 2.2. The world frame, $\mathcal{W}$, is defined by axes x_W, y_W, and z_W with z_W pointing upward. The body frame, $\mathcal{B}$, is attached to the center of mass of the quadrotor with x_B coinciding with the preferred forward direction and z_B perpendicular to the plane of the rotors pointing vertically up during perfect hover (see Fig. 2.2). Rotor 1 is on the positive x_B-axis, 2

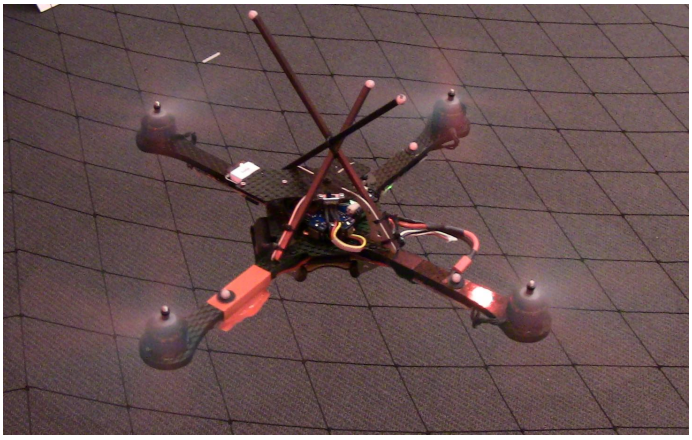


Figure 2.1: Hummingbird Quadrotor [1]

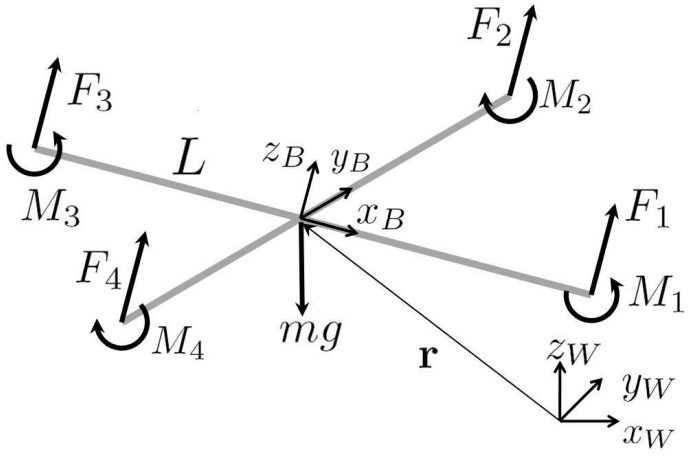


Figure 2.2: Coordinate systems and forces/moments acting on the quadrotor frame.

on the positive y_B-axis, 3 on the negative x_B-axis, 4 on the negative y_B-axis. We use $Z - X - Y$ Euler angles to model the rotation of the quadrotor in the world frame. To get from $\mathcal{W}$ to $\mathcal{B}$, we first rotate about z_W by the yaw angle, ψ, then rotate about the intermediate x-axis by the roll angle, ϕ, and finally rotate about the y_B axis by the pitch angle, θ. The rotation matrix for transforming coordinates from $\mathcal{B}$ to $\mathcal{W}$ is given by

$$R = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix},$$

where $c\theta$ and $s\theta$ denote $\cos(\theta)$ and $\sin(\theta)$, respectively, and similarly for ϕ and ψ. The position vector of the center of mass in the world frame is denoted by $\mathbf{r}$. The forces on the system are gravity, in the $-z_W$ direction, and the forces from each of the rotors, F_i, in the z_B direction. The equations governing the acceleration of the center of mass are

$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \Sigma F_i \end{bmatrix}. \tag{2.1}$$

The components of angular velocity of the robot in the body frame are p, q, and r. These values are related to the derivatives of the roll, pitch, and yaw angles according to

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}.$$

In addition to forces, each rotor produces a moment perpendicular to the plane of rotation of the blade, M_i. Rotors 1 and 3 rotate in the $-z_B$ direction while 2 and 4 rotate in the z_B direction. Since the moment produced on the quadrotor is opposite to the direction of rotation of the blades, M_1 and M_3 act in the z_B direction while M_2 and M_4 act in the $-z_B$ direction. We let L be the distance from the axis of rotation of the rotors to the center of the quadrotor. The moment of inertia matrix referenced to the center of mass along the $x_B - y_B - z_B$ axes, I, is found by weighing individual components of the quadrotor and

building a physically accurate model in SolidWorks. The angular acceleration determined by the Euler equations is

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{2.2}$$

2.1.2 Motor Model

Each rotor has an angular speed ω_i and produces a vertical force F_i according to

$$F_i = k_F \omega_i^2. \tag{2.3}$$

Experimentation with a fixed rotor at steady-state shows that $k_F \approx 6.11 \times 10^{-8} \frac{\text{N}}{\text{rpm}^2}$. The rotors also produce a moment according to

$$M_i = k_M \omega_i^2. \tag{2.4}$$

The constant, k_M, is determined to be about $1.5 \times 10^{-9} \frac{\text{Nm}}{\text{rpm}^2}$ by matching the performance of the simulation to the real system.

The exact relationship between the actual and commanded motor speed is a complicated function of the motor controller and the propeller and motor dynamics. The true performance is a function of the speed of the rotor and whether the speed is increasing or decreasing. However, for simplicity a simple first order motor model is used for controller development and simulation throughout this work. The rotor speed is approximately related to the commanded speed by a first-order differential equation

$$\dot{\omega_i} = k_m(w_i^{\text{des}} - w_i).$$

This motor gain, k_m, is found to be about $20\,\text{s}^{-1}$ by matching the performance of the simulation to the real system. The desired angular velocities, ω_i^{des}, are limited to a minimum and maximum value determined through experimentation to be approximately $1200\,\text{rpm}$ and $7800\,\text{rpm}$.

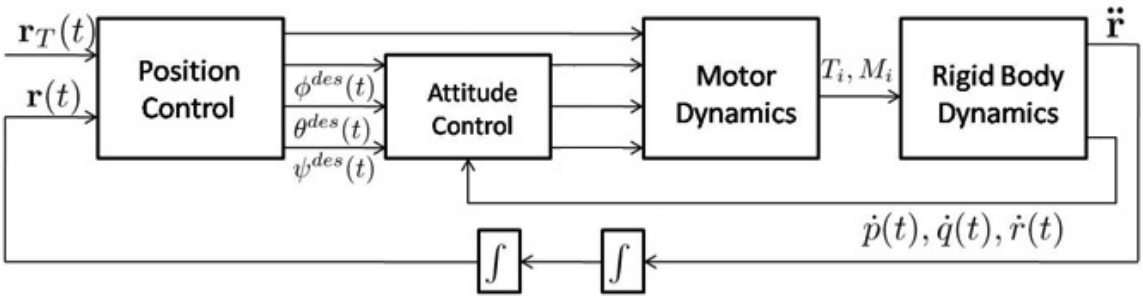


Figure 2.3: The nested control loops for position and attitude control.

2.2 Small Angle Control

Each robot is controlled independently by nested feedback loops as shown in Fig. 2.3. The inner attitude control loop uses onboard accelerometers and gyros to control the roll, pitch, and yaw and runs at approximately 1 kHz [34], while the outer position control loop uses estimates of position and velocity of the center of mass to control the trajectory in three dimensions. Similar nesting of control loops is presented in previous works [13, 21, 30, 34, 40].

Our controllers are derived by linearizing the equations of motion and motor models (3.3 – 2.4) at an operating point that corresponds to the nominal hover state, $\mathbf{r} = \mathbf{r}_0$, $\theta = \phi = 0$, $\psi = \psi_0$, $\dot{\mathbf{r}} = 0$, and $\dot{\phi} = \dot{\theta} = \dot{\psi} = 0$, where the roll and pitch angles are small ($c\phi \approx 1$, $c\theta \approx 1$, $s\phi \approx \phi$, and $s\theta \approx \theta$). At this hover state, the nominal thrusts from the propellers must satisfy

$$F_{i,0} = \frac{mg}{4},$$

and the motor speeds are given by

$$\omega_{i,0} = \omega_h = \sqrt{\frac{mg}{4k_F}}.$$

2.2.1 Attitude Control

We now present an attitude controller to track trajectories in $SO(3)$ that are close to the nominal hover state where the roll and pitch angles are small. From (2.2), if we assume

that the products of inertia are small (ideally, they are zero because the axes are close to the principal axes) and $I_{xx} \approx I_{yy}$ because of the symmetry then:

$$I_{xx}\dot{p} = u_2 - qr(I_{zz} - I_{yy}) \tag{2.5a}$$

$$I_{yy}\dot{q} = u_3 - pr(I_{xx} - I_{zz}) \tag{2.5b}$$

$$I_{zz}\dot{r} = u_4. \tag{2.5c}$$

We can also assume the component of the angular velocity in the z_B direction, r, is small so the rightmost terms in (2.5a) and (2.5b) which are products involving r are small compared to the other terms. We note that near the nominal hover state $\dot{\phi} \approx p$, $\dot{\theta} \approx q$, and $\dot{\psi} \approx r$. For these reasons we can use simple proportional derivative control laws that take the form

$$\begin{aligned} u_{2,des} &= k_{p,\phi}(\phi^{\text{des}} - \phi) + k_{d,\phi}(p^{\text{des}} - p) \\ u_{3,des} &= k_{p,\theta}(\theta^{\text{des}} - \theta) + k_{d,\theta}(q^{\text{des}} - q) \\ u_{4,des} &= k_{p,\psi}(\psi^{\text{des}} - \psi) + k_{d,\psi}(r^{\text{des}} - r). \end{aligned} \tag{2.6}$$

The vector of desired rotor speeds can be found from the desired net force ($u_{1,des}$) and moments ($u_{2,des}$, $u_{3,des}$ and $u_{4,des}$) by inverting

$$\mathbf{u}_{des} = \begin{bmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F L & 0 & -k_F L \\ -k_F L & 0 & k_F L & 0 \\ k_M & -k_M & k_M & -k_M \end{bmatrix} \begin{bmatrix} \omega_{1,des}^2 \\ \omega_{2,des}^2 \\ \omega_{3,des}^2 \\ \omega_{4,des}^2 \end{bmatrix}. \tag{2.7}$$

2.2.2 Position Control

Here we present two representative position control methods that use the roll and pitch angles as inputs via a method similar to a backstepping approach [45]. The first, a hover controller, is used for station-keeping or maintaining the position at a desired x, y, and z location. The second tracks a trajectory in three dimensions.

Hover Controller

Here we use pitch and roll angle to control position in the x_W and y_W plane, u_4 to control yaw angle, and u_1 to control position along z_W. We let $\mathbf{r}_T(t)$ and $\psi_T(t)$ be the trajectory and yaw angle we are trying to track. Note that $\psi_T(t) = \psi_0$ for the hover controller. The command accelerations, $\ddot{r}_i^{\text{des}}$, are calculated from PID feedback of the position error, $e_i = (r_{i,T} - r_i)$, as

$$(\ddot{r}_{i,T} - \ddot{r}_i^{\text{des}}) + k_{d,i}(\dot{r}_{i,T} - \dot{r}_i) + k_{p,i}(r_{i,T} - r_i) + k_{i,i}\int (r_{i,T} - r_i) = 0,$$

where $\dot{r}_{i,T} = \ddot{r}_{i,T} = 0$ for hover.

Then we linearize (3.3) to get the relationship between the desired accelerations and roll and pitch angles

$$\begin{aligned}
\ddot{r}_1^{\text{des}} &= g(\theta^{\text{des}} \cos\psi_T + \phi^{\text{des}} \sin\psi_T) \\
\ddot{r}_2^{\text{des}} &= g(\theta^{\text{des}} \sin\psi_T - \phi^{\text{des}} \cos\psi_T) \\
\ddot{r}_3^{\text{des}} &= \frac{u_{1,des}}{m}.
\end{aligned}$$

These relationships are inverted to compute the desired roll and pitch angles for the attitude controller, from the desired accelerations, as well as $u_{1,des}$

$$\phi^{\text{des}} = \frac{1}{g}(\ddot{r}_1^{\text{des}} \sin\psi_T - \ddot{r}_2^{\text{des}} \cos\psi_T) \tag{2.9a}$$

$$\theta^{\text{des}} = \frac{1}{g}(\ddot{r}_1^{\text{des}} \cos\psi_T + \ddot{r}_2^{\text{des}} \sin\psi_T) \tag{2.9b}$$

$$u_{1,des} = m\ddot{r}_3^{\text{des}}. \tag{2.9c}$$

The position control loop for the hover controller runs at the rate data is received from Vicon (normally around 100 Hz), while the inner attitude control loop runs at 1 kHz. There is the usual trade-off in optimizing the control gains between speed of response and stability. Experimental results show (see the representative trial in Figs. 2.7(a)–2.7(b)) for a tightly optimized "stiff" controller the horizontal positioning errors are within 2 cm and the error in the vertical direction is always less than 0.6 cm. However, this set of gains

leads to a relatively small basin of attraction. By optimizing the gains for a softer response we can increase the size of this basin of attraction. We can experimentally characterize this basin by perturbing the quadrotor from the hover state, and measuring the response of the hover controller. We found the robot to be quite robust if we used the "softer" controller, allowing it to recover from disturbances as large as $1.5\,\mathrm{m}$ (3 body lengths) in the horizontal direction and $2.0\,\mathrm{m}$ (4 body lengths) in the vertical direction, pitch or roll angle errors of $60\,^\circ$, and velocity errors of up to $3.0\,\frac{\mathrm{m}}{\mathrm{s}}$.

3D Trajectory Control

The 3D Trajectory Controller is used to follow three-dimensional trajectories with modest accelerations so the near-hover assumptions hold. We use an approach similar to those described in [40, 62]. We have a method for calculating the closest point on the trajectory, $\mathbf{r}_T$, to the the current position, $\mathbf{r}$. Let the unit tangent vector of the trajectory associated with that point be $\hat{\mathbf{t}}$ and the desired velocity vector be $\dot{\mathbf{r}}_T$. We define the position and velocity errors as

$$\mathbf{e}_p = ((\mathbf{r}_T - \mathbf{r}) \cdot \hat{\mathbf{n}})\hat{\mathbf{n}} + ((\mathbf{r}_T - \mathbf{r}) \cdot \hat{\mathbf{b}})\hat{\mathbf{b}}$$

and

$$\mathbf{e}_v = \dot{\mathbf{r}}_T - \dot{\mathbf{r}}.$$

Note that here we ignore position error in the tangent direction by only considering position error in the normal, $\hat{\mathbf{n}}$, and binormal, $\hat{\mathbf{b}}$, directions.

We calculate the commanded acceleration, $\ddot{r}_i^{\text{des}}$, from PD feedback of the position and velocity errors:

$$\ddot{r}_i^{\text{des}} = k_{p,i}e_{i,p} + k_{d,i}e_{i,v} + \ddot{r}_{i,T}.$$

Note that the $\ddot{r}_{i,T}$ terms represent feedforward terms on the desired accelerations. At low accelerations these terms can be ignored but at larger accelerations they can significantly improve controller performance. Finally we use (2.9a), (2.9b), and (2.9c) to compute the desired roll and pitch angles as well as $u_{1,des}$.

2.2.3 Non-dimensional tuning

We rely on the dynamic model of the vehicle to tune the control law. In order to tune the attitude controller we consider the equation of motion for rotation about the body frame x axis:

$$I_{xx}\ddot{\phi} + k_{d,\phi}\dot{\phi} + k_{p,\phi}\phi = 0 \tag{2.10}$$

This is a second order system so we compare it to

$$\ddot{\phi} + 2\xi\omega_n\dot{\phi} + \omega_n^2\phi = 0 \tag{2.11}$$

where ξ represents the damping ratio for the second order system and ω_n represents its natural frequency. Here we design our controllers to have a certain damping ratio and natural frequency so we pick the attitude control gains according to

$$k_{p,\phi} = I_{xx}\omega_n^2 \tag{2.12}$$

and

$$k_{d,\phi} = 2I_{xx}\xi\omega_n \tag{2.13}$$

Typically we design the attitude controller to be close to critically damped, $\xi \approx 1$, with $\omega_n \approx 9\frac{rad}{s}$.

This response of the attitude controller is limited by the time delay in the motor response which is a complicated function of the propeller and motor controller. In practice we use the method described here as a starting point for tuning our controller experimentally. A similar analysis of the equations of motion are performed to design the position controller to have a certain damping ratio and natural frequency. This non-dimensional approach to controller tuning has been used on vehicles ranging from 65 to 1000 grams.

2.3 Large Angle Control

The control laws presented in the previous sections rely on the assumption that the roll and pitch angles are small. Indeed, most of the work in this area uses controllers that are

derived from the linearization of the model around hover conditions and are stable only under reasonably small roll and pitch angles [40]. In order to follow trajectories requiring large lateral accelerations it is necessary to relax small angle assumptions and allow for significant excursions from the hover state.

In this section we present a controller which does not rely on the Euler angle parameterization of the orientation of the quadrotor. The key problem with the previously presented controller is that when the angular errors are large, the difference in Euler angles between the current and desired attitude is no longer a good metric for defining the orientation error. Rather, one needs to write this error as the 3-D rotation required to get from the current to the desired orientation. Here we use rotation matrices to compute this error as in [51] but one can equivalently use quaternions as is described in [78]. Other than the change in the orientation error metric, the large angle controller described here is conceptually very similar to the controller presented in the previous section.

2.3.1 Model for Control

In practice, the motor dynamics are relatively fast compared to the rigid body dynamics and the aerodynamics. Incorporating the dynamics leads to a fifth order dynamic model with added complexity without significant improvement in performance. Thus we will use a dynamic model based on (3.3-2.2):

$$\dot{\mathbf{r}} = \mathbf{v} \tag{2.14}$$

$$\dot{\mathbf{v}} = -g\mathbf{z}_W + \frac{u_1}{m}\,\mathbf{z}_B \tag{2.15}$$

$${}^W\dot{R}_B = {}^W R_B\,\hat{\omega}_{\mathcal{BW}} \tag{2.16}$$

$$\dot{\omega}_{\mathcal{BW}} = \mathcal{I}^{-1}\left[-\omega_{\mathcal{BW}} \times \mathcal{I}\omega_{\mathcal{BW}} + \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}\right] \tag{2.17}$$

with the state given by the position and velocity of the center of mass and the orientation (locally parameterized by Euler angles) and the angular velocity:

$$\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r]^T$$

or without the parameterization by the the position and velocity of the center of mass and the rotation matrix ${}^W R_B$ and the angular velocity $\omega_{\mathcal{BW}}$. The input is simply:

$$\mathbf{u} = [u_1, u_2, u_3, u_4]^T$$

where u_1 is the force from all the propellers and u_2, u_3, and u_4 are the moments about the body frame axes.

2.3.2 Differential Flatness

In this section we show that the quadrotor dynamics with the four inputs is *differentially flat* [77]. In other words, the states and the inputs can be written as algebraic functions of four carefully selected *flat outputs* and their derivatives. This facilitates the automated generation of trajectories since any smooth trajectory (with reasonably bounded derivatives) in the space of flat outputs can be followed by the underactuated quadrotor.

Our choice of flat outputs is given by:

$$\sigma = [x, y, z, \psi]^T,$$

where $\mathbf{r} = [x, y, z]^T$ are the coordinates of the center of mass in the world coordinate system and ψ is the yaw angle defined earlier. We will define a trajectory, $\sigma(t)$, as a smooth curve in the space of flat outputs:

$$\sigma(t) : [t_0, t_m] \rightarrow \mathbb{R}^3 \times SO(2) \tag{2.18}$$

We will now show that the state of the system can be written in terms of σ and its derivatives. In other words, there exists a smooth map:

$$(\mathbf{x}, \mathbf{u}) = \Phi(\sigma, \dot{\sigma}, \ddot{\sigma}, \dddot{\sigma}, \ddddot{\sigma})$$

Position and orientation

The position, velocity, and acceleration of the center of mass are trivially functions of σ:

$$\begin{aligned}
[x, y, z]^T &= [\sigma_1, \sigma_2, \sigma_3]^T \\
[\dot{x}, \dot{y}, \dot{z}]^T &= [\dot{\sigma}_1, \dot{\sigma}_2, \dot{\sigma}_3]^T \\
[\ddot{x}, \ddot{y}, \ddot{z}]^T &= [\ddot{\sigma}_1, \ddot{\sigma}_2, \ddot{\sigma}_3]^T
\end{aligned}$$

To see that ${}^W R_B$ is a function of the flat outputs and their derivatives, consider the equations of motion (4.2, 4.3). From (4.2),

$$\mathbf{z}_B = \frac{\mathbf{t}}{\|\mathbf{t}\|},$$

where

$$\mathbf{t} = \left[\ddot{\sigma}_1,\ \ddot{\sigma}_2,\ \ddot{\sigma}_3 + g\right]^T, \tag{2.19}$$

which defines the body frame z axis of the quadrotor.

Given the yaw angle, $\sigma_4 = \psi$, we can write the unit vector

$$\mathbf{x}_C = \left[\cos\sigma_4,\ \sin\sigma_4,\ 0\right]^T,$$

by a rotation of the inertial frame through the yaw angle ψ as shown in Figure 7.1. We can determine $\mathbf{x}_B$ and $\mathbf{y}_B$ as follows:

$$\mathbf{y}_B = \frac{\mathbf{z}_B \times \mathbf{x}_C}{\|\mathbf{z}_B \times \mathbf{x}_C\|},\ \mathbf{x}_B = \mathbf{y}_B \times \mathbf{z}_B,$$

provided $\mathbf{x}_C \times \mathbf{z}_B \neq 0$. In other words, we can uniquely determine

$${}^W R_B = [\mathbf{x}_B\ \ \mathbf{y}_B\ \ \mathbf{z}_B]$$

provided we never encounter the singularity where $\mathbf{z}_B$ is parallel[1] to $\mathbf{x}_C$.

[1] Although from a theoretical standpoint we can determine ${}^W R_B$ from the flat outputs and their derivatives almost everywhere, there is a practical limitation in using this map at points near this singularity since the rotation matrix can undergo large changes even with small changes of the flat output. Our practical fix to this problem is discussed later in Section 5.1.

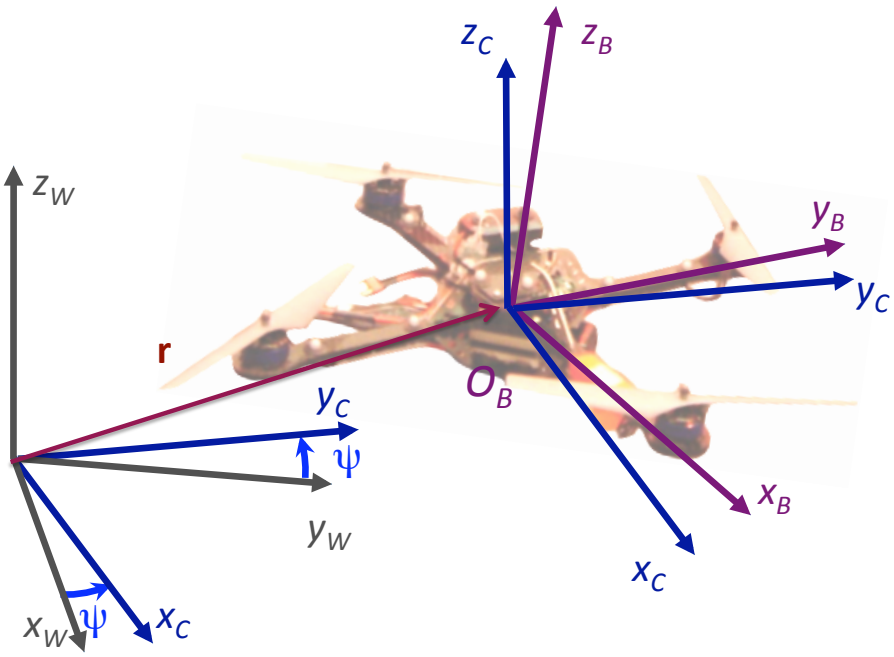


Figure 2.4: The flat outputs and the reference frames.

Angular Velocity

To show the angular velocity is a function of the flat outputs and their derivatives, take the first derivative of (4.2):

$$m\dot{\mathbf{a}} = \dot{u}_1\mathbf{z}_B + \omega_{\mathcal{BW}} \times u_1\mathbf{z}_B \tag{2.20}$$

Projecting this expression along z_B, and using the fact that $\dot{u}_1 = \mathbf{z}_B \cdot m\dot{\mathbf{a}}$, we can substitute $\dot{u}_1$ into (2.20) to define the vector $\mathbf{h}_\omega$:

$$\mathbf{h}_\omega = \omega_{\mathcal{BW}} \times \mathbf{z}_B = \frac{m}{u_1}(\dot{\mathbf{a}} - (\mathbf{z}_B \cdot \dot{\mathbf{a}})\mathbf{z}_B)$$

$\mathbf{h}_\omega$ is the projection of $\frac{m}{u_1}\dot{\mathbf{a}}$ onto the $x_B - y_B$ plane. The first two body frame components of angular velocity, p and q, are found as follows

$$p = -\mathbf{h}_\omega \cdot \mathbf{y}_B, \; q = \mathbf{h}_\omega \cdot \mathbf{x}_B$$

To find the third component we must analyze the equation relating the derivatives of the euler angles to the angular velocity:

$$\omega_{BW} = [\mathbf{x}_C \;\; \mathbf{y}_B \;\; \mathbf{z}_W] \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.21}$$

Provided $\mathbf{y}_B \times \mathbf{z}_W \neq 0$ the matrix in (2.21) is invertible and we can write

$$[\mathbf{x}_C \;\; \mathbf{y}_B \;\; \mathbf{z}_W]^{-1W} R_B \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.22}$$

We can use the third scalar equation in (2.22) with the previously found body-frame angular velocity components, p and q, as well as the chosen value for the derivative of the yaw angle, $\dot{\psi}$, to find the $\mathbf{z}_B$ component of the angular velocity, r.

Angular accelerations

To write angular accelerations as functions of the flat outputs and their derivatives, we take the first derivative of (2.20):

$$\begin{aligned} m\ddot{\mathbf{a}} &= \ddot{u}_1 \mathbf{z}_B + 2\omega_{\mathcal{BW}} \times \dot{u}_1 \mathbf{z}_B \\ &+ \omega_{\mathcal{BW}} \times \omega_{\mathcal{BW}} \times u_1 \mathbf{z}_B + \alpha_{\mathcal{BW}} \times u_1 \mathbf{z}_B \end{aligned} \tag{2.23}$$

Note that here $\alpha_{\mathcal{BW}}$ represents the derivative of the angular velocity, $\omega_{\mathcal{BW}}$, in the body frame. Projection of (2.23) along z_B yields

$$\ddot{u}_1 = \mathbf{z}_B \cdot m\ddot{\mathbf{a}} - \mathbf{z}_B \cdot (\omega_{\mathcal{BW}} \times \omega_{\mathcal{BW}} \times u_1 \mathbf{z}_B)$$

The components of the angular acceleration $\alpha_{\mathcal{BW}}$ along x_B and y_B are found by computing:

$$\mathbf{h}_\alpha = \alpha_{\mathcal{BW}} \times \mathbf{z}_B.$$

Then

$$\dot{p} = -\mathbf{h}_\alpha \cdot \mathbf{y}_B, \;\; \dot{q} = \mathbf{h}_\alpha \cdot \mathbf{x}_B$$

To find the z_B component of the $\alpha_{\mathcal{BW}}$ we must take the first derivative of equation (2.21):

$$ {}^{W}R_B \left(\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) = \omega_{CW} \times \dot{\phi}\mathbf{x}_C + \omega_{BW} \times \dot{\theta}\mathbf{y}_B + [\mathbf{x}_C \;\; \mathbf{y}_B \;\; \mathbf{z}_W] \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} \tag{2.24} $$

Provided $\mathbf{y}_B \times \mathbf{z}_W \neq 0$ we can write:

$$ A \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \mathbf{b} = \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} \tag{2.25} $$

where A and b are a matrix and a vector, respectively, of known value. Then we can calculate $\dot{r}$ from the third scalar equation in (2.25) given the previously found values , $\dot{p}$ and $\dot{q}$, and the chosen value for the second derivative of the yaw angle, $\ddot{\psi}$.

Inputs

The net thrust from the quadrotor propellers is seen to be a direct function of the flat outputs and their derivatives from Equation (4.2,2.19):

$$ u_1 = m\|\mathbf{t}\|. $$

Given the angular velocity and acceleration are functions of the flat outputs and the derivatives we use the Euler equations to compute the inputs as functions of these variables:

$$ \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \mathcal{I} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \mathcal{I} \begin{bmatrix} p \\ q \\ r \end{bmatrix} $$

2.3.3 Control Law

We now present a controller to control along a tracking trajectories, $\sigma_T(t) = [\mathbf{r}_T(t)^T, \psi_T(t)]^T$. This controller is very similar to the one in our previous work [62] with exceptions that

will be pointed out later. First we define the errors on position and velocity:

$$\mathbf{e}_p = \mathbf{r} - \mathbf{r}_T,\ \mathbf{e}_v = \dot{\mathbf{r}} - \dot{\mathbf{r}}_T$$

Next we compute the desired force vector for the controller and the desired body frame z axis:

$$\mathbf{F}_{des} = -K_p\mathbf{e}_p - K_v\mathbf{e}_v - mg\mathbf{z}_W + m\ddot{\mathbf{r}}_T$$

where K_p and K_v are positive definite gain matrices. Note that here we assume $\|\mathbf{F}_{des}\| \neq 0$. Next we project the desired force vector onto the actual body frame z axis in order to compute the desired force for the quadrotor and the first input:

$$u_{1,des} = \mathbf{F}_{des} \cdot \mathbf{z}_B$$

To determine the other three inputs, we must consider the rotation errors. First, we observe that the desired z_B direction is along the desired thrust vector:

$$\mathbf{z}_{B,des} = \frac{\mathbf{F}_{des}}{\|\mathbf{F}_{des}\|}$$

Thus if $\mathbf{e}_3 = [0, 0, 1]^T$, the desired rotation ${}^W R_B$ denoted by R_{des} for brevity is given by:

$$R_{des}\mathbf{e}_3 = \mathbf{z}_{B,des}.$$

Knowing the specified yaw angle along the trajectory, $\psi_T(t)$, we compute $\mathbf{x}_{B,des}$ and $\mathbf{y}_{B,des}$ as in the previous section.:

$$\mathbf{x}_{C,des} = \left[\cos\psi_T,\ \sin\psi_T,\ 0\right]^T,$$

and

$$\mathbf{y}_{B,des} = \frac{\mathbf{z}_{B,des} \times \mathbf{x}_{C,des}}{\|\mathbf{z}_{B,des} \times \mathbf{x}_{C,des}\|},\ \mathbf{x}_{B,des} = \mathbf{y}_{B,des} \times \mathbf{z}_{B,des},$$

provided $\mathbf{x}_{C,des} \times \mathbf{z}_{B,des} \neq 0$. This defines the desired rotation matrix R_{des}. While mathematically this singularity is a single point in $SO(3)$, this computation results in large changes in the unit vectors in the neighborhood of the singularity. To fix this problem, we observe that $-\mathbf{x}_{B,des}$ and $-\mathbf{y}_{B,des}$ are also consistent with the desired yaw angle and body

frame z axis. In practice we simply check which one of the solutions is closer to the actual orientation of the quadrotor in order to calculate the desired orientation, R_{des}.

Next we define the error on orientation:

$$\mathbf{e}_R = \frac{1}{2}(R_{des}^T R - R^T R_{des})^\vee$$

where $^\vee$ represents the *vee map* which takes elements of $so(3)$ to $\mathbb{R}^3$. This is the major departure from the small angle controller where the angular errors were computed using the small angle assumption.

The angular velocity error is simply the difference between the actual and desired angular velocity in body frame coordinates:

$$\mathbf{e}_\omega = {}^B[\omega_{\mathcal{BW}}] - {}^B[\omega_{\mathcal{BW},T}]$$

Now the desired moments and the three remaining inputs are computed as follows:

$$\begin{bmatrix} u_{2,des} \\ u_{3,des} \\ u_{4,des} \end{bmatrix} = -K_R \mathbf{e}_R - K_\omega \mathbf{e}_\omega \tag{2.26}$$

where K_R and K_ω are diagonal gain matrices. This allows unique gains to be used for roll, pitch, and yaw angle tracking. Finally we compute the desired rotor speeds to achieve the desired $\mathbf{u}$. This is done by inverting (4.1).

Note that the linearization about the hover point for this nonlinear controller is the same as the small angle controller. This nonlinear controller presented here adds two new important features. First, the orientation error is not based on the Euler angles which contain singularities. Second, the desired force is projected onto the actual z body axis. Proofs of stability and convergence are presented for a similar controller in [51] but with (a) the addition of feedforward terms including the angular acceleration; (b) the inclusion of feedback terms cancelling the $\omega \times \mathcal{I}\omega$ in (4.4); (c) the assumption that all gain matrices are scalar multiples of the identity (e.g., $K_R = k_R I$); and (d) the assumption that motor dynamics are insignificant. Under these conditions the dynamics are exponentially stable provided the initial conditions satisfies two conditions:

$$tr\left[I - R_{des}^{T}(0)^{W}R_{B}(0)\right] < 2$$

$$\|\mathbf{e}_{\omega}(0)\|^{2} < \frac{2}{\lambda_{min}(\mathcal{I})}k_{R}(1 - \frac{1}{2}tr\left[I - R_{des}^{T}(0)^{W}R_{B}(0)\right])$$

and almost globally exponential attractiveness of the complete dynamics with less restrictive conditions. It is important to note that these results assume infinitely fast motor dynamics, perfect sensing, zero time delay, and perfect knowledge of m and $\mathcal{I}$. In our system, the dynamics cancelling terms involving $\omega \times \mathcal{I}\omega$ are small compared to the other terms and can be neglected since $\mathcal{I}$ is nearly diagonal. We do not include feedforward terms on angular acceleration, which can be significant, and do not use the same gains in all directions as this is not desirable. Thus our realization of the controller is different and does not quite satisfy these assumptions. However, the controller yields good tracking performance even with very large roll and pitch angles.

2.4 Experiments

2.4.1 Experimental Setup

The hardware, software, and implementation details of the experiments follows. The pose of the quadrotor is observed using a VICON motion capture system at some rate less than $225\,\text{Hz}$ [8]. The position is numerically differentiated and low-pass filtered to compute the linear velocity of the robot. These values are available to MATLAB via ROS [6] and a ROS-MATLAB bridge [7]. All commands are computed in MATLAB using the latest state estimate at the rate of the VICON. The commands in MATLAB are bridged to ROS and the most recent command is sent to the robot via ZIGBEE at a fixed rate of $100\,\text{Hz}$. This fixed rate is due to the limited bandwidth of ZIGBEE ($57.6\,\text{kbps}$). Commands sent to the robot consist of the gains and desired attitude, desired angular velocities, and thrust values.

The robot (Fig. 1.2.1) is sold commercially [1]. The quadrotor follows a standard four-propeller design and is equipped with two embedded processors running at $1\,\text{kHz}$ (denoted

as high-level (HL) and low-level (LL)), an IMU (running at 300 Hz). The HL processor runs our custom firmware, receives commands via ZIGBEE, and sends motor commands to the LL processor to execute those commands. The LL processor provides attitude estimates to the HL processor. A comparison between attitude estimates and VICON shows very similar results. Latencies and data flow are shown in Fig. 2.5.

2.4.2 Controller Performance

Here we illustrate the performance of the basic control modes. We performed experiments to characterize the performance of the attitude controller. The quadrotor was commanded to gain vertical velocity in order to enter a free-fall state. Then a desired pitch angle was commanded for 0.6 seconds while a nominal net thrust of $0.2mg$ was commanded. The data shown in Fig. 2.6 demonstrates the performance for desired angles of 60° and 90°.

We present time histories of positions for both the hover and trajectory following controllers in Figs. 2.7(a)–2.7(d). Standard deviations for the $x - y$ and z errors are shown in Tab. 2.1 for hover control with stiff gains and 3-D path following along a specified circular path. The VICON motion capture provides the position update at up to 225 Hz. To simulate a slower localization method we tested the hover control with the position and velocity updated at slower rates as shown in Table 2.1. Note that we found the performance of the hover controller to be the same at rates above 35 Hz. In addition to being slower, an alternate localization method could have larger errors which would further decrease performance. We simulated this situation by adding gaussian noise to the position estimate from the motion capture system as is also shown in Table 2.1. However, despite these limitations similar trajectories as those demonstrated in this work should be possible, albeit less precise, with a less accurate localization method.

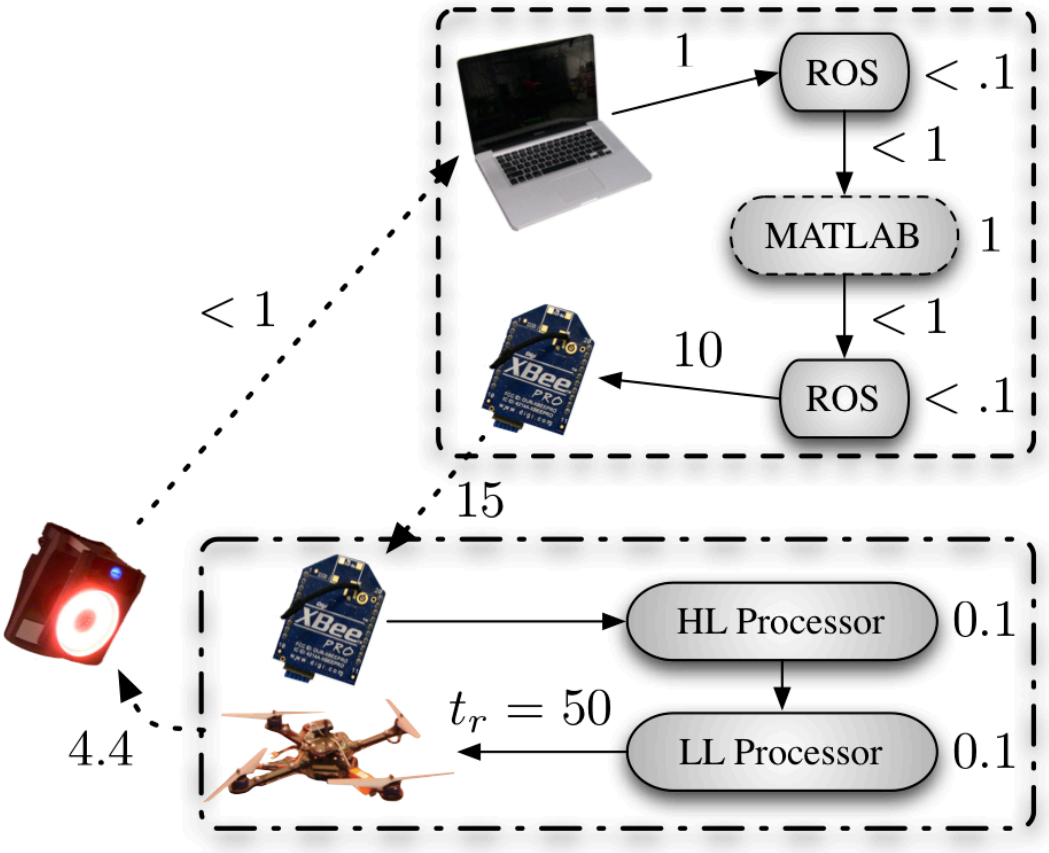
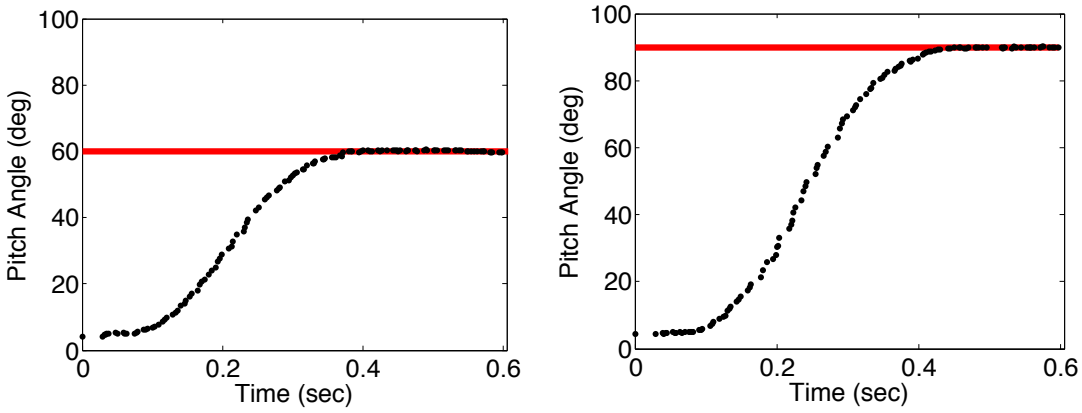


Figure 2.5: Latencies (ms) in experimental system. Here the motion capture system operates at 225 Hz. Quadrotor pose and velocity data is received and bridged to Matlab. Commands are computed in Matlab and the latest command is sent to the robot via Zigbee at 100 Hz (throttled due to bandwidth limitations). Commands are received on the robot; the high-level (HL) embedded micro-processor computes direct motor commands, then sends the motor commands to the lower-level (LL) processor. Our custom firmware runs on the HL processor and the proprietary firmware runs on the LL processor. The motor response time is denoted as t_r. Negligible times (< 0.01 ms) are not noted. The worst case response of the system from observation to motors rotating based on the observation is approximately 85 ms with t_r as the dominant limiting factor.



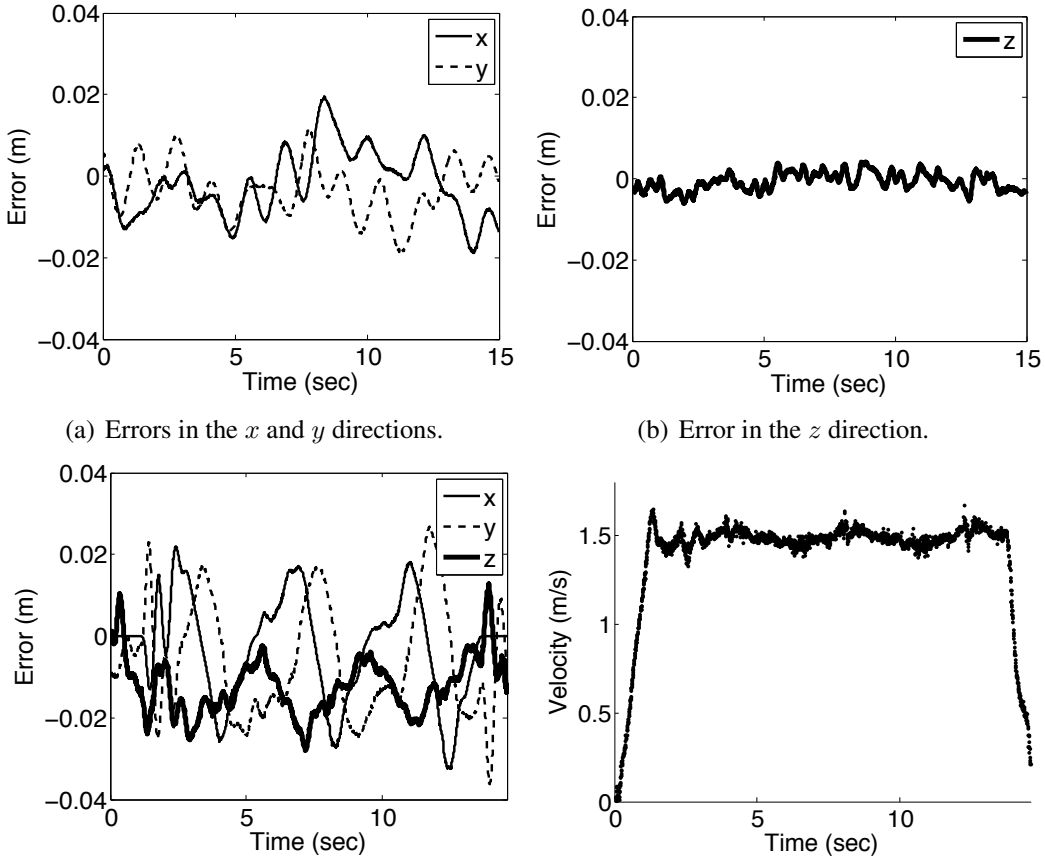
(a) Tracking 60°

(b) Tracking 90°

Figure 2.6: Pitch angle response to step inputs.

Type	Update Rate (Hz)	Added Noise	$x-y$ error (cm)	z error (cm)
Hover Control	225	0	0.7	0.3
Hover Control	20	0	0.8	0.6
Hover Control	20	$\sigma = 1$ cm	1.1	0.9
Hover Control	20	$\sigma = 2$ cm	1.6	0.9
Hover Control	20	$\sigma = 4$ cm	1.8	1.3
Hover Control	10	0	1.5	0.85
Hover Control	6	0	2.6	1.2
Path Following	225	0	1.3	0.7

Table 2.1: Controller performace data. *Update Rate* corresponds to the rate the position and velocity were updated in the controller. *Added Noise* corresponds to the standard deviation of the gaussian noise that was added to the position estimate along each axis. Standard deviations are shown in the rightmost columns for hover control and 3-D path following of a 1 meter radius circle with the axis making a 45° angle from the vertical at 1.5 $\frac{m}{s}$.



(a) Errors in the x and y directions.

(b) Error in the z direction.

(c) Errors in position.

(d) Velocity history.

Figure 2.7: Representative results with the hover controller (Figs. 2.7(a)–2.7(b)). Experimental results from the 3D trajectory controller tracking a circle of radius 1 m with the axis making a 45° angle from the vertical at $1.5\ \frac{m}{s}$ (Figs. 2.7(c)–2.7(d)). Here the position and velocity update rate is 100 Hz.

Chapter 3

Multi-Vehicle Modeling and Control

In this chapter, we consider the problem of controlling multiple quadrotor robots that cooperatively grasp and transport a payload in three dimensions. We model the quadrotors both individually and as a group rigidly attached to a payload. We propose individual robot control laws defined with respect to the payload that stabilize the payload along three-dimensional trajectories. We describe the design of a gripping mechanism attached to each quadrotor that permits autonomous grasping of the payload. An experimental study with teams of quadrotors cooperatively grasping, stabilizing, and transporting payloads along desired three-dimensional trajectories is presented with performance analysis over many trials for different payload configurations.

3.1 Introduction

Autonomous grasping, manipulation, and transportation of objects is a fundamental area of robotics research important to applications which require robots to interact and effect change in their environment. With recent advancements in relevant technologies and commercially available micro aerial vehicles (MAVs), the problem of autonomous grasping, manipulation, and transportation is advancing to the aerial domain in both theory and experiments. However, MAVs have a reduced payload capacity due to power constraints

and are fundamentally limited in their ability to manipulate and transport objects of any significant size. In this paper we address this limitation and consider the problem of controlling multiple quadrotor robots that cooperatively grasp and transport a payload in three dimensions.

We approach the problem by first developing a model for a team of quadrotors rigidly attached to a payload (Sect. 3.3). In Sect. 3.5, we propose individual robot control laws defined with respect to the payload that stabilize the payload along three-dimensional trajectories. An experimental study with teams of quadrotors cooperatively grasping, stabilizing, and transporting payloads of different configurations to desired positions and along three-dimensional trajectories is presented in Sect. 3.7.

3.2 Related Literature

The problem of aerial manipulation using cables is analyzed in [28, 60] with the focus on finding robot configurations that ensure static equilibrium of the payload at a desired pose while respecting constraints on the tension. We address a different problem as the robots use "grippers" that grasp the payload via rigid connections at multiple locations. The modeling of contact constraints is considerably simpler as issues of form or force closure are not relevant. Additionally, contact conditions do not change in our case (e.g., rolling to sliding, or contact to no contact). However, the system is statically indeterminate and the coordination of multiple robots is significantly more complex than in the case when the payload is suspended from aerial robots. In particular, as the problem is over-constrained the robots must control to move in directions that are consistent with kinematic constraints.

There is extensive literature on multi-fingered grasping and legged locomotion that discusses the problem of coordinating robot actuators with kinematic constraints [20, 48, 69]. However, our work is different in many ways. First, unlike legs or fingers, we have less control over the wrenches that can be exerted at each contact. Each robot is capable of

controlling propellers to exert wrenches of a fixed pitch, (i.e., a thrust and a moment proportional to the thrust, both perpendicular to the plane of the rotor). Second, the robot system can be underactuated if the planes associated with each rotor are all parallel. In fact, this is generally the case in formation flight and it may be desirable to grasp the payload at multiple points, allowing the quadrotors to be in parallel horizontal planes. Third, the control of quadrotors necessitate dynamic models that reconcile the aerodynamics of flight with the mechanics of cooperative manipulation.

In this work we take advantage of the fact that we have access to many rotors to generate the thrust necessary to manipulate payloads. A similar concept is presented in [64], where the authors propose control laws that drive a distributed flight array consisting of many rotors along a desired trajectory. However, our control methods differ considerably as we are working with quadrotor robots and must derive feedback control laws based on the control inputs required by these robots. Similar to the concept of using multiple rotors in a flight array is the development of an aerial robot with more than four rotors (as in quadrotors), such as the commercially available *Falcon* with eight rotors from Ascending Technologies, GmBH [1].

A gripping mechanism is presented in this work that enables autonomous grasping of the payload by the quadrotors. Toward this design, we build upon considerable research in the area of climbing robots which generally rely on clinging to surface asperities via microspine arrays [16]. Similar designs with microspine arrays enable aerial vehicles to perch on vertical walls [27]. These robots do not require penetration to cling to the wall. However, in our work, the normal forces required to grasp objects are much higher compared to the shear forces that are exerted on the surfaces interfacing with the spines. Using similar microspine technology, we utilize the advantages of penetration in softer material such as wood and cardboard to attach to horizontal planar surfaces.

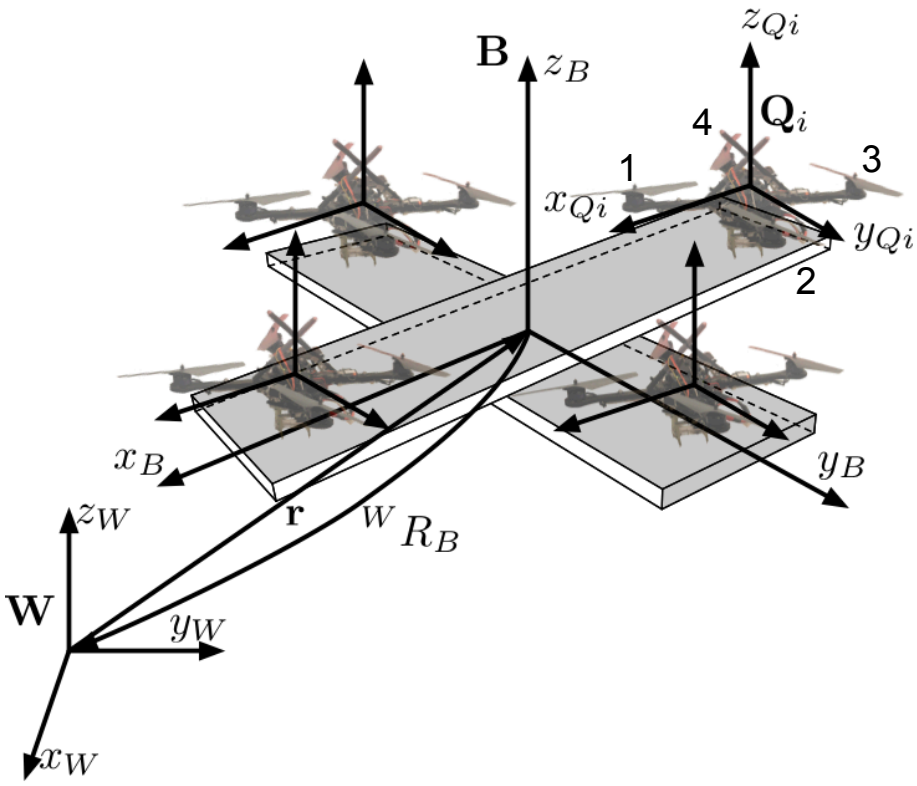


Figure 3.1: The coordinate systems.

3.3 Modeling

3.3.1 Coordinate Systems

The coordinate frames for the multiple quadrotor system are shown in Fig. 3.1. The world frame, $\mathbf{W}$, is defined by axes x_W, y_W, and z_W with z_W pointing upward. We consider n quadrotors rigidly attached to a body frame, $\mathbf{B}$. It is assumed that the body frame axes are chosen as the principal axes of the entire system. Each quadrotor has an individual body frame, $\mathbf{Q}_i$, attached to its center of mass with z_{Qi} perpendicular to the plane of the rotors and pointing vertically up. Let (x_i, y_i, z_i) be the coordinates of the center of mass of the i^{th} quadrotor in $\mathbf{B}$ coordinates and ψ_i be the relative yaw angle. For this quadrotor, rotor 1 is on the positive x_{Qi}-axis, 2 on the positive y_{Qi}-axis, 3 on the negative x_{Qi}-axis, 4 on the negative y_{Qi}-axis. We require the z_{Qi} axes and z_B to be parallel. We still use ZXY Euler

angles to model the rotation of the body (and the quadrotors) in the world frame. The position vector of the center of mass of the body in the world frame is denoted by $\mathbf{r}$. The rotation matrix, Euler angles, angular velocities, and position vector to the center of mass of the i^{th} quadrotor are denoted as ${}^{W}R_{Qi}$, $(\phi_i,\ \theta_i,\ \psi_i)$, $(p_i,\ q_i,\ r_i)$, and $\mathbf{r}_i$, respectively.

3.3.2 Equations of Motion

Each of the j rotors on each of the the i quadrotors produces a force, $F_{i,j}$, and moment, $M_{i,j}$, in the z_{Qi} direction. These rotor forces can be rewritten as a total force from each quadrotor $F_{q,i}$ as well as moments about each of the quadrotor's body frame axes:

$$\begin{bmatrix} F_{q,i} \\ M_{xq,i} \\ M_{yq,i} \\ M_{zq,i} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & L & 0 & -L \\ -L & 0 & L & 0 \\ \frac{k_M}{k_F} & -\frac{k_M}{k_F} & \frac{k_M}{k_F} & -\frac{k_M}{k_F} \end{bmatrix} \begin{bmatrix} F_{i,1} \\ F_{i,2} \\ F_{i,3} \\ F_{i,4} \end{bmatrix}, \tag{3.1}$$

where L is the distance from the axis of rotation of the rotors to the center of the quadrotor. The total force and moments on the system from the quadrotors in the body frame coordinates, $\mathbf{B}$, are:

$$\begin{bmatrix} F_B \\ M_{xB} \\ M_{yB} \\ M_{zB} \end{bmatrix} = \sum_i \begin{bmatrix} 1 & 0 & 0 & 0 \\ y_i & cos\psi_i & -sin\psi_i & 0 \\ -x_i & sin\psi_i & cos\psi_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_{q,i} \\ M_{xq.i} \\ M_{yq.i} \\ M_{zq.i} \end{bmatrix}. \tag{3.2}$$

Note that z_i is not present in (3.2) so this formulation allows for quadrotors in different planes. If we let m be the mass of the entire system and ignore air drag. Then the equations governing the acceleration of the center of mass are simply

$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + {}^{W}R_B \begin{bmatrix} 0 \\ 0 \\ F_B \end{bmatrix}. \tag{3.3}$$

The moment of inertia matrix for the entire system referenced to the center of mass along the $x_B - y_B - z_B$ axes is denoted by I. We assume $x_B - y_B - z_B$ are chosen such that I is diagonal. The angular accelerations determined by the Euler equations are

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} M_{xB} \\ M_{yB} \\ M_{zB} \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}.$$

3.4 Control Basis Vectors

The linear system in (3.2) defines four equations with $4n$ unknowns and can be rewritten as:

$$\begin{bmatrix} F_B, & M_{xB}, & M_{yB}, & M_{zB} \end{bmatrix}^{\mathrm{T}} = A\mathbf{u},$$

where $A \in \mathbb{R}^{4\times 4n}$ is fixed and determined by the relative positions and orientations of the n quadrotors. Here $\mathbf{u} \in \mathbb{R}^{4n}$ contains the four control inputs for each of the quadrotors:

$$\mathbf{u} = [F_{q,1}, M_{xq,1}, M_{yq,1}, M_{zq,1}, ..., F_{q,n}, M_{xq,n}, M_{yq,n}, M_{zq,n}]^T$$

For a system with more than one quadrotor the linear system is underdetermined so we have a choice on how to achieve net forces and moments on the entire system. Here we design our control basis vectors to minimize the cost function:

$$J = \sum_i w_{Fi} F_{q,i}^2 + w_{Mxi} M_{xq,i}^2 + w_{Myi} M_{yq,i}^2 + w_{Mzi} M_{zq.i}^2.$$

For example, we may wish to choose a control input basis vector u_{Mx} to achieve a unit moment in x_B while minimizing the cost function:

$$\mathbf{u}_{Mx} = \underset{\mathbf{u}}{\operatorname{argmin}} \{J | [0,1,0,0]^{\mathrm{T}} = A\mathbf{u}\}. \tag{3.4}$$

A natural way to treat the point-wise minimization of the function J is by choosing control inputs using the Moore-Penrose inverse. First we define $H \in \mathbb{R}^{4n\times 4n}$ so that $J = \|H\mathbf{u}\|_2^2$:

$$H = \text{diag}\left(\sqrt{w_{F1}}, \sqrt{w_{Mx1}}, \sqrt{w_{My1}}, \sqrt{w_{Mz1}}, ..., \sqrt{w_{Fn}}, \sqrt{w_{Mxn}}, \sqrt{w_{Myn}}, \sqrt{w_{Mzn}}\right).$$

After algebraic manipulation:

$$\mathbf{u}_{Mx} = H^{-1}(AH^{-1})^{+}[0,1,0,0]^{\mathrm{T}} = H^{-2}A^{\mathrm{T}}(AH^{-2}A^{\mathrm{T}})^{-1}[0,1,0,0]^{\mathrm{T}}, \tag{3.5}$$

where $^{+}$ denotes the Moore-Penrose inverse. This same process is used to compute control basis vectors $\mathbf{u}_F$, $\mathbf{u}_{My}$, and $\mathbf{u}_{Mz}$.

We now consider the special case in which all quadrotors are identical and axially symmetric meaning roll and pitch can be treated the same way. Indeed this is the case in our experimental testbed. In this case $w_{Fi} = w_F$, $w_{Mxi} = w_{Myi} = w_{Mxy}$, and $w_{Mzi} = w_{Mz}$. Consider the following term from (3.5) for this case:

$$AH^{-2}A^{T} = \begin{bmatrix} \frac{n}{w_F} & \frac{\sum y_i}{w_F} & -\frac{\sum x_i}{w_F} & 0 \\ \frac{\sum y_i}{w_F} & \frac{\sum y_i^2}{w_F} + \frac{n}{w_{Mxy}} & \frac{-\sum x_i y_i}{w_F} & 0 \\ -\frac{\sum x_i}{w_F} & \frac{-\sum x_i y_i}{w_F} & \frac{\sum x_i^2}{w_F} + \frac{n}{w_{Mxy}} & 0 \\ 0 & 0 & 0 & \frac{n}{w_{Mz}} \end{bmatrix}. \tag{3.6}$$

Here we can assume that the positions of the quadrotors dominate the mass properties of the entire structure since the quadrotors are heavier than what they can carry. The x and y locations of the center of mass of the payload and quadrotors together are close to that of just the quadrotors so $\sum x_i = \sum y_i = 0$. Additionally, $\sum x_i y_i = 0$, as the principle axes of the quadrotors are aligned with the principal axes of the structure. Therefore, all quadrotors contribute an equal force and yaw moment to produce a net body force or yaw moment:

$$\mathbf{u}_F = \frac{1}{n}\left[1,0,0,0,...,1,0,0,0\right]^{\mathrm{T}}$$
$$\mathbf{u}_{Mz} = \frac{1}{n}\left[0,0,0,1,...,0,0,0,1\right]^{\mathrm{T}}.$$

The control basis vectors for moments in pitch and roll reflect the tradeoff between the weighting factors:

$$\mathbf{u}_{Mx} = \frac{1}{\frac{w_{Mxy}}{w_F}\sum y_i^2 + n}\left[\frac{w_{Mxy}}{w_F}y_1, c\psi_1, s\psi_1, 0, ..., \frac{w_{Mxy}}{w_F}y_n, c\psi_n, s\psi_n, 0\right]^{\mathrm{T}}$$
$$\mathbf{u}_{My} = \frac{1}{\frac{w_{Mxy}}{w_F}\sum x_i^2 + n}\left[-\frac{w_{Mxy}}{w_F}x_1, -s\psi_1, c\psi_1, 0, ..., -\frac{w_{Mxy}}{w_F}x_n, -s\psi_n, c\psi_n, 0\right]^{\mathrm{T}}.$$

Here, an increase in the cost of individual quadrotor moments relative to the forces, w_{Mxy}/w_F, causes the individual body forces used to create a net body moment to increase and the individual body moments from each quadrotor to decrease. This ratio allows a user to tradeoff between the individual quadrotor force and moments used to create body moments in pitch and roll.

3.5 Multi-Vehicle Small Angle Control

Here we recycle the small angle controller used for the single quadrotor for the multi-vehicle system.

3.5.1 Attitude Control

To control the attitude of the body we use proportional derivative control laws that take the form

$$\begin{aligned} M_{xB}^{\text{des}} &= k_{p,\phi}(\phi^{\text{des}} - \phi) + k_{d,\phi}(p^{\text{des}} - p) \\ M_{yB}^{\text{des}} &= k_{p,\theta}(\theta^{\text{des}} - \theta) + k_{d,\theta}(q^{\text{des}} - q) \\ M_{zB}^{\text{des}} &= k_{p,\psi}(\psi^{\text{des}} - \psi) + k_{d,\psi}(r^{\text{des}} - r). \end{aligned} \tag{3.7}$$

3.5.2 Hover Controller

Here we use pitch and roll angle of the entire system to control position in the x_W and y_W plane, M_{zB}^{des} to control yaw angle, and F_B^{des} to control position along z_W. We let $\mathbf{r}_T(t)$ and $\psi_T(t)$ be the trajectory and yaw angle we are trying to track. Note that $\psi_T(t) = \psi_0$ for the hover controller. The command accelerations, $\ddot{\mathbf{r}}^{\text{des}}$, are calculated from PID feedback of the position error, $\mathbf{e} = (\mathbf{r}_T - \mathbf{r})$, as

$$(\ddot{\mathbf{r}}_T - \ddot{\mathbf{r}}^{\text{des}}) + K_d(\dot{\mathbf{r}}_T - \dot{\mathbf{r}}) + K_p(\mathbf{r}_T - \mathbf{r}) + K_i \int (\mathbf{r}_T - \mathbf{r}) = \mathbf{0},$$

where $\dot{\mathbf{r}}_T = \ddot{\mathbf{r}}_T = \mathbf{0}$ for hover. We linearize (3.3) to get the relationship between the desired accelerations and roll and pitch angles

$$\begin{aligned}
\ddot{r}_1^{\text{des}} &= g(\theta^{\text{des}} \cos\psi_T + \phi^{\text{des}} \sin\psi_T) \\
\ddot{r}_2^{\text{des}} &= g(\theta^{\text{des}} \sin\psi_T - \phi^{\text{des}} \cos\psi_T) \\
\ddot{r}_3^{\text{des}} &= \frac{1}{m} F_B^{\text{des}} - g.
\end{aligned}$$

These relationships are inverted to compute the desired roll and pitch angles for the attitude controller, from the desired accelerations, as well as F_B^{des}:

$$\begin{aligned}
\phi^{\text{des}} &= \frac{1}{g}(\ddot{r}_1^{\text{des}} \sin\psi_T - \ddot{r}_2^{\text{des}} \cos\psi_T) \\
\theta^{\text{des}} &= \frac{1}{g}(\ddot{r}_1^{\text{des}} \cos\psi_T + \ddot{r}_2^{\text{des}} \sin\psi_T) \\
F_B^{\text{des}} &= m(\ddot{r}_3^{\text{des}} + g).
\end{aligned}$$

We substitute these into (3.7) to yield the desired net body force and moments. From these quantities the control inputs for individual quadrotors are computed using the control basis vectors developed in Sec. 3.4:

$$\mathbf{u} = F_B^{\text{des}} \mathbf{u}_F + M_{xB}^{\text{des}} \mathbf{u}_{Mx} + M_{yB}^{\text{des}} \mathbf{u}_{My} + M_{zB}^{\text{des}} \mathbf{u}_{Mz}.$$

We calculate the desired angular velocities from (2.3,4.1) for each of the $4n$ rotors to determine the lowest level control input.

3.5.3 3D Trajectory Control

The trajectory controller is used to follow 3D trajectories with modest accelerations so the near-hover assumptions hold. This is nearly the same controller used for the single quadrotor. We have a method for calculating the closest point on the trajectory, $\mathbf{r}_T$, to the the current position, $\mathbf{r}$. Let the unit tangent vector of the trajectory associated with that point be $\hat{\mathbf{t}}$ and the desired velocity vector be $\dot{\mathbf{r}}_T$. We define the position and velocity errors as

$$\mathbf{e}_p = ((\mathbf{r}_T - \mathbf{r}) \cdot \hat{\mathbf{n}})\hat{\mathbf{n}} + ((\mathbf{r}_T - \mathbf{r}) \cdot \hat{\mathbf{b}})\hat{\mathbf{b}}$$

and

$$\mathbf{e}_v = \dot{\mathbf{r}}_T - \dot{\mathbf{r}}.$$

Note that here we ignore position error in the tangent direction by only considering position error in the normal, $\hat{\mathbf{n}}$, and binormal, $\hat{\mathbf{b}}$, directions. This is done because we are more concerned about reducing the cross-track error rather than error in the tangent direction of the trajectory.

We calculate the commanded acceleration, $\ddot{\mathbf{r}}^{\text{des}}$, from PD feedback of the position and velocity errors:

$$\ddot{\mathbf{r}}^{\text{des}} = K_p\mathbf{e}_p + K_d\mathbf{e}_v + \ddot{\mathbf{r}}_T.$$

Note that $\ddot{\mathbf{r}}_T$ represents feedforward terms on the desired accelerations. At low accelerations these terms can be ignored but at larger accelerations they can significantly improve controller performance. Finally, we use the process described in Sec. 3.5.2 to compute the desired angular velocities for each rotor.

3.6 Decentralized Control Law

We assume the quadrotors are attached rigidly to the body. As long as each quadrotor knows its fixed relative position and orientation with respect to the body and the goal of the body controller (hover location or desired trajectory) then this controller can be decentralized. If each quadrotor senses its own orientation and angular velocity then the orientation and angular velocity of the body are calculated as follows:

$$^{W}R_B = {}^{W}R_{Q_i}\,{}^{Q_i}R_B \quad \text{and} \quad [p, q, r]^{\mathrm{T}} = {}^{B}R_{Q_i}[p_i, q_i, r_i]^{\mathrm{T}}.$$

From the position and velocity of the i^{th} quadrotor, the position and velocity of the center of mass of the body are calculated as:

$$\begin{aligned}\mathbf{r} &= \mathbf{r}_i - {}^{W}R_B[x_i, y_i, z_i]^{\mathrm{T}} \\ \dot{\mathbf{r}} &= \dot{\mathbf{r}}_i - \omega_{\mathbf{B}} \times \left({}^{W}R_B[x_i, y_i, z_i]^{T}\right).\end{aligned}$$

Each quadrotor runs a hover or velocity controller along with the attitude controller (3.7). While we evaluated this decentralized approach experimentally on the robots, for the results presented in this work, we use a centralized formulation. For centralized control the state estimates from the n quadrotors are combined to create a single estimate of the state of the entire body from which the control inputs are computed. This averaging reduces the noise on the state estimate of the entire body and thus results in a cleaner control signal. For this reason, the centralized formulation yields better results.

3.7 Results

In this section we describe results from two experimental trials designed to evaluate the performance of the multi-vehicle controllers and demonstrate cooperative transportation of payloads in 3D.

The hardware, software, and implementation details of the experiments follows. The pose of the quadrotor is observed using a VICON motion capture system at 100 Hz [8]. The pose is numerically differentiated to compute the linear and angular velocities of the robot. These values are available to MATLAB via ROS [6] and a ROS-MATLAB bridge [7]. All commands are computed in MATLAB using the latest state estimate at the rate of the VICON. The commands in MATLAB are bridged to ROS and the most recent command is sent to the robot via ZIGBEE at a fixed rate of 100 Hz. This fixed rate is due to the limited bandwidth of ZIGBEE (57.6 kbps). Commands sent to the robot consist of the gains and desired attitude, angular velocities, and thrust values described in Sect. 3.5.

The first experimental trial consists of a team of four quadrotors rigidly attached to different payload configurations (see Figures 3.2 and 3.4). For this test, we wish to focus on cooperative manipulation and transportation and as such use a payload structure built of wood with quadrotors attachments made via Velcro for easy rearranging. The total mass and x and y principal moments of inertia for each configuration (payload and quadrotors)

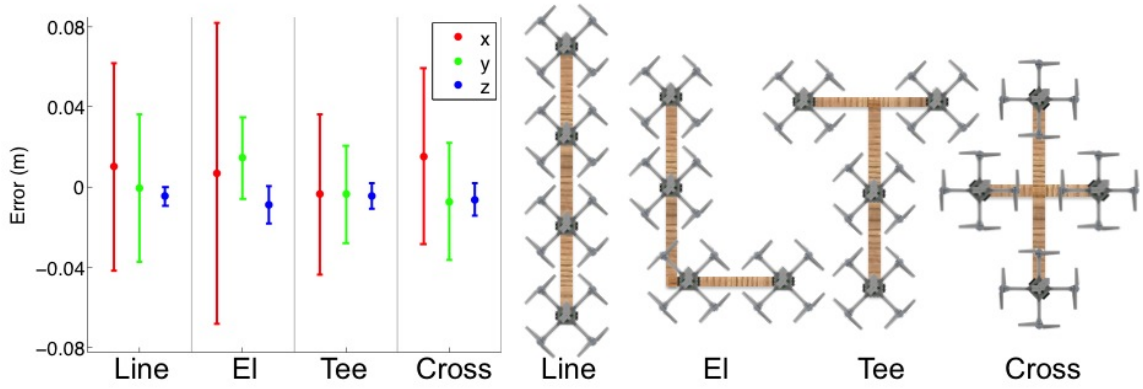


Figure 3.2: Hover performance data for 40 seconds for four configurations. The center dot represents the mean error and the error bars represent one standard deviation. Graphical depictions of the four configurations.

Configuration	I_{xx} (kg m^2)	I_{yy} (kg m^2)	m (kg)
Line	0.0095	0.73	3.33
El	0.079	0.50	3.33
Tee	0.082	0.43	3.33
Cross	0.11	0.19	3.23

Table 3.1: Mass and Inertia Properties.

are shown in Table 3.1. Note that the mass of a single quadrotor with a battery is about 500 g, so in each of these configurations the total payload is greater than 1.2 kg.

For each configuration, the control basis vectors are computed as described in Sect. 3.4 with $w_{Mxy}/w_F = 2$. We chose this ratio as our connection to the payload is stronger in resisting a force pulling it away from a surface than a moment in pitch or roll. Data for each configuration is shown in Fig. 3.2. Note that for each configuration, control along the x axis is intentionally performed with the body angle corresponding to the larger principal moment of inertia, I_{yy}. The performance along the x axis is worse than the y axis as expected. A large moment of inertia limits the bandwidth of the control on that angle. This decrease in attitude control performance leads to decreased position control performance along that axis. Here we note that position control for a single quadrotor is much better

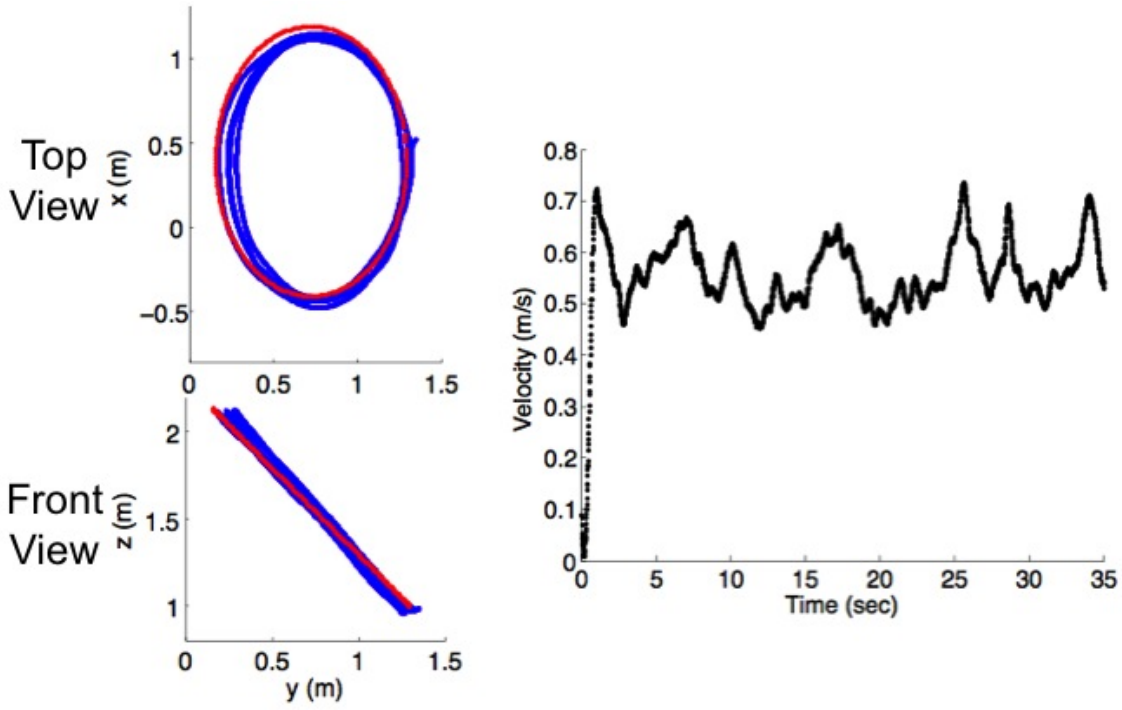


Figure 3.3: Trajectory tracking data for the *Cross* configuration along a $0.8,$ m radius circle tilted at 45° from horizontal at 0.6 m/s.

than for any of the multi-robot structures because their moments of inertia are much larger.

The trajectory tracking controller in Sect. 3.5.3 is implemented on the *Cross* configuration for which data is shown in Fig. 3.3. We see that the system performs well and controls to the desired trajectory in three-dimensions.

The gripping mechanism described in [58] is used on two quadrotors to pick up and transport an 0.8 m, 320 g structure as shown in Fig. 3.4. The quadrotors first descend to the structure and engage the gripping mechanism. The quadrotors ascend with the structure and fly twice along the same circular trajectory as in Fig. 3.3 at $0.5\,\mathrm{m/s}$. Finally, the quadrotors descend to structures initial location, disengage the gripping mechanism, and depart.

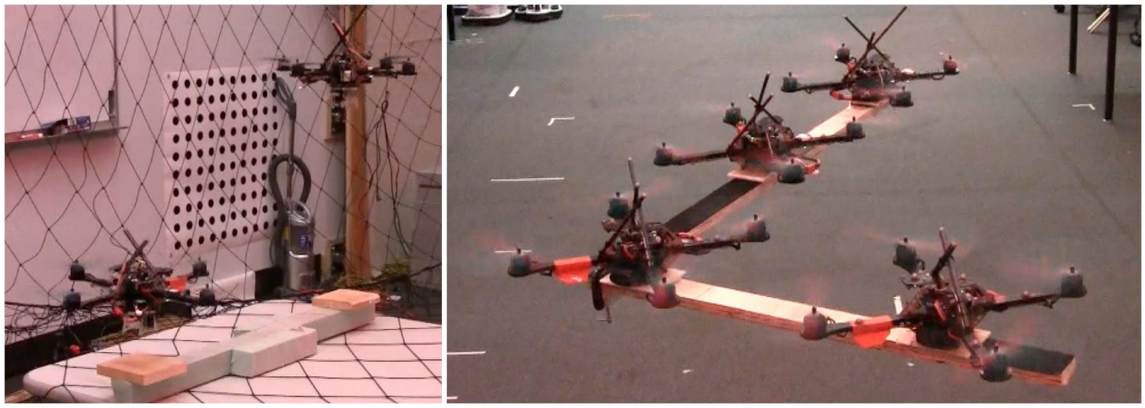


Figure 3.4: Left: Image from an experiment with the gripping mechanism enabling cooperative grasping, manipulation, and transportation. Right: Four quadrotors carrying a payload in the *El* configuration. Videos of the experiments are available at http://tinyurl.com/penndars.

Chapter 4

Online Parameter Estimation

This chapter addresses mechanics, estimation and control for aerial grasping. We present the design of several light-weight, low-complexity grippers that allow quadrotors to grasp and perch on branches or beams and pick up and transport payloads. We then show how the robot can use rigid body dynamic models and sensing to verify a grasp, to estimate the the inertial parameters of the grasped object, and to adapt the controller and improve performance during flight. We present experimental results with different grippers and different payloads and show the robot's ability to estimate the mass, the location of the center of mass and the moments of inertia to improve tracking performance.

4.1 Introduction

In recent years, we have seen extensive research on unmanned aerial vehicles (UAVs) [21, 23, 40, 65]. UAVs offer promises of speed and access to regions that are otherwise inaccessible to ground robotic vehicles. Although most UAV research has focused on fixed-wing aerial vehicles, rotorcrafts such as quadrotors have important benefits. These vehicles can be scaled down to small sizes and can operate in closed, confined environments such as inside buildings. With the smaller size comes increased agility and the

Figure 4.1: Quadrotor assembling part on a cubic structure.

ability to adapt to the environment. Micro UAVs can enter buildings, obtain information in environments that are dangerous for humans, and perch on walls or on joists for monitoring and surveillance.

However, most research on UAVs has typically been limited to monitoring and surveillance applications where the objectives are limited to "look" and "search" but "don't touch." Indeed contact with the environment is avoided as UAVs are primarily used for fly-throughs, such as surveillance and search and rescue. Thus the underlying research focuses on navigation and observation of the environment while minimizing interactions with that environment. By allowing UAVs, specifically quadrotors, to interact with the environment, we get an entire new set of applications.

First, allowing robots to fly and perch on rods or beams allows them to increase the endurance of their missions. Indeed, if perches are equipped with charging stations, robots can recharge their batteries extending their lives substantially. Second, the ability to grasp objects allows robots to access payloads that are unavailable to ground robots. There are

fewer workspace constraints for aerial robots. Third, robots are able to assemble structures and scaffolds of arbitrary height in three dimensions without requiring such special purpose structures as tower cranes.

There are many challenges in aerial grasping for micro UAVs. The biggest challenge arises from their limited payload. While multiple robots can carry payloads with grippers [58] or with cables [61], their end effectors and grippers have to be light weight themselves and capable of grasping complex shapes. Second, the dynamics of the robot are significantly altered by the addition of payloads. Indeed this is also an attraction in assembly because aerial robots can use this to sense disturbance forces and moments. However, for payload transport, it is necessary that the robots be able to estimate the inertia of the payload and adapt to it to improve tracking performance.

This chapter addresses the mechanics, design and control for aerial grasping. The next section presents the background literature in relevant areas. In Sec. 4.3 several grippers we have developed for aerial grasping are described. Section 4.4 explains the dynamic model and control strategy. Section 4.5 discusses parameter estimation methods and their application to the quadrotor dynamics. Finally, we present experimental results in Sec. 4.6.

4.2 Background

Aerial Manipulation Aerial manipulation with full-size helicopters using cables is performed extensively in several applications. The logging industry uses helicopters to transport logs and equipment from areas that trucks cannot access while power companies use this approach to transport and assemble transmission line towers in remote areas. Smaller size rotorcraft have also been used with cables to manipulate objects. The authors of [19] designed a simplified model for modeling single/multiple small size helicopters, objects and ropes. In [61], the authors focused on position and orientation control of a towed object in 6 dimensions by treating multiple quadrotors as anchors whose positions could

be controlled in three dimensions.

In contrast to aerial towing, direct manipulation using grippers can be used to manipulate objects aerially. In [66], the authors develop an elastic constraint contact model to study the flight stability of a small RC helicopter with a compliant gripper in contact with an object and/or the ground. In [53, 58], we used grippers described in Sec. 4.3 to pick up parts and explored multi-robot cooperative transport requiring grasping. None of this previous work, however, considers the changes in flight dynamics due to the grasped payload.

Parameter Estimation Estimation of unknown parameters in dynamic systems is a problem of great interest in robotics. A common approach is to use linear least-squares methods in systems where the unknown parameters are linear in the equations of motion. These approaches have been applied in a number of systems from robotic manipulators [24] to underwater vehicles [22]. We will adopt a similar strategy for estimating the unknown parameters that change when a quadrotor transports payloads.

We then use the estimated parameters in the control laws that determine the behavior of the system. This is similar but not equivalent to the idea of adaptive control. In adaptive control an online parameter estimator provides estimates of certain unknown system parameters at each instant to a control law that is based on those parameters [50]. One example of adaptive control is model reference adaptive control (MRAC) in which the goal is to make the closed loop system perform like a desired model [42]. In all types of adaptive control, the performance and stability of the system are functions of the coupling between the parameter estimator and the control law. In the approach presented here, the parameter estimation is separated from the control law in order to decouple them. The parameter estimation is performed first and then the results of the parameter estimator are used in the control law.

4.3 Gripper Design

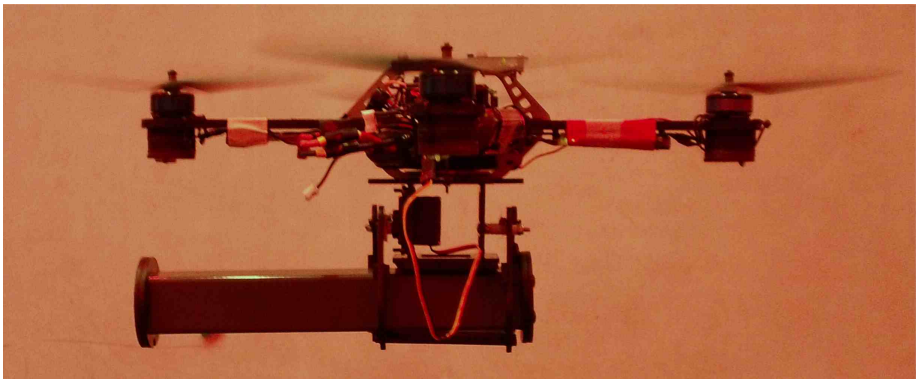
While not the primary focus of this thesis our group has developed a number of grippers to enable quadrotors to pick up and transport payloads. We have used impactive and ingressive methods of grasping. Impactive grippers use solid jaws or fingers in contact with an object to produce the necessary grasping force while ingressive grippers depend on surface deformation, possibly penetration of the surface, to grasp the object [63].

Impactive Gripper

Impactive grippers typically use clamping motions either to enclose or to apply sufficient normal forces and by extension frictional forces to the walls of the object to overcome gravity. In general, these clamping motions are easy to generate using singly actuated mechanisms, which can alleviate weight concerns. Impactive grippers are particularly useful when the gripper can be designed for a small set of parts and orientations such as the gripper shown Fig. 4.2(a) used for assembling structures with quadrotors [53]. Although this impactive gripper was designed to carry specific parts it can also grasp a wider range of objects.

Ingressive Grippers

Impactive grippers require that the object's dimensions are compatible with the gripper's geometry. In contrast, ingressive grippers excel at handling objects that do not have well-defined attachment points which are made of wood, foam, fabric or other porous or deformable materials. The ingressive grippers shown in Fig. 4.3 use the paradigm of penetration into surfaces with metal hooks to attach to surfaces. They use opposing hooks which allows for large normal forces with respect to the penetration force. We have developed an actively engaging gripper shown in Fig. 4.3(a) [57, 58] in which a servo drives the hooks into a surface. We have also developed a passively engaging gripper shown Figs. 4.2(b) and 4.3(b) in which the hooks passively engage in a surface when a surface contacts a



(a) The beam is grasped with an impactive gripper.

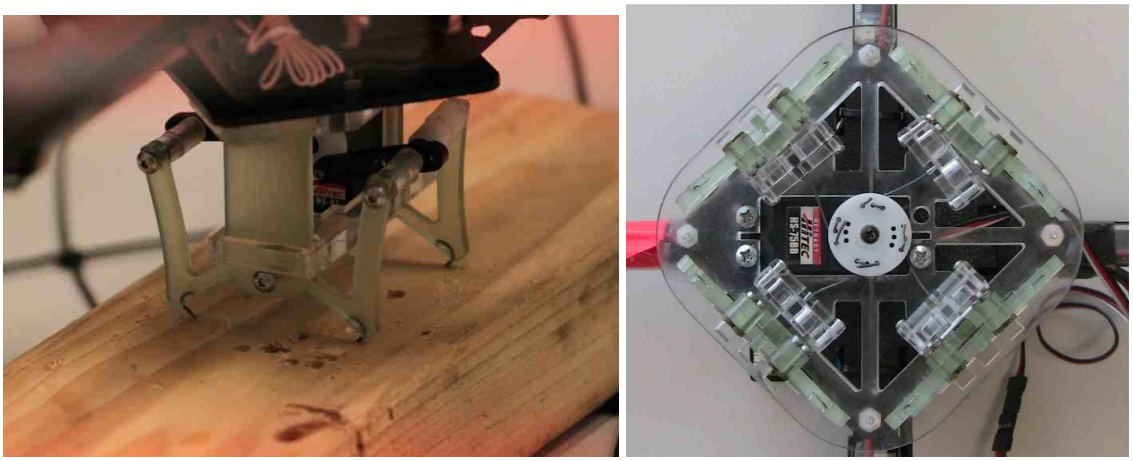
(b) The flat piece of wood is held by an ingressive gripper.

Figure 4.2: Quadrotors carrying objects

trigger pin at the center of the gripper [59].

4.4 System Dynamics and Control

When a quadrotor transports payload the dynamic model of the system changes. We describe the dynamic model and control strategy in this section.



(a) Actively Engaging (b) Passively Engaging

Figure 4.3: Ingressive Grippers

4.4.1 Dynamic Model

The coordinate systems including the world frame, $\mathcal{W}$, and body frame, $\mathcal{B}$, as well as the free body diagram for the quadrotor are shown in Fig. 4.4. We use $Z-X-Y$ Euler angles to define the roll, pitch, and yaw angles (ϕ, θ, and ψ). The rotation matrix from $\mathcal{B}$ to $\mathcal{W}$ is given by ${}^{W}R_{B}$. The angular velocity of the robot is denoted by ω, denoting the angular velocity of frame $\mathcal{B}$ in the frame $\mathcal{W}$, with components p, q, and r in the body frame. Each rotor has an angular speed ω_i and produces a force F_i and moment M_i according to

$$F_i = k_F \omega_i^2, \; M_i = k_M \omega_i^2.$$

Since the motor dynamics are fast compared to the rigid body dynamics and the aerodynamics, we will assume that rotor speeds can be instantly achieved during the controller development. Therefore, the control input $\mathbf{u}$, where u_1 is the net body force and u_2, u_3, u_4 are the body moments, can be expressed in terms of the rotor speeds as

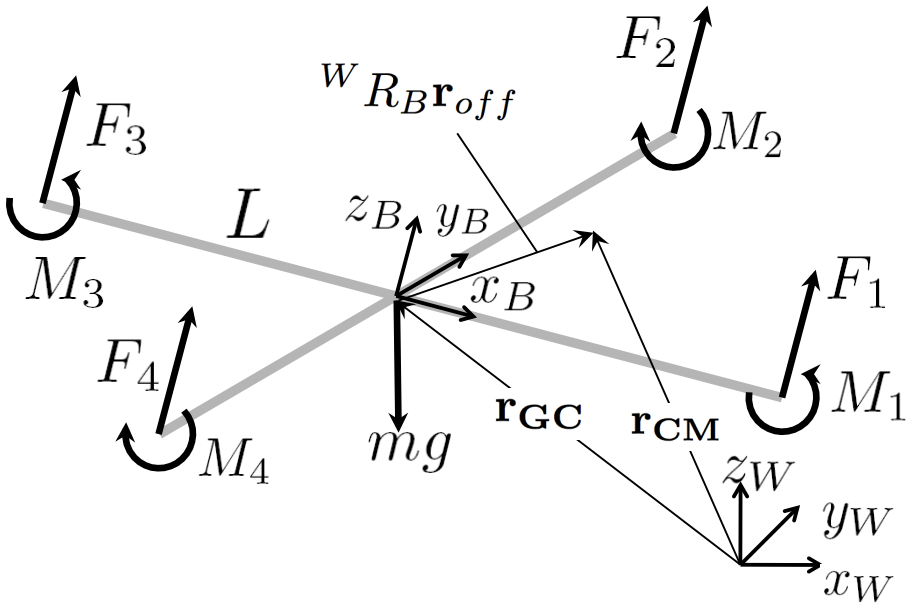


Figure 4.4: Free Body Diagram and Coordinate Systems

$$\mathbf{u} = \begin{bmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F L & 0 & -k_F L \\ -k_F L & 0 & k_F L & 0 \\ k_M & -k_M & k_M & -k_M \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}, \tag{4.1}$$

where L is the distance from the axis of rotation of the rotors to the center of the quadrotor.

The position vector of the center of mass and geometric center $\mathbf{r}_{CM}$ and $\mathbf{r}_{GC}$, respectively, are related according to

$$\mathbf{r}_{CM} = \mathbf{r}_{GC} + {}^W R_B \mathbf{r}_{\text{off}},$$

where $\mathbf{r}_{\text{off}} = [x_{\text{off}}, y_{\text{off}}, z_{\text{off}}]^T$ are offsets in body-frame coordinates.

The forces on the system are gravity, in the $-z_W$ direction, and $u_1 = \sum F_i$, in the z_B direction where F_i is the force from each rotor. Newton's equations of motion governing the acceleration of the geometric center are

$$m\left(\ddot{\mathbf{r}}_{GC} + \alpha \times \mathbf{r}_{\text{off}} + \omega \times (\omega \times \mathbf{r}_{\text{off}})\right) = -mg\mathbf{z}_W + u_1 \mathbf{z}_B, \tag{4.2}$$

where α is the angular acceleration of frame $\mathcal{B}$ in frame $\mathcal{W}$.

The Euler equation for the system is

$$\mathcal{I}_{CM}\dot{\omega} = -\omega \times \mathcal{I}_{CM}\omega + \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} - \mathbf{r}_{\text{off}} \times \begin{bmatrix} 0 \\ 0 \\ u_1 \end{bmatrix} \tag{4.3}$$

where $\mathcal{I}_{CM}$ is the moment of inertia matrix referenced to the center of mass along the $x_B - y_B - z_B$ axes. Note that the net propeller force, u_1, creates a moment in (4.3) since u_1 acts at the geometric center which is not coincident with the center of mass.

4.4.2 Quadrotor Control

We now present a controller to follow specified near-hover trajectories in position of the geometric center and yaw angle, $\mathbf{r}_T(t)$ and $\psi_T(t)$. First, we define the errors on position and velocity as

$$\mathbf{e}_p = \mathbf{r}_{GC} - \mathbf{r}_T, \ \mathbf{e}_v = \dot{\mathbf{r}}_{GC} - \dot{\mathbf{r}}_T.$$

Next we compute the desired force vector:

$$\mathbf{F}_{des} = -K_p\mathbf{e}_p - K_i \int \mathbf{e}_p - K_v\mathbf{e}_v + \hat{m}g\mathbf{z}_W + \hat{m}\ddot{\mathbf{r}}_T,$$

where K_p, K_i, and K_v are positive definite gain matrices and $\hat{m}$ is the estimate of the system mass. Note that this controller effectively treats the two terms $m\alpha \times \mathbf{r}_{\text{off}}$ and $m\omega \times (\omega \times \mathbf{r}_{\text{off}})$ in (4.2) as disturbances that are rejected by the feedback. This is done because these terms are generally small relative to the other terms in practice.

The magnitude of the desired force vector $\mathbf{F}_{des}$ is the first control input, u_1. The desired roll and pitch angles, ϕ_{des} and θ_{des}, are found by determining the attitude which achieves the desired lateral components of $\mathbf{F}_{des}$ for the desired yaw angle ψ_T. The net body frame moments are then computed according to:

$$\begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = K_R \begin{bmatrix} \phi_{des} - \phi \\ \theta_{des} - \theta \\ \psi_T - \psi \end{bmatrix} - K_\omega \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} \hat{y}_{\text{off}} u_1 \\ -\hat{x}_{\text{off}} u_1 \\ 0 \end{bmatrix},$$

where K_R and K_ω are diagonal gain matrices and $\hat{x}_{\text{off}}$ and $\hat{y}_{\text{off}}$ are estimates of the center of mass offsets. Finally we compute the desired rotor speeds to achieve the desired $\mathbf{u}$ by inverting (4.1).

The departure from previous work [56, 62], is that we explicitly include our estimates of the mass of the system and center of mass offsets in the controller, these changes can be significant during payload transport. The offset causes pitching and rolling moments that change with the net thrust produced. Explicitly including this term leads to significantly improved tracking performance as shown in Sec. 4.6.

4.5 Estimation of Inertial Parameters

4.5.1 Method Overview

We use methods which require the unknown parameters, θ, to appear linearly in the equations of motion [41, 50]. We will first outline the proposed parameter adaptation algorithm (PAA) described in [50] which will be used to determine flight parameters. The PAA must be implemented in discrete time so we will consider systems written as

$$y(k+1) = \theta^T \phi(k) \tag{4.4}$$

where k represents the time step, θ is the *parameter vector*, ϕ is the *measurement vector*, and y is the system output. Note that we use Tustin's method to convert the differential equations of motion to discrete time equations. The PAA will try to find an *estimated parameter vector*, $\hat{\theta}$, at each time step. The estimated parameter vector is used to predicted the output of the system as

$$\hat{y}(k+1) = \hat{\theta}(k+1)^T \phi(k) \tag{4.5}$$

The *a priori* prediction error is

$$\epsilon^0(k+1) = y(k+1) - \hat{\theta}(k)^T \phi(k) \tag{4.6}$$

and the *a posteriori* prediction error is

$$\epsilon(k+1) = y(k+1) - \hat{\theta}(k+1)^T \phi(k) \tag{4.7}$$

Given this framework there are a number of methods that can be used to estimate the parameter vector. One of the simpler methods is the instantaneous gradient method. Here the cost function to be minimized is

$$\min_{\hat{\theta}(k)} J(k+1) = \left(\epsilon^0(k+1)\right)^2 \tag{4.8}$$

Using this cost function the parameter vector is modified at each time step to move down the gradient of the cost function according to:

$$\hat{\theta}(k+1) = \hat{\theta}(k) + F\phi(k)\epsilon^0(k+1) \tag{4.9}$$

where F is some positive definite matrix adaptation gain. While simple to implement, the instantaneous gradient method has a few drawbacks. It is difficult to tune the adaptation gain. If the gain is too high then oscillations about the minimum will occur. However, a high adaptation gain is desirable at the beginning in order to ensure fast convergence [50]. Also, the instantaneous gradient method attempts to minimize the error at each time step but does not necessarily lead to the minimization of the error over some finite time span. This is the motivation for the recursive least squares algorithm where the cost function is

$$\min_{\hat{\theta}(k)} J(k) = \sum_{i=1}^{k} \lambda_1^{(k-i)} \left(y(i) - \hat{\theta}^T(k)\phi(i-1)\right)^2 \tag{4.10}$$

and $\lambda_1 \leq 1$ is a *forgetting factor* which is used to weight old data less than new data. Here the cost function is a function of the squared error over t time steps. Note that this cost function can be minimized in a batch fashion or recursively [50].

Batch Least-Squares

The solution to the batch least-squares problem with a forgetting factor of 1 for data collected over k time steps is

$$\theta_{LS} = \hat{R}_{\phi\phi}^{-1} \hat{r}_{\phi y}$$

where

$$\hat{R}_{\phi\phi} = \frac{1}{kp}\sum_{i=1}^{k}\sum_{j=1}^{p}\phi_j(i-1)\phi_j(i-1)^T$$

and

$$\hat{r}_{\phi y} = \frac{1}{kp}\sum_{i=1}^{k}\sum_{j=1}^{p}\phi_j(i-1)y_j(i)$$

Recursive Least-Squares

The method can also be applied recursively so that the estimate changes at each time step as new data is received. In this case the parameter vector estimate is updated according to:

$$\hat{\theta}(k+1) = \hat{\theta}(k) + F(k+1)\sum_{j=1}^{p}\phi_j(k)\epsilon_j^0(k+1)$$

where

$$\epsilon_j^0(k+1) = y_j(k+1) - \hat{\theta}^T(k)\phi_j(k).$$

Here $F(k)$ is a weight matrix that is also recursively updated based of the inversion of:

$$F(k)^{-1} = \lambda_1 F(k+1)^{-1} + \sum_{j=1}^{p}\phi_j(k)\phi_j(k)^T \tag{4.11}$$

Persistent Excitation

Both approaches require ϕ_j to be *persistently exciting*. The measurement vectors, ϕ_j, are persistently exciting if

$$\lim_{k_1\to\infty}\frac{1}{k_1}\sum_{i=1}^{k_1}\sum_{j=1}^{p}\phi_j(i)\phi_j(i)^T > 0.$$

Intuitively, this conditions means that the dynamics are excited sufficiently to identify the unknown parameters. In the batch least-squares method, persistent excitation guarantees that $\hat{R}_{\phi\phi}$ is invertible. In the recursive least-squares method, we see from (4.11) that if the term $\sum_{j=1}^{p}\phi_j(k)\phi_j(k)^T$ becomes non full rank then the adaptive gain, $F(k)$, will tend towards infinity if the forgetting factor, λ_1, is less than one.

4.5.2 Application to Quadrotor Dynamics

These least-squares methods parameter estimation methods are applied to the equations of motion of the quadrotor as described here.

Estimation of payload parameters during hover

When the quadrotor is commanded to hover in place the derivatives of the position and Euler angles are nominally zero. Eliminating these terms from the equations of motion simplifies the parameter estimation method. Consider the z equation of motion during hover:

$$0 = u_1 \left(\mathbf{z}_B \cdot \mathbf{z}_W \right) - mg.$$

Treating m as the unknown parameter and applying the batch least-squares method for k measurements yields

$$\hat{m} = \frac{1}{gk} \sum_{i=1}^{k} u_1(i) \left(\mathbf{z}_B(i) \cdot \mathbf{z}_W \right).$$

Next we consider the Euler equation for the y axis during hover treating the net body force as a known constant, $\bar{u}_1$, where barred coordinates represent the average over the time span of collected data. The unknown offset, x_{off}, found via least-squares is simply an average over collected data:

$$\hat{x}_{\text{off}} = \frac{\bar{u}_3}{\bar{u}_1}.$$

The y axis offset is found similarly. Note the moments of inertia cannot be found during hover because the terms multiplying them are zero. An experimental benefit to this approach is that averages can be computed on the onboard controller and sent back to a control computer at a slower rate than what is required for the method described in Sec. 4.5.2 where applied moments and angular velocities are constantly changing. Note that this method assumes that the robot is not being subjected to any disturbances.

Estimation of payload mass in the presence of disturbances

Newton's equations provide three equations which give information about the mass of the system:

$$\begin{aligned} m\ddot{x} &= u_1 (\mathbf{z}_B \cdot \mathbf{x}_W) + F_x \\ m\ddot{y} &= u_1 (\mathbf{z}_B \cdot \mathbf{y}_W) + F_y \\ m\ddot{z} &= u_1 (\mathbf{z}_B \cdot \mathbf{z}_W) - mg \end{aligned}$$

where F_x and F_y are the lateral aerodynamic disturbance forces.

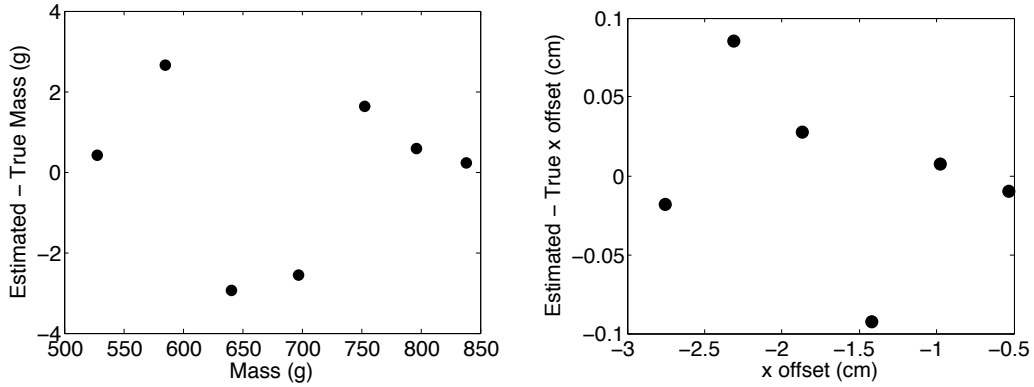
In this formulation the unknown parameter vector is $\theta = [1/m, F_x/m, F_y/m]^T$. Since the mass of the quadrotor and gripper are fixed this method can be used to determine the mass of the payload. The recursive least-squares method can run in real-time and is especially useful for identifying if the quadrotor has successfully picked up or dropped payload.

Estimation of payload inertia

The Euler equations for a system in which the center of mass is offset from the geometric center by some vector $\mathbf{r}_{\text{off}}$ are given by (4.3). We make two assumptions to make these nonlinear equations suitable for our parameter estimation methods: (1) the body axes of the quadrotor are close to the principal axes so its products of inertia are small and (2) the excitation is primarily about one axis so the $\omega \times (I \times \omega)$ term can be neglected. Under these assumptions the equation of motion about the y axis is

$$I_{yy}\dot{q} = u_3 + u_1 x_{\text{off}},$$

where u_1 is the net body force from the props, u_3 is the applied moment along the y axis, and x_{off} is the center of mass offset in the x direction. Here the unknown parameter vector is $\theta = [1/I_{yy}, x_{\text{off}}/I_{yy}]^T$. The least-squares estimators can be applied using flight data for which the pitch dynamics are sufficiently excited. An equivalent procedure is used along for the z and x axes to estimate I_{zz}, I_{xx}, and y_{off}.



(a) Changing Mass

(b) Changing x offset

Figure 4.5: Data for mass and center of mass offset during hover.

4.6 Experimental Results

4.6.1 Estimation of payload parameters during hover

Mass was incrementally added to the system, and the quadrotor was commanded to hover in place for about 10 seconds for each condition. The mass of the system was estimated with the method described in Sec. 4.5.2 as shown in Fig. 4.5(a). The average net thrust was computed by averaging the commanded net thrust sent from the control computer. Note that the error in the prediction is less than 3 grams in all cases.

A mass of 66 grams was added to a quadrotor weighing 687 grams at a number of positions along the x axis in order to offset the center of mass. The theoretical change in the center of mass and the error in estimated center of mass offset are shown in Fig. 4.5(b). Note that the error in the center of mass position estimate is less than 1 mm in all cases. The average net thrust was computed as described above and the average moments were computed by averaging the commanded moments on the onboard microprocessor and sending out these averages at a rate of 10 Hz. Sending out data at a low rate allowed the quadrotor to be computer controlled via the same XBEE module.

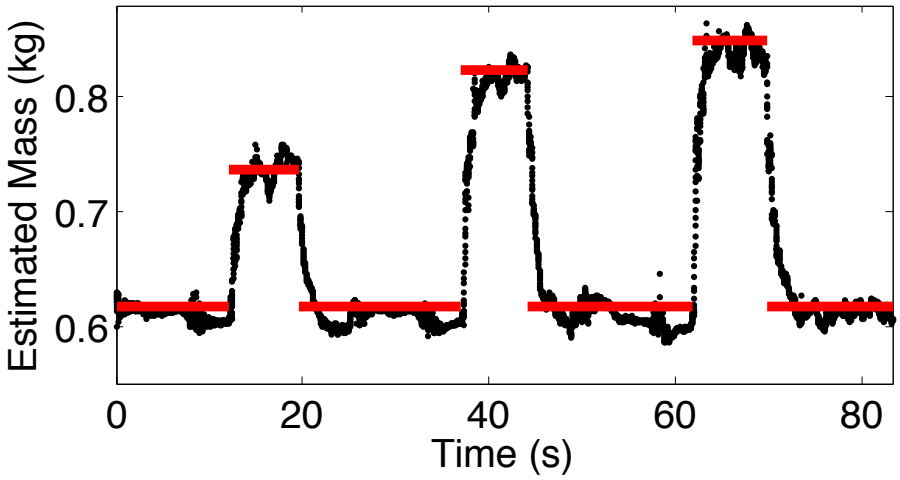


Figure 4.6: Mass estimate (black) and true mass (red) for quadrotor picking up and dropping of three payloads with different weights. The pick up and drop off events can be observed from the change in the true mass.

4.6.2 Estimation of payload mass in the presence of disturbances

The procedure described in Sec. 4.5.2 was applied in a scenario where the quadrotor was commanded to pick up three payloads of different mass and drop them at a prescribed location. For this estimator the accelerations were computed by numerical differentiating the position estimates from VICON. The applied forces were calculated from the commanded net thrust and the orientation sensed from VICON. Since the mass changes in this experiment the recursive least-squares method was used. The estimated mass during this experiment along with the true mass is shown in Fig. 4.6. As expected, the estimated disturbance forces (not shown) are small. For this data a forgetting factor, λ_1, of 0.985 was used at a data collection rate of 100 Hz. Note that for a forgetting factor of 0.985, 63.1% (about $1 - \frac{1}{e}$) of the cost function is based on the last 66 data points. Comparing the response of this filter to a first order system, the approximate equivalent time constant is 0.66 seconds (66 times 0.01 seconds). Choosing a larger forgetting factor makes the mass estimate less noisy but at the cost of responding slower to changes in system parameters.

4.6.3 Estimation of payload inertia

The procedure described in Sec. 4.5.2 was implemented on the quadrotor. For this estimator the angular accelerations were computed by numerically differentiating the angular velocities sensed by the 3-axis rate gyro onboard the quadrotor. The commanded body moments and net force are also available on the onboard controller. In the current experimental setup, it is impossible to both send and receive commands at high rates using a single XBEE module. For this reason in these experiments, the quadrotor was commanded by a pilot using the RC, and the XBEE was used to send data out at a rate of 100 Hz.

This method was applied for the three cases shown in Fig. 4.7. For the last case the point mass is offset from the center of mass by 30 cm in the x direction. For each case, the pilot commanded oscillatory excitation about each of the three axes independently. Data for the pitch excitation for the *Unloaded* and *Meterstick + Mass* cases are shown in Fig. 4.8. Note that the larger moment of inertia can observed from the larger moment in (b) required to produce less angular acceleration. Also, the center of mass offset can be observed from the nonzero mean moment required in (b).

As shown in Table 4.1, the batch least-squares method was applied to estimate the unknown parameters for each combination of case and axis. Note that the y axis offset remain unchanged as expected. For the *Meterstick + Mass* case the x axis offset increases by about 3.2 cm. Theoretically, it should increase by about 2.6 cm. The inertia estimates behave qualitatively as expected. The moment of inertia along the x axis increases slightly for the added payloads. As expected, significant increases in I_{yy} and I_{zz} are observed for the added payloads.

4.6.4 Controller Compensation

The gripper was loaded with a 120 gram horizontal beam as far to one side as possible in the $-x$ direction as shown in Fig. 4.2(a). The mass and center of mass were estimated

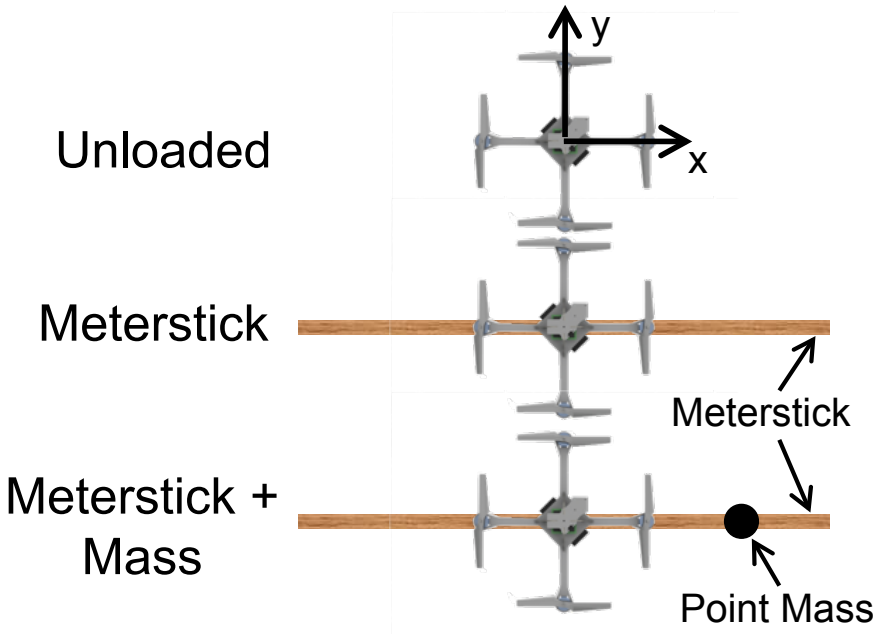
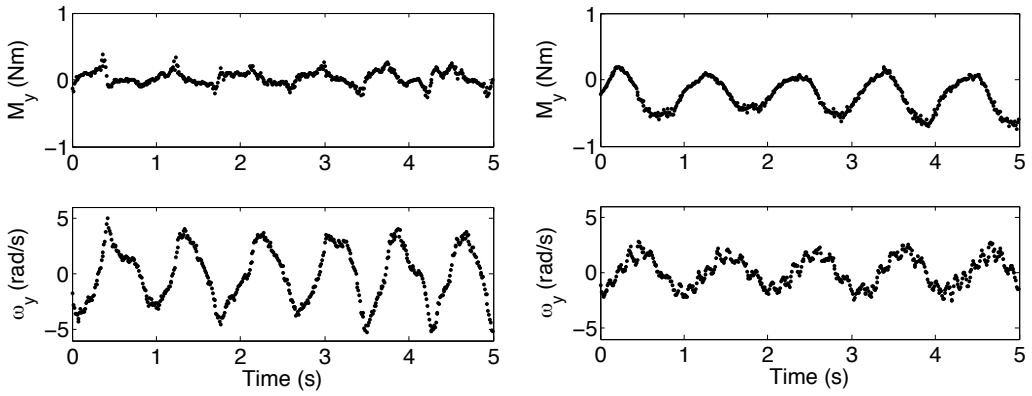


Figure 4.7: Three loading cases tested. The meterstick weighs 125 grams and the point mass weighs 66 grams.

using the methods described in Sec. 4.5.2 using 3 seconds of data collected while hovering in place. The quadrotor was then commanded to fly along a sine wave along the z-axis with amplitude 0.767 meters and frequency 2.77 rad/s using the compensated and uncompensated controller. The compensated controller used the estimated mass and center of mass offset while the uncompensated controller assumed the offset to be zero and the mass to be the mass of the quadrotor and gripper with no part. Plots of the errors in the x and z directions as well as their standard deviations are shown in Fig. 4.9. Note that

Case	I_{xx}	I_{yy}	I_{zz}	x_{off}	y_{off}
Unloaded	3.9	4.4	4.9	-0.30	0.12
Meterstick	5.2	21.5	15.3	-0.26	0.12
Meterstick + Mass	5.8	32.6	21.9	2.89	0.13

Table 4.1: Identified Properties (all inertias in gm^2 and lengths in cm)



(a) Unloaded Case (b) Meterstick + Mass Case

Figure 4.8: Pitch Moment and Pitch Angular Velocity Data

the x performance is in the direction of the center of mass offset. Including the estimated parameters leads to significantly improved tracking performance for this trajectory. This improvement will be significant whenever the trajectory calls for significant changes in the thrust commanded since a changing thrust causes a changing moment which is explicitly cancelled using the controller described here.

4.7 Concluding Remarks

Picking up and transporting payload is a valuable capability for unmanned aerial vehicles. We have demonstrated the ability to grasp and manipulate a number of items using quadrotor helicopters with several grippers that we have designed. When quadrotors transport objects important flight parameters change. Methods for identifying the mass, center of mass offset, and moments of inertia using batch and recursive least-squares methods were described in this paper. A controller was described which explicitly accounts for the estimated mass and center of mass offset. Experimental data was presented to demonstrate all parameter estimation methods. Significant improvement in tracking performance was shown with the inclusion of the estimated parameters.

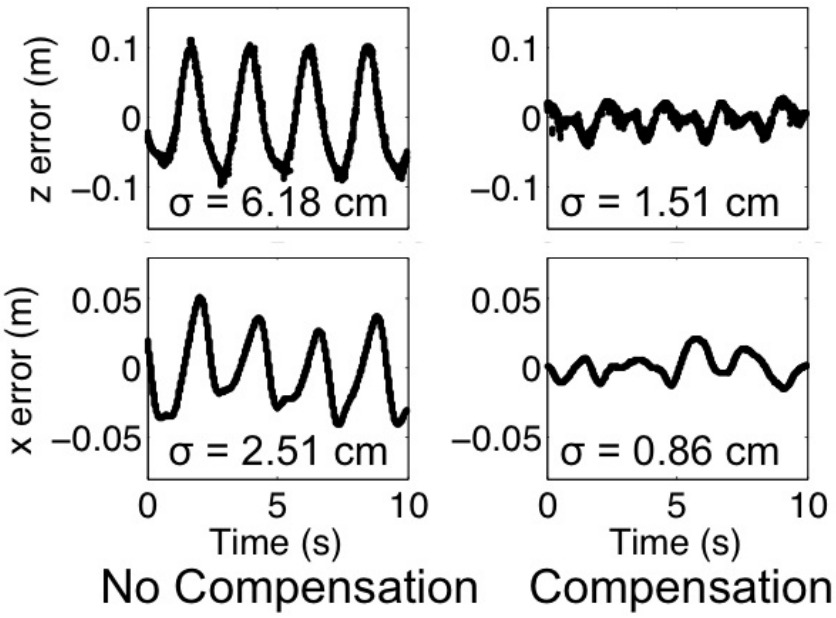


Figure 4.9: Time histories and standard deviations of x and z errors for uncompensated and compensated controller for following the trajectory $z(t) = 0.77\sin(2.77t)m$ with x and y constant for 10 seconds.

Adaptive control, didn't want to put the adaptation too closely in the control loop. Simple straightforward based on a well understood models. Frequency based techniques are available but I didn't really investigate those.

Chapter 5

Trajectory Generation via Sequencing

In Chapter 2 we described several controllers for a quadrotor. Namely, an attitude controller, a hover controller, and a 3D trajectory controller. Even a single controller type can be serve several different purposes. For example, a hover controller with *stiff* gains can be used to hold a position very precisely while a hover controller with *soft* gains can be used to reject large disturbance and recover from a larger range of intial conditions. These three simple types of control can be sequenced in an intelligent way to perform complex tasks. Of course, sequencing is useful for simple experiments. For example, one may wish to fly to a certain position and hover and then fly somewhere else.

The focus in this chapter is on the design of aggressive trajectories for quadrotors via the composition of three simple control modes. In particular we consider the design of dynamically feasible trajectories and controllers that drive a quadrotor to a desired state (position, orientation, linear and angular velocity) in state space. We focus on the development of a family of trajectories defined as a sequence of segments, each with a controller parameterized by a goal state or region in state space. Each controller is developed from the dynamic model of the robot, and then automatically refined through successive experimental trials to account for errors in the dynamic model and noise in the actuators and sensors. We show that this approach permits the development of trajectories and controllers enabling aggressive maneuvers such as flying through narrow, vertical gaps

and perching on inverted surfaces with high precision and repeatability.

Aggressive maneuvers with aerial robots is an area of active research. Exciting results have been demonstrated for perching with fixed-wing aerial vehicles [25, 27] as well perching on inclined surfaces with a small helicopter [18]. A number of groups have demonstrated aggressive aerial maneuvers with small-scale rotorcraft [9, 33, 54]. In this area considerable effort is focused on strategies for generating sequences of controllers that stabilize the robot to a desired state. In [32, 33], Gillula et al. present an optimization-based control design methodology that generates a sequence of stabilizing controllers that drive a robot to a hover state after entering a flipping maneuver. The authors are able to provide guarantees of recovery from a flipping maneuver based on the robot model and present experimental results to validate their approach. Tedrake proposes an optimization-based design methodology with similar guarantees using a guided sparse sampling of the state space and creating sequences of stabilizing controllers that drive the system to a desired state through a sequence of sampled states [75].

Impressive results are also shown using methods based on reinforcement learning or iterative adaptation of the control law. In [54], Lupashin et al. propose a control law with initial parameters for flipping a quadrotor multiple times. The control law is executed many times and corrected after each trial toward the desired performance. A similar strategy is presented in [9, 74], where the authors develop a minimal control law model and refine the model based on data collected from an expert human operator executing the aggressive maneuver. In both cases, system models are based on first principles.

In contrast to the work presented in [9, 32], we address the challenge of designing trajectories in the full, 12-dimensional state space with an underactuated robot with only four actuators. Specifically, we consider goal states parameterized by an arbitrary position, linear velocity, roll, pitch and the derivatives of roll and pitch. We depart from the optimization-based methods described in [32, 33, 75] and incremental search techniques [52] because these methods do not appear to scale to 12 dimensions. The apprenticeship learning methods in [9, 74] require an expert human operator to generate data for

model and control identification and therefore limit the ability of the control law to handle cases not considered *a priori* by the human operator. Indeed, in several cases considered in this paper, it was not possible for a trained human operator to fly the robot in the specified manner.

Similar problems have also been addressed using model predictive control (MPC) [46, 79]. With these approaches, guarantees of convergence are only available when the linearized model is fully controllable [79] or if a control Lyapunov function can be synthesized [43]. As such it appears to be difficult to directly apply such techniques for the trajectory generation of a quadrotor

5.1 Control

Our basic approach relies on the development of three controllers (described next), and the composition of these controllers (described in the next section). Each controller is tuned automatically to obtain the desired performance using a combination of off-line system identification techniques and on-line parameter identification techniques [50, 59]. These controllers are have the following objectives:

1. Attitude Control: Driving the robot to a desired roll, pitch and yaw, while maintaining a constant nominal thrust in the body-fixed frame;

2. Hover Control: Achieving and maintaining a desired three-dimensional position and yaw angle;

3. 3-D Path Following: Controlling the center of mass to follow a prescribed three-dimensional path while maintaining a specified, possibly varying yaw angle along the path, with a specified velocity (and acceleration) profile along the path.

In this section, we present methods used for all three controllers. The controllers for hover and path following rely on small angle assumptions. In other words, we assume small deviations from the nominal hover position. This assumption is relaxed in [55].

5.2 Trajectory Generation via Sequential Composition

In the previous section we described three controllers. Each controller can be naturally thought of as a mode in a hybrid system [14]. Each mode has a number of underlying *parameters* and *set points* that determine its behavior as shown in Table 5.1. The parameters determine the feedback performance and the set points govern the feed-forward component that determine the orientation, position, or path to be tracked. Changing the parameters of a controller allows us to use it to serve different purposes. For example, a hover controller with stiff gains can be used to hold a position very precisely while a hover controller with soft gains can be used to reject large disturbance and recover from a larger range of initial conditions.

These three simple types of controllers can be sequenced in an intelligent way to perform complex tasks. The transitions between modes of operation can be time-triggered or event-triggered. A time-triggered transition between modes arises when it is necessary to spend a specified amount of time in a particular mode (for example, hover) before exiting out of the mode (for example, to fly to a new position). An event-triggered transition arises when a specified region in state space is reached or a specified condition on state and input variables is satisfied.

In this section, we will describe two applications of the basic idea of mode switching for trajectory generation. In the first example, we will develop a method for controlling to any position in space with a specified range of velocities and pitch angles. This enables flying through windows at different specified angles and perching on vertical or inclined surfaces. In the second example, we demonstrate approaches to robustly recover from failed perching attempts.

	Attitude Control	Hover	3-D Path Following
Parameters for feedback control	k_p, k_d for ϕ, θ, ψ	k_p, k_d for ϕ, θ, ψ k_p, k_i, k_d for x, y, z	k_p, k_d for ϕ, θ, ψ k_p, k_i, k_d for x, y, z
Set points and feed-forward control	$\phi^{des}, \theta^{des}, \psi^{des}$ $p^{des}, q^{des}, r^{des}$	$\mathbf{r}_T, \psi_T$	$\mathbf{r}_T(\xi), \dot{\mathbf{r}}_T(\xi), \ddot{\mathbf{r}}_T(\xi), \psi_T(\xi)$

Table 5.1: Parameters underlying feedback control and set points and desired paths for feed-forward control for each of the three control modes

5.2.1 Sequence for Aggressive Trajectories

Trajectory Description

We design a sequence of controllers to reach a goal state, G, with a specified position, $\mathbf{r}_G$, velocity, $\mathbf{v}_G$, yaw angle, ψ_G, and pitch angle, θ_G, with zero angular velocity and roll angle. The sequence consists of 5 phases:

- Phase 1 - hover control (stiff) to a desired position;
- Phase 2 - 3-D path following toward a desired position, $\mathbf{r}_L$, and yaw angle, ψ_G;
- Phase 3 - attitude control to desired pitch angle, θ_G, yaw angle, ψ_G, and zero roll;
- Phase 4 - attitude control to zero pitch angle, yaw angle, ψ_G, and zero roll;
- Phase 5 - hover control (soft) to a desired position.

The sequence can also be illustrated as a hybrid automaton in Fig. 5.1.

The yaw angle is controlled to be a specified ψ_G during all phases. In Phase 2, the robot controls along a $3-$D trajectory to build up momentum and reach a commanded velocity $\mathbf{v}_L$ at a *launch point*, $\mathbf{x}_L$. Phase 3 is initiated when the quadrotor passes the plane perpendicular to the desired velocity at the launch point. In Phase 3, the robot's attitude is controlled to a commanded pitch angle and a roll angle of zero. Note that during Phase 3, a constant net thrust is commanded. Phase 4 and 5 are recovery phases. In Phase 4, we

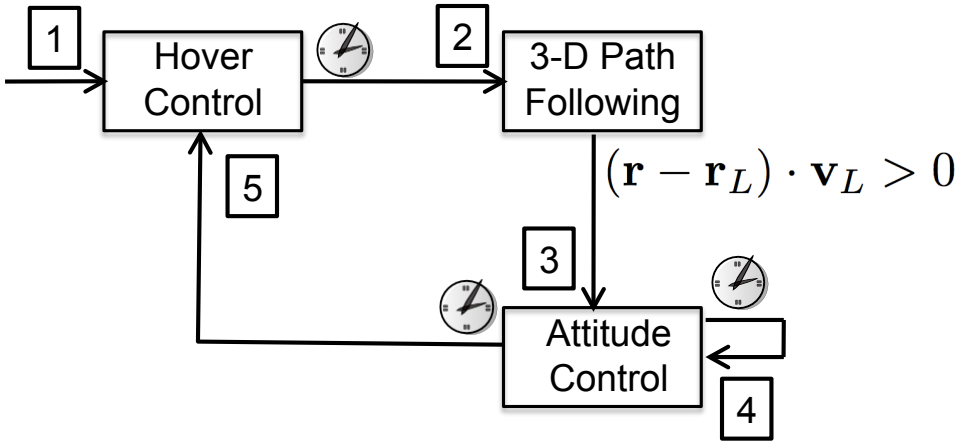


Figure 5.1: Composition of controllers to execute an aggressive trajectory to a goal state. The clocks at the base of the arrows represent time-triggered transitions whereas mathematical conditions represent event-triggered transitions. The phase number is labeled near the tip of each arrow.

use the attitude controller to control to a pitch and roll angle of zero. In Phase 5, a soft hover controller is used to stabilize to a position in space with zero velocity.

Initial Parameter Selection

The pitch tracking controller used in Phase 3 is tuned so that the settling time is approximately T_s seconds and the response is close to critically damped in response to a step input. For this reason, we design the system to reach the state G, T_s seconds after starting Phase 3. The position of the launch point, $\mathbf{r}_L$, and the velocity vector at the launch point, $\mathbf{v}_L$, necessary to achieve the desired state G are found via backwards integration of the equations of motion (3.3) from G to L. Here we assume the pitch angle tracks a trajectory with a critically damped response characterized by a settling time of T_s seconds, the yaw angle is ψ_G, the roll angle is zero, and the net thrust is equal to the desired thrust. During Phase 2, the quadrotor starts at hover at $\mathbf{r}_S$ and follows the line segment from $\mathbf{r}_S$ to $\mathbf{r}_L$ with commanded velocity $\mathbf{v}_L$. So the start position, $\mathbf{r}_S$, is simply a chosen distance, l, in the direction of $-\mathbf{v}_L$ from $\mathbf{r}_L$ according to

$$\mathbf{r}_S = \mathbf{r}_L - l\frac{\mathbf{v}_L}{\|\mathbf{v}_L\|}. \tag{5.1}$$

Parameter Adaptation

The real quadrotor does not perform exactly the same as the model. The system will not reach the exact launch point, $\mathbf{r}_L$, or have the exact desired velocity, $\mathbf{v}_L$, at the launch point. Additionally the quadrotor will not have an exactly zero pitch angle at the launch point and the angle performance will not be exactly critically damped with a settling time of T_s seconds. These deviations are caused by air drag, rotor dynamics, and actuator saturation limits. It is difficult to accurately model these effects so we iterate on experimental trials to achieve the goal state G using a form of iterative learning [15].

In this approach, starting with the initial parameter selection, we iterate n_a times on the commanded pitch angle during Phase 3. We let the commanded pitch angle at iteration k be θ_C^k. We let θ_{act}^k be the actual pitch angle achieved T_s seconds after entering Phase 3 during iteration k and update with a step size parameter, $\gamma_\theta \leq 1$, as follows

$$\theta_C^{k+1} = \theta_C^k + \gamma_\theta(\theta_G - \theta_{act}^k) \tag{5.2}$$

Since the pitch angle achieved during Phase 3 is not strongly affected by the velocity commanded during Phase 2, we can iterate on the commanded velocity without significantly affecting the pitch angle. We use the same strategy as for pitch angle and iterate n_v more times on velocity:

$$\mathbf{v}_C^{k+1} = \mathbf{v}_C^k + \gamma_v(\mathbf{v}_G - \mathbf{v}_{act}^k) \tag{5.3}$$

Finally, we run the final parameters for n_x trials and let $\bar{\mathbf{r}}_{act}$ be the average position achieved T_s seconds after entering Phase 3 during the trials. The entire trajectory is then simply shifted by the difference between the desired position, $\mathbf{r}_G$, and the actual position, $\bar{\mathbf{r}}_{act}$, as follows:

$$\mathbf{r}_L = \mathbf{r}_L + (\mathbf{r}_G - \bar{\mathbf{r}}_{act})$$

$$\mathbf{r}_S = \mathbf{r}_S + (\mathbf{r}_G - \bar{\mathbf{r}}_{act})$$

Note that the gains for all the controllers are designed ahead of time. During parameter adaptation only the feed-forward control parameters, the commanded pitch angle and the three components of the commanded velocity are adapted in this iterative learning process.

5.2.2 Sequence for Robust Perching

The mode-switching sequence described in the previous section can be used for perching on surfaces at different angles. In this section, we show how this approach can be used to perch on vertical surfaces and recover from failed attempts.

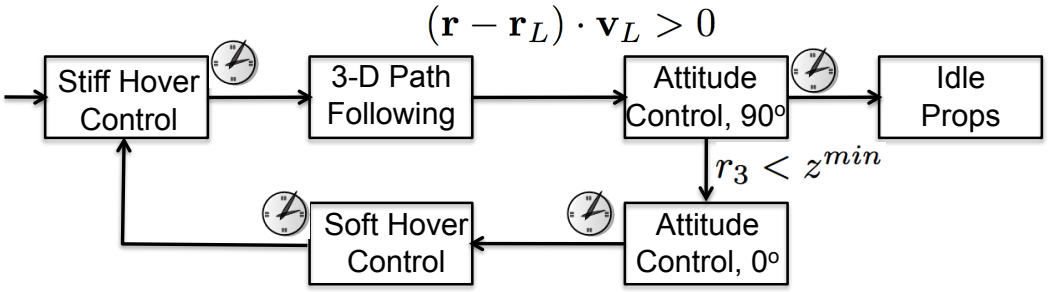


Figure 5.2: Control strategy for robust perching on a vertical wall. The clocks at the base of the arrows represent time-triggered transitions whereas mathematical conditions represent event-triggered transitions.

In the 3-D path following mode, the robot controls along a $3-$D line segment at a commanded velocity, $\mathbf{v}_L$, towards a launch point, $\mathbf{r}_L$. Attitude control to 90° is initiated when the quadrotor passes the plane perpendicular to the desired velocity at the launch point after which the robot's attitude controls to a commanded pitch angle of 90° and a roll angle of zero. At this point if the quadrotor successfully attaches to the vertical surface, the propellers are controlled to idle.

If the quadrotor misses perching on the wall then steps must be taken for the quadrotor to recover. First we must detect that the quadrotor has failed to attach to the wall. We do this by sensing that the quadrotor has dropped below the height of the perch, z^{min}. After sensing that the quadrotor has missed the perch, it switches to the attitude control to 0° mode for a specified time and then to hover at a distance away from the wall with a controller with soft gains and a large basin of attraction. After recovering from the failed perching attempt, the quadrotor tries the perching sequence again until it succeeds.

5.3 Sequence for Robust Landing

Here we describe a different sequence of the controllers designed to robustly land on a small horizontal surface. We assume the position and velocity of the quadrotor are available for the feedback controller and the x and y position of the landing location can be sensed with zero mean error. We do not require the exact z height of the landing location to be sensed as it is detected from a change in quadrotor performance. The sensing of an event or the passing of a specified amount of time triggers a change in the controller mode. A diagram illustrating the control strategy is shown in Figure 5.3.

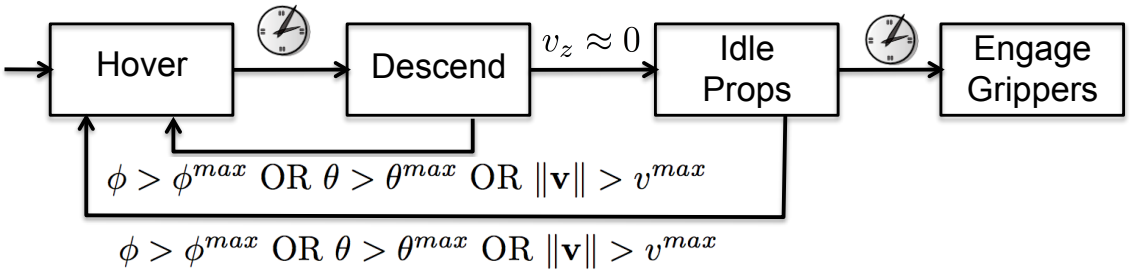


Figure 5.3: Control strategy for robust landing. The clocks at the base of the arrows represent time-triggered transitions whereas mathematical conditions represent event-triggered transitions.

First, the quadrotor is controlled to **Hover** above the desired landing location. Next it is commanded to **Descend** at a specified velocity. While in the **Descend** mode the quadrotor waits to sense an event before transitioning to the next mode. If an error is sensed the quadrotor is controlled to hover at its original location. If a z velocity close to zero is sensed then the quadrotor has likely made contact with the surface and the quadrotor enters the **Idle Props** mode. If an error is sensed in this mode the quadrotor is commanded to **Hover** at its original location. If no error occurs in some amount of time then the grippers are engaged.

The concept of an error is an important part of this control strategy. Here we sense an error by checking if the roll angle, pitch angle, or velocity are above the threshold values ϕ^{max}, θ^{max}, and v^{max}. These conditions are designed to catch situations when the

quadrotor is falling off the desired landing location or when the quadrotor is in free fall because it switched into the **Idle Props** mode when it was not actually on a surface.

5.4 Experiment Design and Implementation Details

In this work we present a systematic approach for designing trajectories and associated controllers that permit aggressive maneuvers with quadrotor robots. We consider four experimental scenarios:

- flying through a vertical opening at varying angles;
- flying downward through a horizontal opening;
- flying upward through a horizontal opening;
- perching on a target at varying angles.

Note that adhesion during perching is achieved by placing Velcro® on the underside of the quadrotor and on a target location. Graphics depicting the four scenarios are shown in Fig. 5.4. The first and last scenarios include cases at various angles. For each of the cases, the quadrotor executed 15 trials of the trajectory after completing 2 iterations of (5.2) and 4 iterations of (5.3). We report the results of these experiments in Sect. 5.5.

5.5 Results

5.5.1 Aggressive Trajectories

Nine cases of the four scenarios shown in Fig. 5.4 were tested. All of the cases use a desired yaw angle, ψ_G, of 90°. The details of these cases are shown in Table 5.2. In all

Case	Description	v_G (x,y,z) (m/s)
1	Vertical Window at $45°$	(2, 0, 0)
2	Vertical Window at $60°$	(2, 0, 0)
3	Vertical Window at $75°$	(2, 0, 0)
4	Vertical Window at $90°$	(2, 0, 0)
5	Down Through Horizontal Window at $90°$	(0, 0, -1.5)
6	Up Through Horizontal Window at $90°$	(0, 0.4, 2.2)
7	Perch at $60°$	(0, 0.8$\cos(30)$, -0.8$\sin(30)$)
8	Perch at $90°$	(0, 0.8, 0)
9	Perch at $120°$	(0, 0.8$\cos(30)$, 0.8$\sin(30)$)

Table 5.2: The 9 test cases used to generate the experimental results for this paper.

of the vertical window cases (1-4) the desired velocity is 2 m/s through the window and zero in the other directions. This speed is large enough for the quadrotor to coast through the window at the desired angle. For descending though the horizontal window (case 5), the desired downward speed is 1.5 m/s and zero in the other directions. This speed has to be small enough to give the quadrotor time to recover after passing through the window. Ascending through the horizontal window (case 6) is the most difficult case because the quadrotor must achieve enough vertical speed to coast upward through the window. For each of the perching cases (7-9), the desired velocity was set to be 0.8 m/s normal to the perching surface. This speed is large enough to guarantee adhesion given proper alignment of the quadrotor to the perching surface. Representative images from cases 4, 5, and 9 are shown in Fig. 5.5.

The performance of the iteration scheme for a representative case (case 8) in Fig. 5.7. After the first iteration the initial angle error drops from $10°$ to less than $2°$ for the rest of the iterations. The velocity adaptation begins after iteration 3. The velocity error improves significantly in iteration 4 and continues to stay low for the remainder of the iterations.

After performing the iteration scheme for each of the cases, 15 trials with the same parameters are run for each case. A summary of the data collected from the 15 trials for all of the cases is shown in Figs. 5.8 and 5.9(b). For a representative case (case 3)

the standard deviation on the achieved angle is 0.9° and the standard deviations on the velocity and position are around 8 cm/s and 2 cm, respectively, for all axes.

5.5.2 Robust Perching

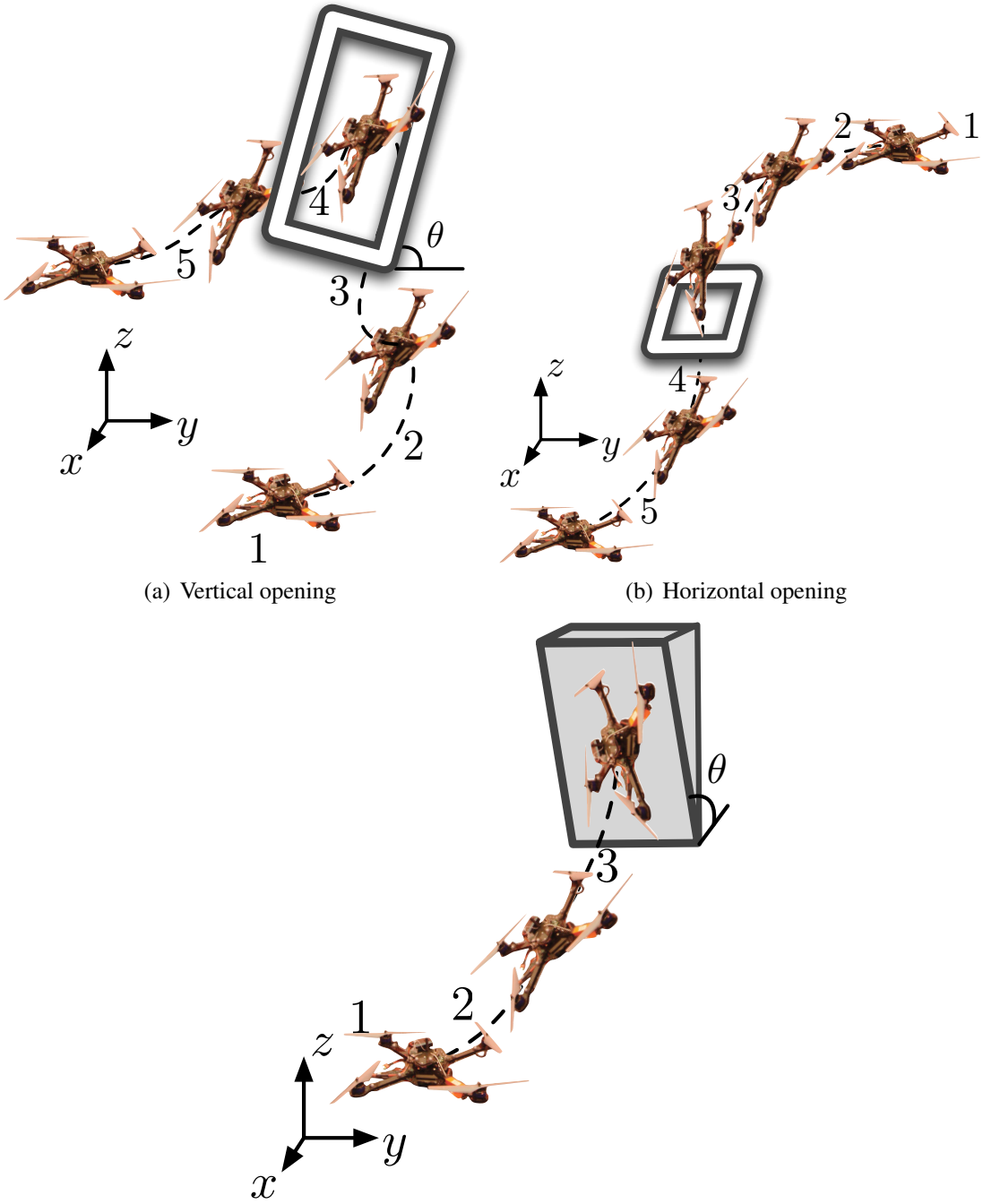
The quadrotor was commanded to perch on a vertical surface for ten trials. In order to guarantee failure no mechanism was placed on the quadrotor to enable attachment. For all ten trials the quadrotor recovered to a stable hover. A representative trial is shown in Figure 5.9(a). The quadrotor launches to the wall then controls to a pitch angle of 90°. After failing to attach to the wall the quadrotor is controlled to a pitch angle of 0° and then finally to a stable hover.

5.5.3 Robust Landing

The quadrotor was commanded to land on the wide side of a two-by-four ten times. All ten trials were successful and the standard deviations in the x and y landing locations were both less than 1 cm. The two-by-four was then displaced in various directions from the quadrotor's target location for 40 trials. On all 40 trials the quadrotor successfully recovered to a stable hover. A representative trial is shown in Figure 5.10. The quadrotor descends until it hits the board and then switches to the idle propellers mode. The quadrotor begins to fall off the board and enters the recovery hover mode after the roll angle exceeds 10°. The quadrotor then recovers to its original position and is ready to perform another attempt at landing.

5.6 Conclusion

Here we studied the problem of designing dynamically feasible trajectories and controllers that drive a quadrotor to a desired state in state space. We focus on the development of a family of trajectories defined as a sequence of segments, each with a simple controller parameterized by a few gains and a goal state. Each controller is developed from the dynamic model of the robot, but is deliberately kept simple with a relatively few parameters to permit iterative refinement through successive experimental trials. These iterations allow us to account for the inevitable errors in the dynamic model and limitations of the actuators and sensors. Four scenarios are tested experimentally as considered by nine case studies with fifteen trials of each case study. The scenarios include flying through narrow, vertical and horizontal openings and perching on an inverted surface. We show that our approach results in repeatable and precise control along trajectories that demand velocities and accelerations that approach the limits of the vehicle's capabilities.



(a) Vertical opening

(b) Horizontal opening

(c) Perching

Figure 5.4: The four experimental scenarios considered in this work. Flying downward and upward through a horizontal opening are both shown in Fig. 1.5.4(b), where the stages of the robot's progression are reversed for the latter. Note that in Figs. 1.5.4(a) and 1.5.4(c), θ denotes the varied window and perching orientation.

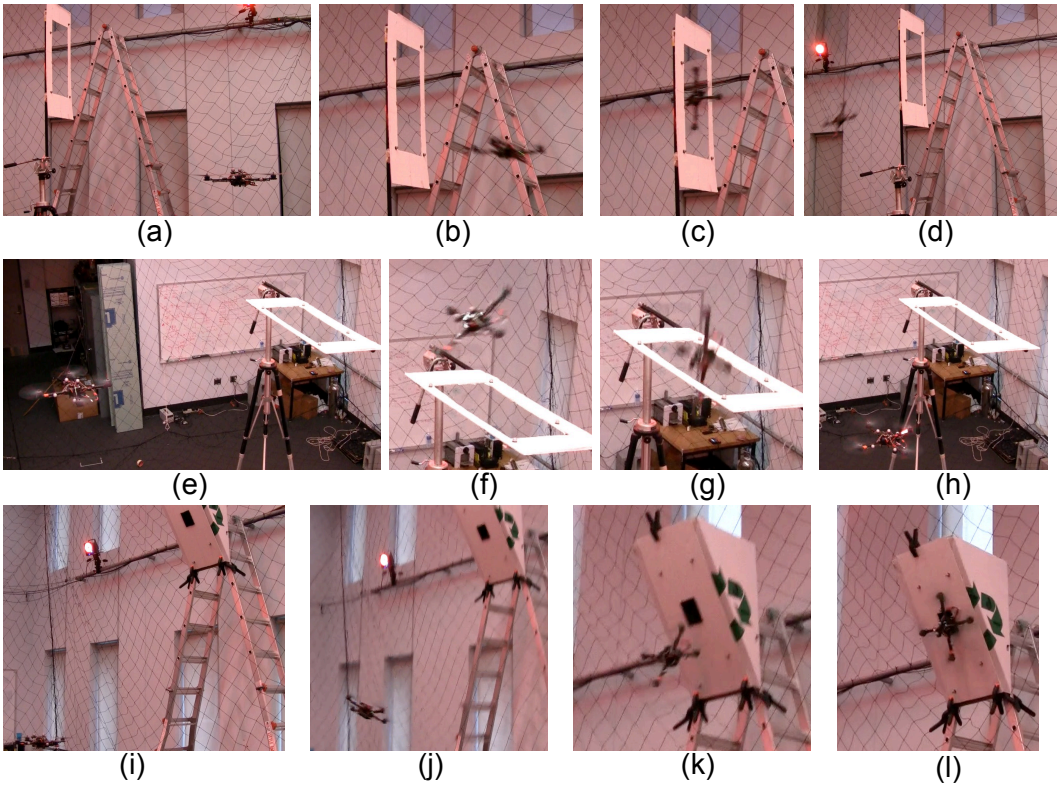
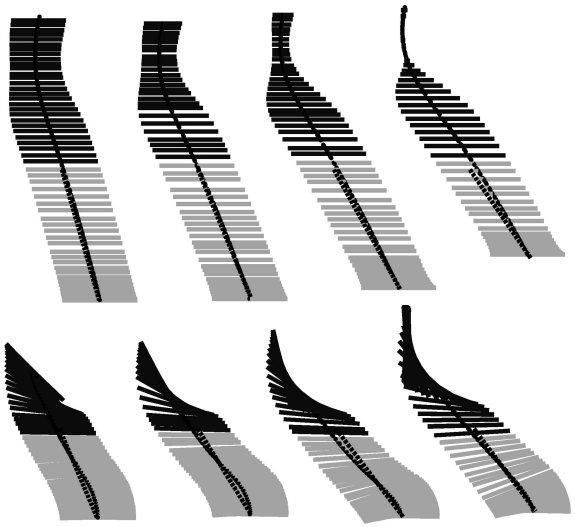
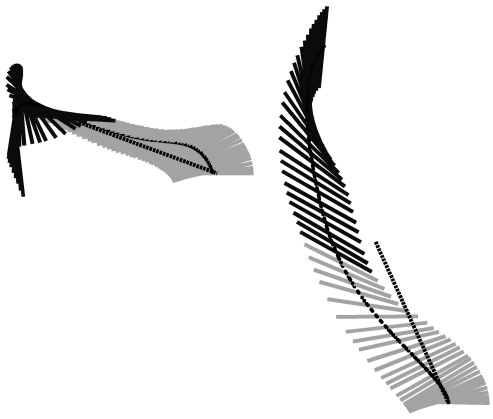


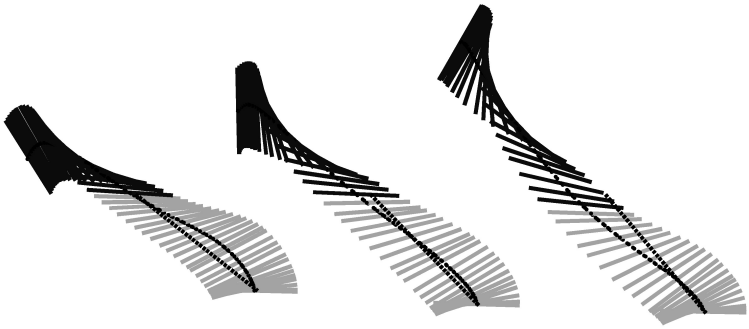
Figure 5.5: Images from representative experimental trials. Figures 5.5(a-d) present a trial of the quadrotor passing through a vertical window at 90° (case 4). Figures 5.5(e-h) show the quadrotor descending through a horizontal window (case 5). Figures 5.5(i-l) show the quadrotor perching on a 120° surface (case 9). Videos of the experiments are available at http://tinyurl.com/quadrotorcontrol.



(a) Vertical Window, Cases 1-4, Top Views (above) and Front Views (below)

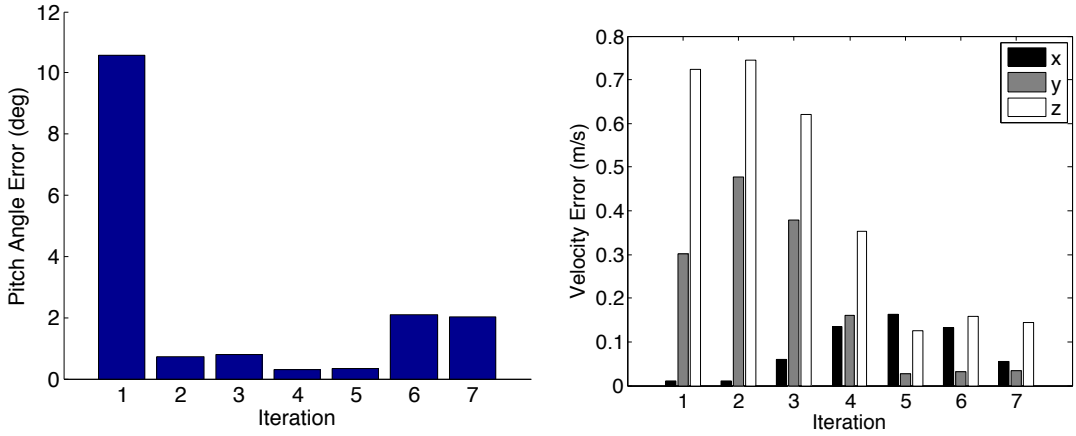


(b) Horizontal Window, Cases 5 and 6, Side Views



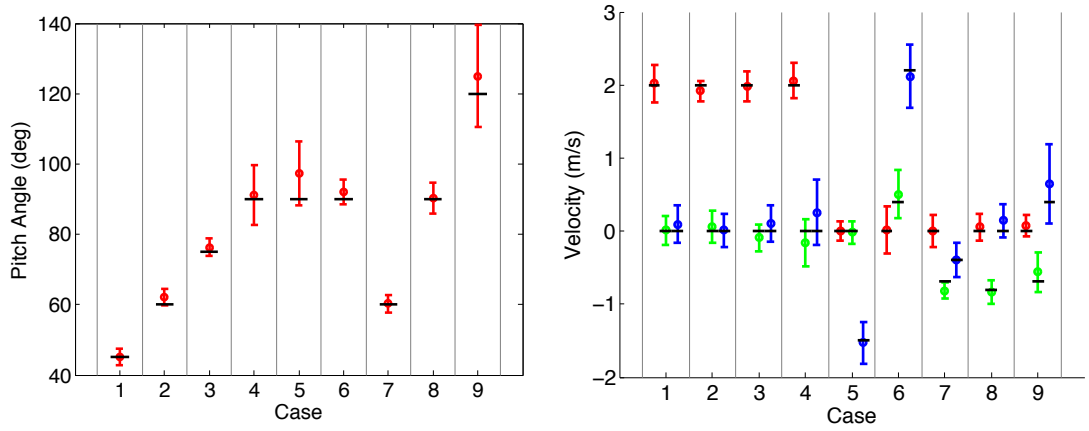
(c) Perching, Cases 7-9, Side Views

Figure 5.6: Experimental data for first two phases for each tested case. The lines represent the orientation of the quadrotor. Phase 2 is shown with light gray lines and phase 3 is shown with dark gray lines. The dotted straight line illustrates the line segment that the quadrotor is commanded to follow during phase 2.



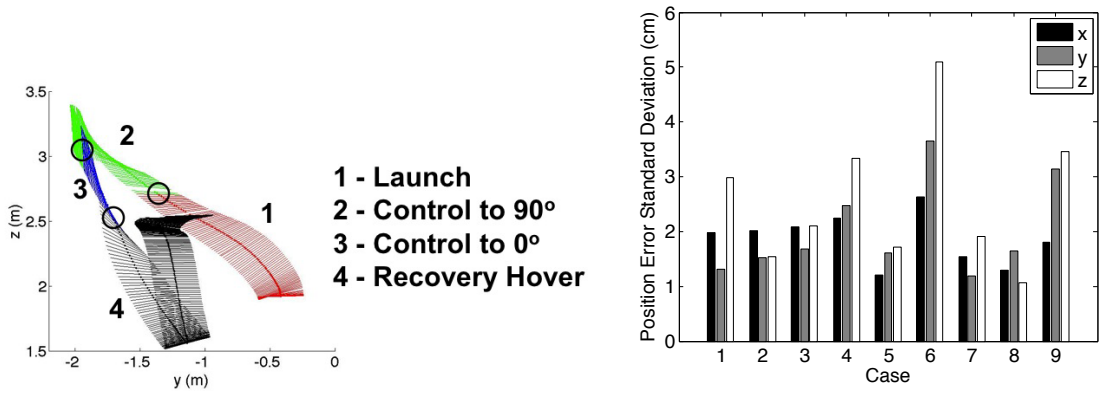
(a) (b)

Figure 5.7: Pitch angle and velocity improvement after iterating when perching at $90°$ (case 8).



(a) (b)

Figure 5.8: Mean pitch angles (a) and velocities (b) for 15 trials of each case. The error bars represent three standards deviations and the solid bars are the desired pitch angles and velocities. In (b) the x, y, and z velocities are shown in the left, middle, and right positions, respectively, for each case.



(a) (b)

Figure 5.9: (a) Recovery from a failed perching attempt on a vertical surface. The circles represent the transitions between control modes. (b) Standard deviations on goal positions for 15 trials for each case.

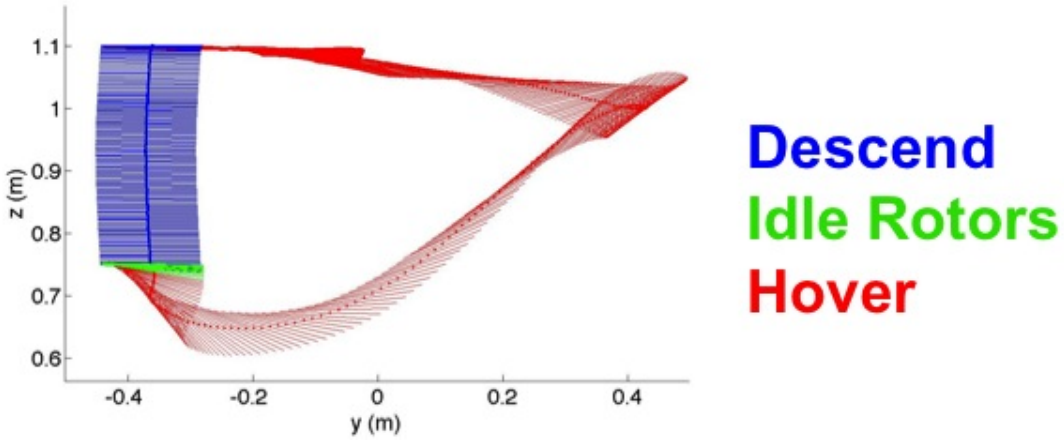


Figure 5.10: Recovery from a failed landing attempt on a horizontal surface.

Chapter 6

Minimum Snap Trajectory Generation using Piecewise Polynomials

In this chapter we address the controller design and the trajectory generation for a quadrotor maneuvering in three dimensions in a tightly constrained setting typical of indoor environments. In such settings, it is necessary to allow for significant excursions of the attitude from the hover state and small angle approximations cannot be justified for the roll and pitch. We develop an algorithm that enables the real-time generation of optimal trajectories through a sequence of 3-D positions and yaw angles, while ensuring safe passage through specified corridors and satisfying constraints on velocities, accelerations and inputs. A nonlinear controller ensures the faithful tracking of these trajectories. Experimental results illustrate the application of the method to fast motion (5-10 body lengths/second) in three-dimensional slalom courses.

6.1 Introduction

Our focus in this chapter is on the modeling, controller design, and trajectory generation for quadrotors. Most of the work in this area uses controllers that are derived from linearization of the model around hover conditions and are stable only under reasonably small

roll and pitch angles [40]. Exploring the full state space using reachability algorithms [33], incremental search techniques [52] or LQR-tree-based searches [75] is impractical for a dynamic system with six degrees of freedom. Some work in this area has addressed aerobatic maneuvers [9, 33, 54, 56]. However, there are no stability and convergence guarantees when the attitude of the rotor craft deviates substantially from level hover conditions. While machine learning techniques have been successful in learning models using data from human pilots [9] and in improving performance using reinforcement learning [54], these approaches do not appear to lend themselves to motion planning or trajectory generation in environments with obstacles. Similar problems have been addressed using model predictive control (MPC) [46, 79]. With these approaches, guarantees of convergence are only available when the linearized model is fully controllable [79] or if a control Lyapunov function can be synthesized [43]. As such it appears to be difficult to directly apply such techniques to the trajectory generation of a quadrotor.

In this paper, we address the controller design and the trajectory generation for a quadrotor maneuvering in three-dimensions in a tightly constrained setting typical of indoor environments. In such settings, it is necessary to develop flight plans that leverage the dynamics of the system instead of simply viewing the dynamics as a constraint on the system. It is necessary to relax small angle assumptions and allow for significant excursions from the hover state. We develop an algorithm that enables the generation of optimal trajectories through a series of keyframes or waypoints in the set of positions and orientations, while ensuring safe passage through specified corridors and satisfying constraints on achievable velocities, accelerations and inputs.

Many quadrotor controllers operate near hover and rely on small angle assumptions for roll and pitch. Several groups have pushed model rotorcrafts beyond these small angles and created exciting aerobatic flights [9, 33, 54, 56]. However, during the large angle portion of these trajectories there is no position control [33, 54, 56] or position control is not precise enough for obstacle avoidance [9]. Here we are interested in using large pitch and roll angles for the purpose of controlling precisely along aggressive trajectories.

In this work, we develop a flexible and powerful trajectory generation method for quadrotors. The goal is to generate and then control along trajectories through cluttered environments or difficult scenarios. The method can be used to generate optimal trajectories through a series of waypoints and also allows safe corridors of different widths to be defined between waypoints. In addition to position, velocity, and acceleration constraints the method is able to incorporate constraints on angular velocities, thrust, and moments required. For this reason, it can be used to generate trajectories that push the limits of the capabilities of the quadrotor. However, it can also generate trajectories that satisfy any arbitrary constraints on trajectory "safeness" (e.g., angular velocity constraints, maximum roll or pitch angles). The method has the additional advantage it requires no expert human operator to train from as in the apprenticeship methods [9] or any experimental training as in iterative methods [54, 56].

The organization of the chapter is as follows. First, we present a model for the quadrotor dynamics and show that the quadrotor dynamics with four inputs is differentially flat and use this as a tool for trajectory generation. Next, a nonlinear controller that does not rely on small angle assumptions on the roll and pitch angle is described. In Section 6.2, the trajectory generation method is described. Finally, the experimental results of a quadrotor flying through a static environment with three narrow gaps, flying through a thrown hula hoop, and catching a bouncing ball are presented.

6.2 Trajectory Generation

Here we describe the trajectory generation method presented in [55]. We build on the results of Ch. 2 and consider trajectories in the flat output space of the form of (2.18). We parameterize the tracking trajectories, $\sigma_T(t)$, using suitable basis functions in $\mathbb{R}^3 \times SO(2)$. In particular, it is convenient to write them as piecewise polynomial functions of order n over m time intervals. The description of the trajectory is as follows.

$$\sigma_T(t) = \begin{cases} \sum_{i=0}^{n} \sigma_{Ti1} t^i & t_0 \leq t < t_1 \\ \sum_{i=0}^{n} \sigma_{Ti2} t^i & t_1 \leq t < t_2 \\ \vdots & \\ \sum_{i=0}^{n} \sigma_{Tim} t^i & t_{m-1} \leq t \leq t_m \end{cases} \tag{6.1}$$

The reason for the choice of this basis is simple. We are interested in finding trajectories that minimize functionals which can be written using these basis functions, namely:

$$\int_{t_0}^{t_m} \mu_r \left|\left| \frac{d^k \mathbf{r}}{dt^k} \right|\right|^2 + \mu_\psi \ddot{\psi}^2 dt$$

where μ_r and μ_ψ are constants that make the integrand nondimensional. For example, Flash and Hogan [29] showed human reaching trajectories appear to minimize the integral of the square of the norm of the jerk (the derivative of acceleration, $k = 3$). These trajectories are quintics that can clearly be written with the basis (7.2). Indeed, there are also studies in human movement [44] that show minimizing the integral of the square norm of derivatives of torques may be a better criterion for modeling motions. In our system, since the inputs u_2 and u_3 appear as functions of the fourth derivatives of the positions, we generate trajectories that minimize the integral of the square of the norm of the snap (the second derivative of acceleration, k=4). The basis (7.2) allows us to go to higher order polynomials which can potentially allow us to satisfy different constraints on the states and the inputs. In what follows we formulate the trajectory generation problem as an optimization of a functional but in a finite dimensional setting.

In order to do this, we first write the constants $\sigma_{Tij} = [x_{Tij}, y_{Tij}, z_{Tij}, \psi_{Tij}]^T$ as a $4nm \times 1$ vector c with decision variables $\{x_{Tij}, y_{Tij}, z_{Tij}, \psi_{Tij}\}$. The trajectory generation problem can then be written in the form of a quadratic program or QP:

$$\begin{aligned} \min \quad & c^T H c + f^T c \\ \text{s.t.} \quad & Ac \leq b \\ & A_{eq} c = b_{eq} \end{aligned} \tag{6.2}$$

where the objective function will incorporate the minimization of the functional while the

constraints can be used to satisfy constraints on the flat outputs and their derivatives and thus constraints on the states and the inputs. A specification of an initial condition, final condition, or intermediate condition on any derivative of the trajectory (e.g., $\frac{d^k x_T}{dt^k}|_{t=t_i}$) can be written as a row of the constraint $A_{eq}c = b_{eq}$. If conditions do not need to be specified exactly then they can be represented with the inequality constraint $Ac \leq b$. After computing the trajectory the methods described in Section 2.3.2 can be used to calculate the angular velocities, angular accelerations, total thrust, and moments required over the entire trajectory.

We now describe two methods of using this technique to generate three-dimensional trajectories for our quadrotor. The first method requires the specification of keyframes in $R^3 \times SO(2)$ where the time required to execute the trajectory is variable and can be scaled to the smallest possible value. We call this *Optimal Keyframe Navigation*. The second method, *Fixed Terminal Time Trajectories*, enables the generation of trajectories with a specified time duration.

6.2.1 Optimal Keyframe Navigation

Here we define a *keyframe* as a position in space along with a yaw angle. Consider the problem of navigating through m keyframes at specified times. In between each keyframe there is a safe corridor that the quadrotor must stay within. This sequence of keyframes could be generated by a planning algorithm. A trivial trajectory that satisfies these constraints is to simply fly in straight lines between keyframes. However this trajectory is inefficient because it has infinite curvature at the keyframes which requires the quadrotor to come to a stop at the keyframes.

Our method generates an optimal trajectory that passes through the keyframes at the given times while staying within the safe corridors. We generate trajectories that smoothly transition through the keyframes with the most natural velocities and accelerations. The optimization program to solve this problem while minimizing the integral of snap squared (without corridor constraints) is shown below.

$$
\begin{aligned}
\min \quad & \int_{t_0}^{t_m} \mu_r \left|\left| \tfrac{d^4\mathbf{r}}{dt^4} \right|\right|^2 + \mu_\psi \ddot{\psi}_T^2 dt && && (6.3)\\
\text{s.t.} \quad & \mathbf{r}_T(t_i) = \mathbf{r}_i, && i = 0, ..., m \\
& \psi_T(t_i) = \psi_i, && i = 0, ..., m \\
& \tfrac{d^p x_T}{dt^p}|_{t=t_j} = 0 \text{ or free}, && j = 0, m,\ p = 1, 2, 3, 4 \\
& \tfrac{d^p y_T}{dt^p}|_{t=t_j} = 0 \text{ or free}, && j = 0, m,\ p = 1, 2, 3, 4 \\
& \tfrac{d^p z_T}{dt^p}|_{t=t_j} = 0 \text{ or free}, && j = 0, m,\ p = 1, 2, 3, 4 \\
& \tfrac{d^p \psi_T}{dt^p}|_{t=t_j} = 0 \text{ or free}, && j = 0, m,\ p = 1, 2
\end{aligned}
$$

Here $\mathbf{r}_T = [x_T, y_T, z_T]^T$ and $\mathbf{r}_i = [x_i, y_i, z_i]$. We assume that $t_0 = 0$ without loss of generality. We use piecewise polynomial functions to define the trajectory and put this problem in the form of (7.3). In between piecewise polynomial functions (at t_1,...,t_{m-1}) we enforce continuity between the first four derivatives of $\mathbf{r}_T$ and first two derivatives of ψ_T.

Nondimensional problem

We note that in (7.1) the quantities x_T, y_T, z_T, and ψ_T are independent in both the cost function and the constraints so this problem can be separated into four separate optimization problems. We now consider a general form of the optimization problem for the nondimensional variable $\tilde{w}(\tau)$ where τ represents nondimensionalized time:

$$
\begin{aligned}
\min \quad & \int_0^1 \tfrac{d^k \tilde{w}(\tau)}{d\tau^k}^2 d\tau && && (6.4)\\
\text{s.t.} \quad & \tilde{w}(\tau_i) = \tilde{w}_i, && i = 0, ..., m \\
& \tfrac{d^p \tilde{w}(\tau)}{d\tau^p}|_{\tau=0} = 0 \text{ or free}, && p = 1, 2, 3, 4 \\
& \tfrac{d^p \tilde{w}(\tau)}{d\tau^p}|_{\tau=1} = 0 \text{ or free}, && p = 1, 2, 3, 4
\end{aligned}
$$

We will show that this nondimensional problem can be transformed into one to solve for any of the variables x_T, y_T, z_T, or ψ_T. First we introduce the dimensional time, $t = \alpha\tau$, and the dimensional variable, w, defined as:

$$w(t) = w(\alpha\tau) = \beta_1 + \beta_2\tilde{w}(\tau)$$

Next we rewrite (6.4) using w and t:

$$\begin{aligned} \min \quad & \frac{\alpha^{2k-1}}{\beta_2}\int_0^\alpha \frac{d^k w(t)}{dt^k}dt \\ \text{s.t.} \quad & w(t_i) = \beta_1 + \beta_2\tilde{w}_i, \quad i = 1, ..., m \\ & \tfrac{d^p w(t)}{dt^p}|_{t=0} = 0 \text{ or free}, \quad p = 1, 2, 3, 4 \\ & \tfrac{d^p w(t)}{dt^p}|_{t=\alpha} = 0 \text{ or free}, \quad p = 1, 2, 3, 4 \end{aligned} \tag{6.5}$$

Note that in this problem the boundary conditions are spatially shifted by β_1 and scaled by β_2 and time is scaled by α. Letting the solution to the nondimensional problem be $\tilde{w}^*$ the solution to the new problem is:

$$w^*(t) = \beta_1 + \beta_2\tilde{w}^*\left(t/\alpha\right)$$

Now let's consider the nondimensional form of (7.1) where the nondimensional $\tilde{\mathbf{r}}$, $\tilde{\psi}$, and τ replace $\mathbf{r}$, ψ, and t. One can solve four nondimensional problems by letting $\tilde{\mathbf{r}}_T = [\tilde{w}_1, \tilde{w}_2, \tilde{w}_3]^T$ and $\tilde{\psi}_T = \tilde{w}_4$. Then the optimal nondimensional solutions, $\tilde{w}_i^*(t)$, can be mapped to the optimal solutions for x_T, y_T, z_T, and ψ_T in the original problem (7.1). The time scale, α, must the same for each variable but the spatial transformation, β_1 and β_2, can be unique.

Adding corridor constraints

We will now add corridor constraints to (7.1). First we define $\mathbf{t}_i$ as the unit vector along the segment from $\mathbf{r}_i$ to $\mathbf{r}_{i+1}$:

$$\mathbf{t}_i = \frac{\mathbf{r}_{i+1} - \mathbf{r}_i}{\|\mathbf{r}_{i+1} - \mathbf{r}_i\|}$$

The perpendicular distance vector, $\delta_i(t)$, from segment i is defined as

$$\delta_i(t) = (\mathbf{r}_T(t) - \mathbf{r}_i) - ((\mathbf{r}_T(t) - \mathbf{r}_i) \cdot \mathbf{t}_i)\mathbf{t}_i$$

A corridor width on the infinity norm, d_i, is defined for each corridor.

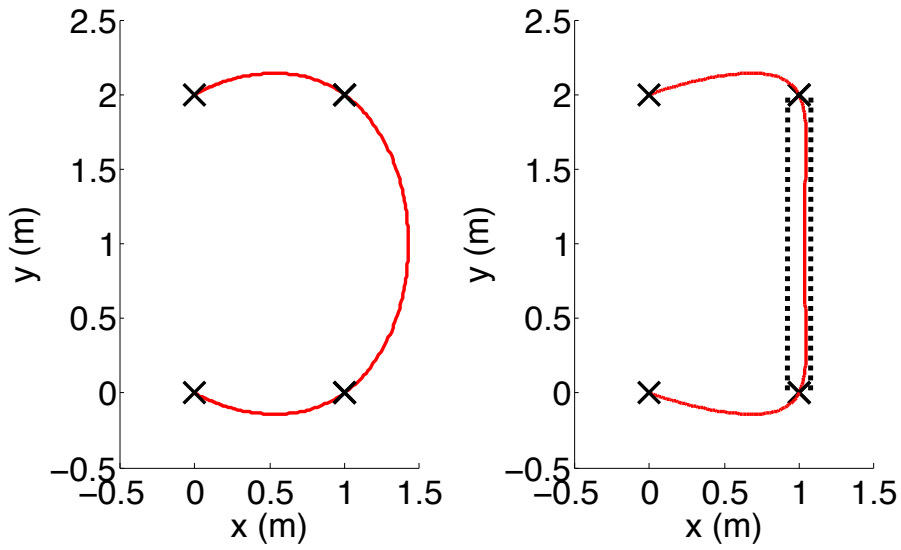


Figure 6.1: Optimal trajectories (red) to pass through 4 keyframes (black). Left: no corridor constraints. Right: corridor constraint between keyframes 2 and 3 forces changes from the unconstrained trajectory on the left.

$$\|\delta_i(t)\|_\infty \leq d_i \text{ while } t_i \leq t \leq t_{i+1}$$

This constraint can be approximately satisfied in the QP by introducing n_c intermediate points as follows

$$|\mathbf{x}_W \cdot \delta_i(t_i + \frac{j}{1+n_c}(t_{i+1} - t_i))| \leq d_i \text{ for } j = 1, ..., n_c$$

and equivalently for $\mathbf{y}_W$ and $\mathbf{z}_W$. Note that the absolute value constraints can each be written as two linear constraints. The use of corridor constraints is shown in Fig. 6.1. In the left figure the optimization problem is solved without any corridor constraints and in the right figure a corridor constraint is added for the 2nd segment ($d_2 = 0.05$ and $n_c = 8$). The trajectory stays within the dotted lines that illustrate the corridor.

Temporal Scaling

Next we consider changing the time to navigate the keyframes by a factor of α so that the new times to reach the keyframes are $t_i = \alpha\tau_i$. We let the nondimensional desired

boundary conditions be $\tilde{\mathbf{r}}_i = \mathbf{r}_i$ and $\tilde{\psi}_i = \psi_i$ in the nondimensional form of (7.1) with the corridor constraints. The solution to the true problem is simply a time-scaled version of the nondimensional solution.

$$\mathbf{r}_T^*(t) = \tilde{\mathbf{r}}_T^*(t/\alpha),\ \psi_T^*(t) = \tilde{\psi}_T^*(t/\alpha)$$

This property can be used to tradeoff between fast, aggressive trajectories and slow, safe trajectories. As α is increased the plan takes longer to execute and becomes *safer*. As α goes to infinity all the derivatives of position and yaw angle go to zero which leads to:

$$\mathbf{u}(t) \to [mg, 0, 0, 0]^T,\ \omega_{\mathcal{BW},\mathcal{T}}(t) \to [0, 0, 0]^T$$

We can therefore satisfy arbitrary constraints of safeness by making α large enough. Conversely, as α is decreased the trajectory takes less time to execute, the derivatives of position increase, and the trajectory becomes more aggressive.

Optimal segment times

In some cases the arrival times at different keyframes is important and may be specified. However, in other cases these arrival times may not matter and we can try to find a more optimal solution by allowing more time for some segments while taking the same amount of time away from the others. Here we describe a method for finding the optimal relative segment times for a given set of keyframes. For this part it is more convenient to think of the time allowed for segment i, T_i, rather than the arrival time for keyframe i, t_i where $T_i = t_i - t_{i-1}$. We then solve the minimization problem:

$$\begin{aligned} \min \quad & f(\mathbf{T}) \\ \text{s.t.} \quad & \textstyle\sum T_i = t_m \\ & T_i \geq 0 \end{aligned} \tag{6.6}$$

where $f(\mathbf{T})$ is the solution the optimization problem (7.1) for segment times $\mathbf{T}$. We solve (6.6) via a constrained gradient descent method. The vectors $\mathbf{g}_i$ are constructed so that the

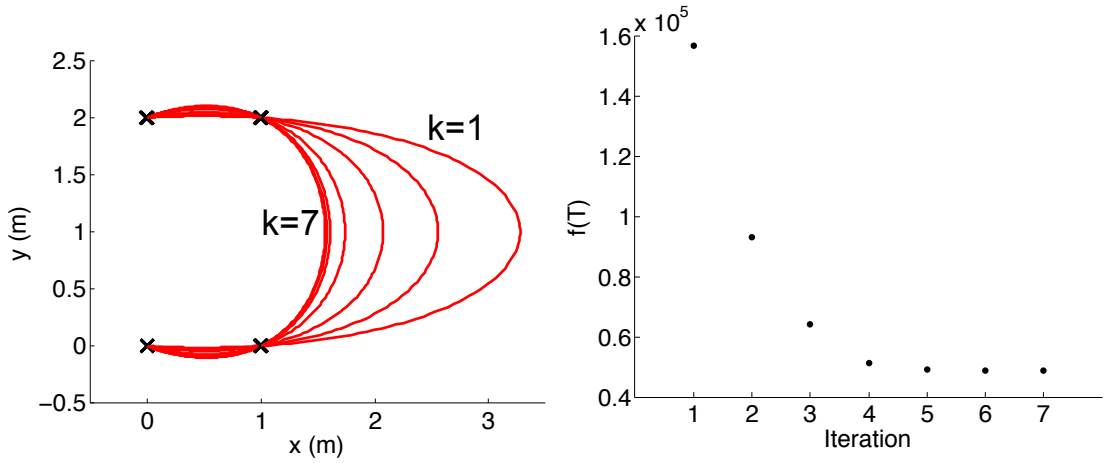


Figure 6.2: Illustration of relative time scaling. Left: Trajectory for different iterations. Right: Cost function vs. iteration.

ith element has a value of 1 and all other elements have the value $\frac{-1}{m-2}$. This is done so that $\sum \mathbf{g}_i = 0$ and $\mathbf{g}_i$ can be added or subtracted from $\mathbf{T}$ and the final time does not change.

Next we numerically compute the directional direction for each $\mathbf{g}_i$ as follows.

$$\nabla_{g_i} f = \frac{f(\mathbf{T} + h\mathbf{g}_i) - f(\mathbf{T})}{h}$$

where h is some small number. Given the estimates of the directional derivatives we perform gradient descent using backtracking line search. An illustration of this method for a trajectory in the $x - y$ plane where the keyframes are simply points is shown in Fig. 6.2. The first choice of segment times allowed too much time for the 2nd segment and the trajectory for this segment deviates significantly from the convex hull formed by the keyframes. After 7 iterations the cost function converges to a minimum. The optimal trajectory appears to be a very natural trajectory for passing through all keyframes which qualitatively validates the choice of the cost function.

6.2.2 Fixed Terminal Time Trajectories

Next we consider the problem of optimally reaching a position and yaw angle in some fixed time from a rest state. We consider cases where the components of the derivatives of

position at the final time, t_1, are either specified to be 0 or are free. Note that this is just a special case of (7.1) with $m = 1$ and all derivatives of $\mathbf{r}_T$ and ψ_T at $t = 0$ specified to be 0.

Spatial scaling

In order to solve this problem quickly, we will exploit the spatial scaling property described previously. We consider a single case of the nondimensional form of (7.1) where $\tilde{\mathbf{r}}_T(0) = \mathbf{0}$ and $\tilde{\mathbf{r}}_T(1) = \mathbf{1}$ and the final velocities are specified the same as in the true problem. Then we solve the nondimensional problem once and transform the solution to find the optimal solution to our actual problem by setting

$$x_T^*(t) = x_0 + (x_1 - x_0)\tilde{x}_T^*(t/t_1)$$

and likewise for $y_T^*(t)$ and $z_T^*(t)$. This is convenient because it is faster to analytically modify a solution than resolve a QP. For this reason, this approach is useful for quickly reacting to dynamic obstacles or dynamic targets. We use it to fly through a thrown circular hoop as shown in the next section. Note that spatial scaling also applies to the problem with multiple keyframes but the property is less useful as the positions of all keyframes must be scaled by the same factor.

6.3 Experiments

The experiments are conducted with Ascending Technologies Hummingbird quadrotor [1]. We use a Vicon motion capture system [8] to estimate the position, orientation, and velocity of the quadrotor and the onboard gyros to estimate the angular velocities. The software infrastructure is described in [62].

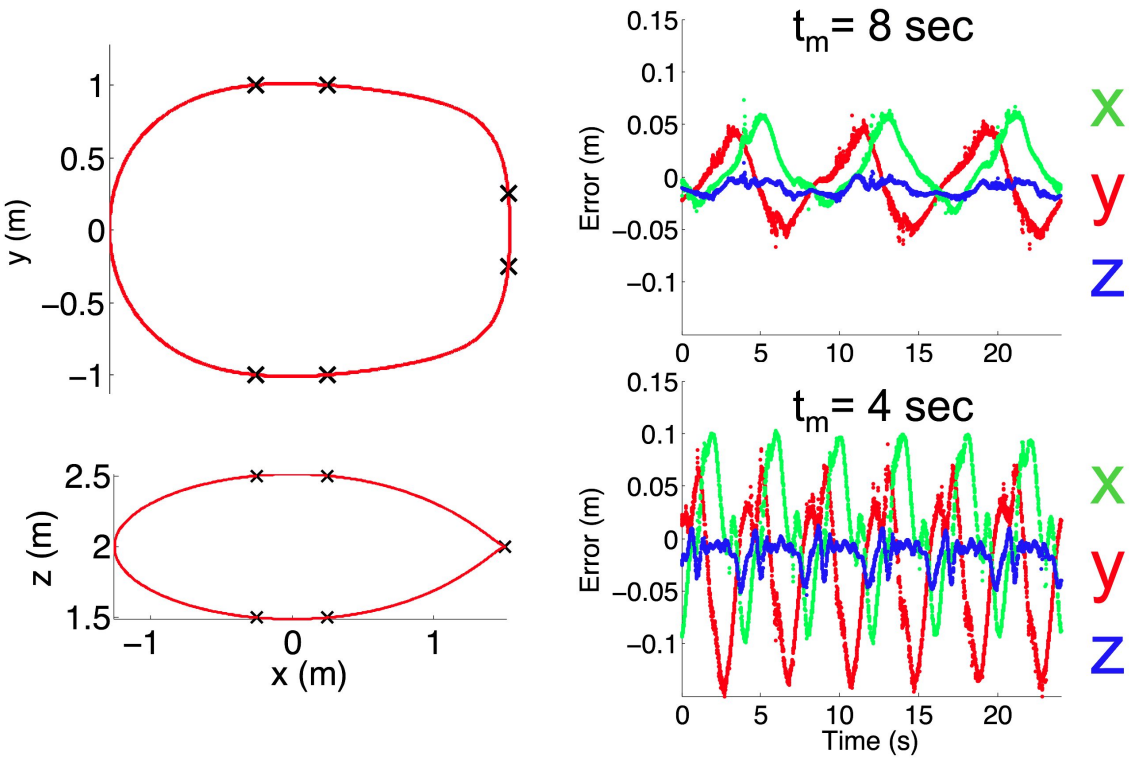


Figure 6.3: Trajectory generated to fly through three gaps (left) and performance data for two traversal speeds (right).

6.3.1 Flying through three static hoops

This experiment demonstrates the ability to fly through environments with several narrow gaps. We design a scenario with three fixed circular hoops the quadrotor must continuously fly through. Six keyframes with the identical yaw angles are selected at 0.25 meters on either side of the gaps with a small corridor constraint, 1 cm, added for the segments passing through the gaps. The corridor widths for the other segments are allowed to be 1 meter so the quadrotor may take a more optimal path where there is no position constraint. Since arrival time at the keyframes is not important for this problem the optimal relative time scaling method is used. The final trajectory generated is shown in Fig. 6.3.

This generated trajectory can be tracked at different speeds. The right side of the Fig. 6.3 shows 24 seconds of performance data for tracking this trajectory in 8 seconds (top)

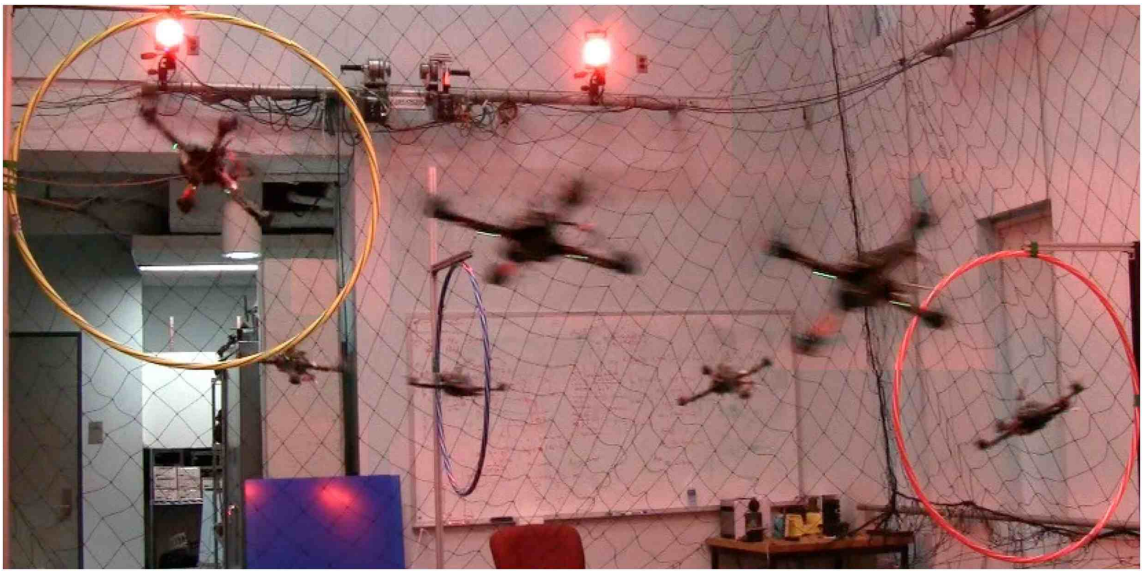


Figure 6.4: Composite image of a single quadrotor quickly flying through three static circular hoops. See attached video or http://tinyurl.com/pennquad.

and 4 seconds (bottom). The data shows that we can tradeoff speed for accuracy. The faster trajectory has velocities as large as 2.6 m/s and roll and pitch angles of up to 40°. Images from the faster experiment are shown in Fig. 6.4.

6.3.2 Flying through a thrown hoop

This experiment demonstrates the capability of avoiding fast moving dynamic obstacles. We use fixed terminal time trajectories to fly through a thrown circular hoop. After detecting that the hoop has been thrown the future position of the hoop is predicted with a quadratic air drag model. The predicted future time and x and y position of descent through a chosen z plane is found. The chosen z-plane is 0.6 meters below the quadrotor. The allowed region for hoop interception is Δx = 1.2 to 1.6 meters and Δy = -0.4 to 0.4 meters, where x is towards the hoop. The time allowed for all trajectories, t_1, is 0.9 seconds. The x and z velocity are allowed to be free so the quadrotor can fly forward and down through the hoop while the y velocity is constrained to be zero as it is assumed the hoop falls approximately straight down. The worst case performance is for the position

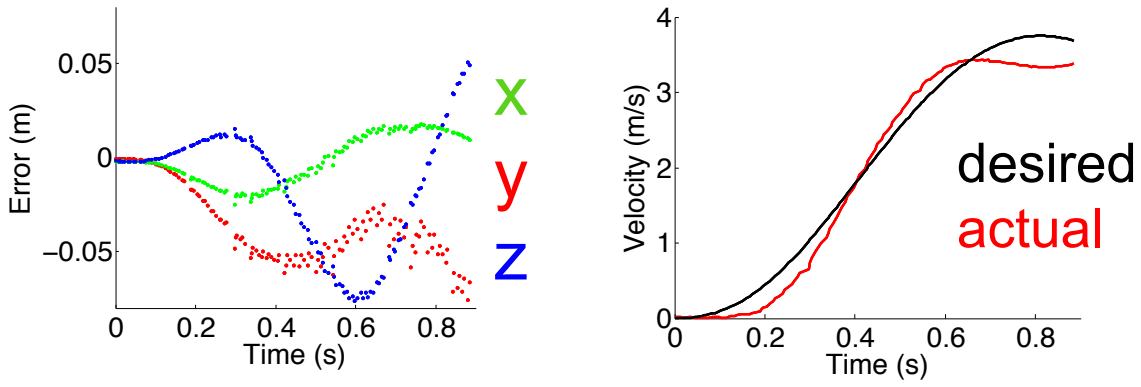


Figure 6.5: Performance data for aggressive fixed terminal time trajectory.

the farthest away ($\Delta x = 1.6$ meters and $\Delta y = 0.4$ meters) for which data is shown in Fig. 6.5.

Even in this worst-case scenario the position error is always less than 8 cm in any direction. Note that this is a highly aggressive trajectory as the quadrotor quickly reaches a velocity of 3.6 m/s and at one point hits a pitch angle of 60°. A series of images showing the full experiment are shown in Fig. 6.6.

6.3.3 Catching a bouncing ball

This experiment demonstrates the ability to continuously replan in order to catch a bouncing ball. To enable this experiment a gripper was developed to throw and catch a golf ball from the underside of a quadrotor. The sides of the gripper can be opened simultaneously in order to drop the ball straight down or can be actuated independently in order to throw the ball to the left or to the right as shown in Fig. 6.7.

Here we describe a method for catching a ball at the peak of its flight when the z velocity is zero. As the ball ascends into the gripper the gripper begins to close. The ball then reaches its peak inside the gripper and by the time the ball begins to descend the gripper has closed and the ball remains inside the gripper. A flowchart outlining the

Figure 6.6: Composite image of a single quadrotor flying through a thrown circular hoop. See attached video or http://tinyurl.com/pennquad.

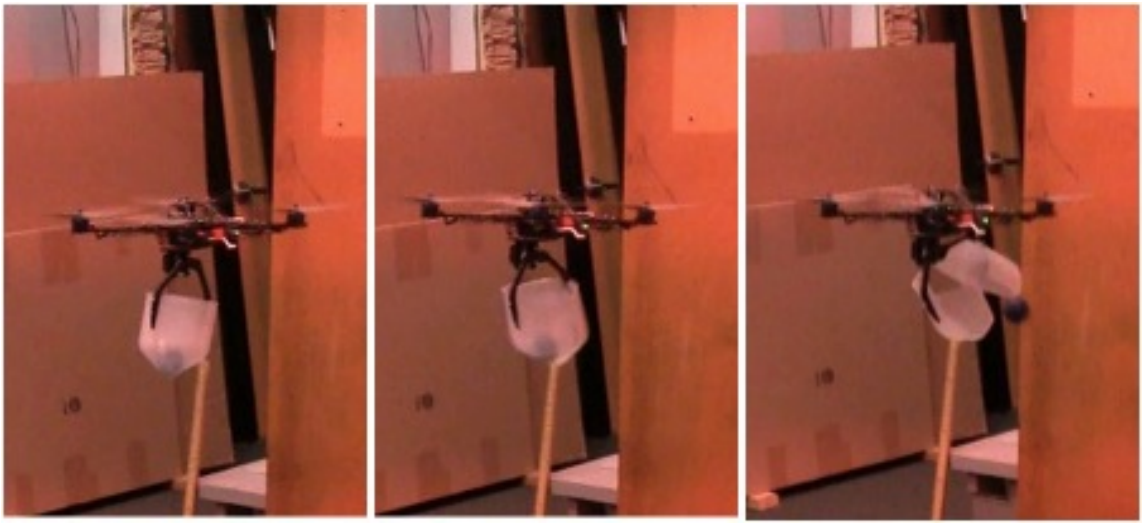


Figure 6.7: The quadrotor throwing the ball to the right.

method is shown in Fig. 6.8. As seen in this figure the method for catching a ball requires two main components, the prediction of the peak of the ball, the *Apogee Estimator*, and the planning to reach the desired state, the *Trajectory Generator*.

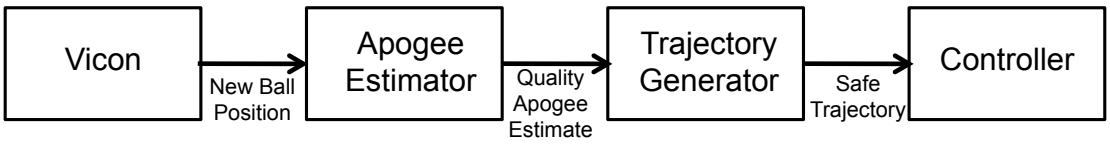


Figure 6.8: Flowchart for catching a ball.

Apogee Estimator

Here we describe our method for predicting when and where the ball will reach its apogee after a bounce. The first requirement is to find the current position and velocity of the ball. We cover the ball with reflective tape and use a Vicon Motion capture system to track it at 150 Hz. There is some noise in the Vicon measurements and Vicon occasionally loses track of the ball. Therefore we developed a method that estimates the state of the ball only if the system has maintained a quality track for at least some amount of time and if the noise is below some threshold.

At each time step we consider the last n measurements (t_i, x_i, y_i, z_i) collected during the last 0.07 seconds of data. Note that we require n to be larger than some threshold. We assume that each measurement is subject to some error (ϵ_{xi}, ϵ_{yi}, ϵ_{zi}). Our measurement model is shown in the equations below.

$$\begin{aligned} x_i &= x_0 + v_{x0}t_i + \epsilon_{xi} \\ y_i &= y_0 + v_{y0}t_i + \epsilon_{yi} \\ z_i &= z_0 + v_{z0}t_i - \tfrac{1}{2}gt_i^2 + \epsilon_{zi} \end{aligned} \tag{6.7}$$

Note that here we assume the ball is undergoing parabolic flight under the acceleration of gravity. This implies that we do not attempt to estimate the state of the ball during a bounce event. We also ignore air drag in the model because at low speeds the air drag on the golf ball is negligible.

Using this measurement model the $3n$ measurements can be written as:

$$A \begin{bmatrix} x_0 \\ v_{x0} \\ y_0 \\ v_{y0} \\ z_0 \\ v_{z0} \end{bmatrix} = A\beta = \mathbf{b} + \epsilon \tag{6.8}$$

where A is a $3n \times 6$ matrix and $\mathbf{b}$ is $3n$ length vector which both contain the measurements. Here ϵ is a $3n$ length vector containing the measurement noise. The intial velocites and positions are represented by β which we estimate using a linear least squares method:

$$\hat{\beta} = (A^T A)^{-1} A^T \mathbf{b} \tag{6.9}$$

In order to accept the estimate as a quality estimate the mean squared error, $\frac{\|A\hat{\beta}-\mathbf{b}\|^2}{3n}$, must be less than some threshold. Given a quality estimate of the initial state of the ball we use a parabolic flight model to find the impact location and velocity of the ball. A golf ball was chosen because it is has a very high coefficient of restitution and also has a consistent bounce. The bouncing surface is a strong factor is determining the bounce of the ball. We use a steel lab benchtop that is 5 cm thick. Our impact model is shown in (6.10) where $-$ and $+$ superscripts represent pre- and post-impact velocities, respectively.

$$\begin{aligned} v_x^+ &= v_x^- \\ v_y^+ &= v_y^- \\ v_z^+ &= -COR(v_z^-)v_z^- \end{aligned} \tag{6.10}$$

Note that we experimentally determined the coefficient of restitution as a linear function of the velocity of impact, $COR(v_z^-)$. We assume that the x and y velocity are the same pre- and post-impact. However, we acknowledge that unmodeled disturbance is added to the ball when it bounces off the ground due to imperfections in the ball and the surface as well as spin on the ball. After calculating the post-impact velocities we use a parabolic flight model to predict the position and velocity of the ball at its apogee.

Trajectory Generator

When a quality apogee estimate is produced the Apogee Estimator feeds the apogee position, velocity, and time to the Trajectory Generator. The Trajectory Generator computes a minimum snap trajectory from the current desired state of the quadrotor to this desired catch state. Note that we require the catch state to have acceleration and jerk of zero so that the catch is performed with the quadrotor at a level attitude with no angular velocity.

It is possible that the ball is thrown out of reach of the vehicle and we must account for this so the quadrotor does not attempt to follow trajectories that are unsafe. Therefore, after a trajectory is planned we find the maximum velocities, accelerations, jerks, and snaps for the planned polynomial trajectories along each axis. If any of these values exceed chosen safe thresholds then the trajectory is determined to be unsafe and is not executed.

Although it is ideal to reach the catch position at the velocity of the ball, it is possible to still catch the ball even if the vehicle does not move at the ball's exact velocity. So, if the planned trajectory is unsafe we next plan a path to the catch position and allow the final velocity to be unspecified. This less constrained trajectory will have a lower cost and it is possible that it will be safe while the first planned trajectory was not. If this trajectory is safe, then it is executed. If not, then the quadrotor simply executes whatever safe trajectory it was already executing. Screenshots from a successful experimental trial are shown in Fig. 6.9.

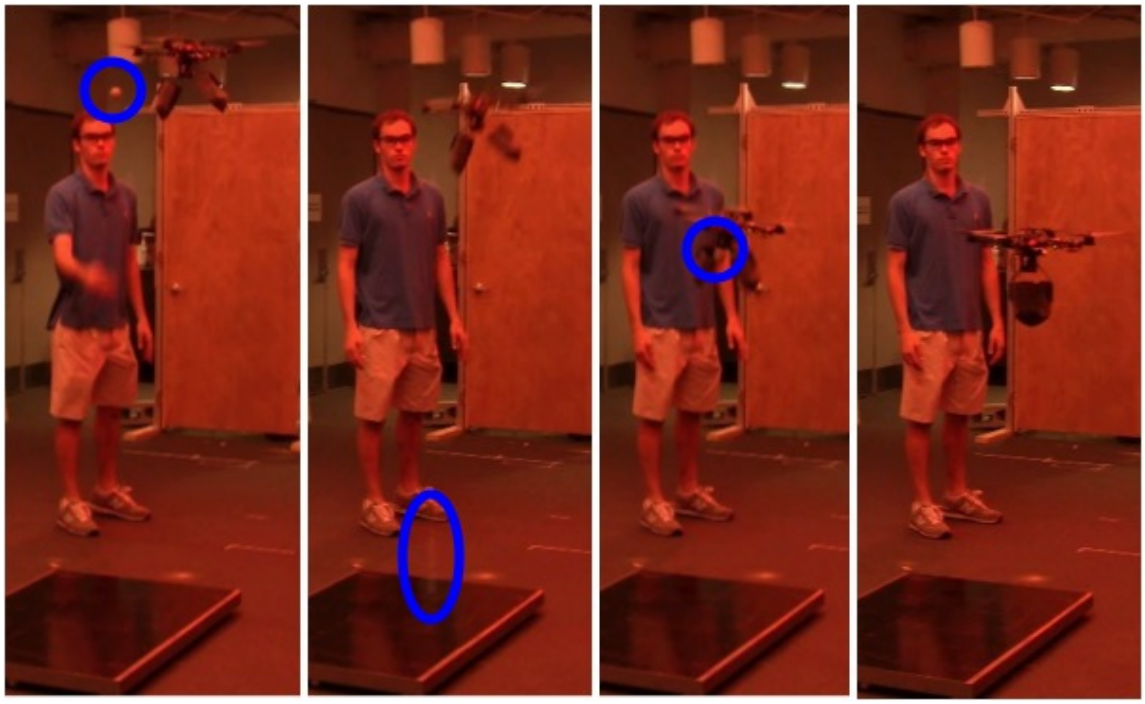


Figure 6.9: Quadrotor catching a thrown ball after a bounce. The ball is highlighted in blue.

Chapter 7

Trajectory Generation with Mixed-Integer Quadratic Programs for Heterogeneous Quadrotor Teams

We present an algorithm for the generation of optimal trajectories for teams of heterogeneous quadrotors in three-dimensional environments with obstacles. We formulate the problem using mixed-integer quadratic programs (MIQPs) where the integer constraints are used to enforce collision avoidance. The method allows for different sizes, capabilities, and varying dynamic effects between different quadrotors. Experimental results illustrate the method applied to teams of up to four quadrotors ranging from 65 to 962 grams and 21 to 67 cm in width following trajectories in three-dimensional environments with obstacles with accelerations approaching $1g$.

7.1 Introduction

Multi-rotor aerial vehicles have become increasingly popular robotic platforms because of their mechanical simplicity, dynamic capabilities, and suitability for both indoor and outdoor environments. In particular, there have been many recent advances in the design [34],

control [54] and planning [35] for quadrotors, rotorcrafts with four rotors. In this paper we present a method for generating optimal trajectories for heterogeneous quadrotor teams like those shown in Fig. 7.1 in environments with obstacles.

Trajectories that quadrotors can follow quickly and accurately should be continuous up to the third derivative of position (or C^3). This is because, for quadrotors, discontinuities in lateral acceleration require instantaneous changes in roll and pitch angles and discontinuities in lateral jerk require instantaneous changes in angular velocity. Finding C^3 trajectories requires planning in a high-dimensional search space which is impractical for methods using reachability algorithms [33], incremental search techniques [52] or LQR-tree-based searches [75]. The problem is exacerbated when planning for multiple vehicles as this further expands the dimension of the search space.

This paper builds on our own previous work [55] in which we showed that the dynamic model for the quadrotor is differentially flat. We used this fact to derive a trajectory generation algorithm that allows us to naturally embed constraints on desired positions, velocities, accelerations, jerks and inputs while satisfying requirements on smoothness of the trajectory. We extend that method in this work to include multiple quadrotors and obstacles. The method allows for different sizes, capabilities, and varying dynamic effects between different quadrotors. We enforce collision avoidance using using integer constraints which transforms our quadratic program (QP) from [55] into a mixed-integer quadratic program (MIQP).

Our work also draws from the extensive literature on mixed-integer linear programs (MILPs) and their application to trajectory planning from Schouwenaars et al. [68, 70–72]. This body of work demonstrates the power and flexibility of integer constraints in similar trajectory planning problems for both fixed-wing aerial vehicles and rotorcraft. A key difference in our approach is that we use piece-wise smooth polynomial functions to synthesize trajectories in the flat output space. Using piece-wise smooth polynomial functions allows us to enforce continuity between waypoints up to any desired derivative of position. Another difference in our work from this previous work on trajectory generation is the use

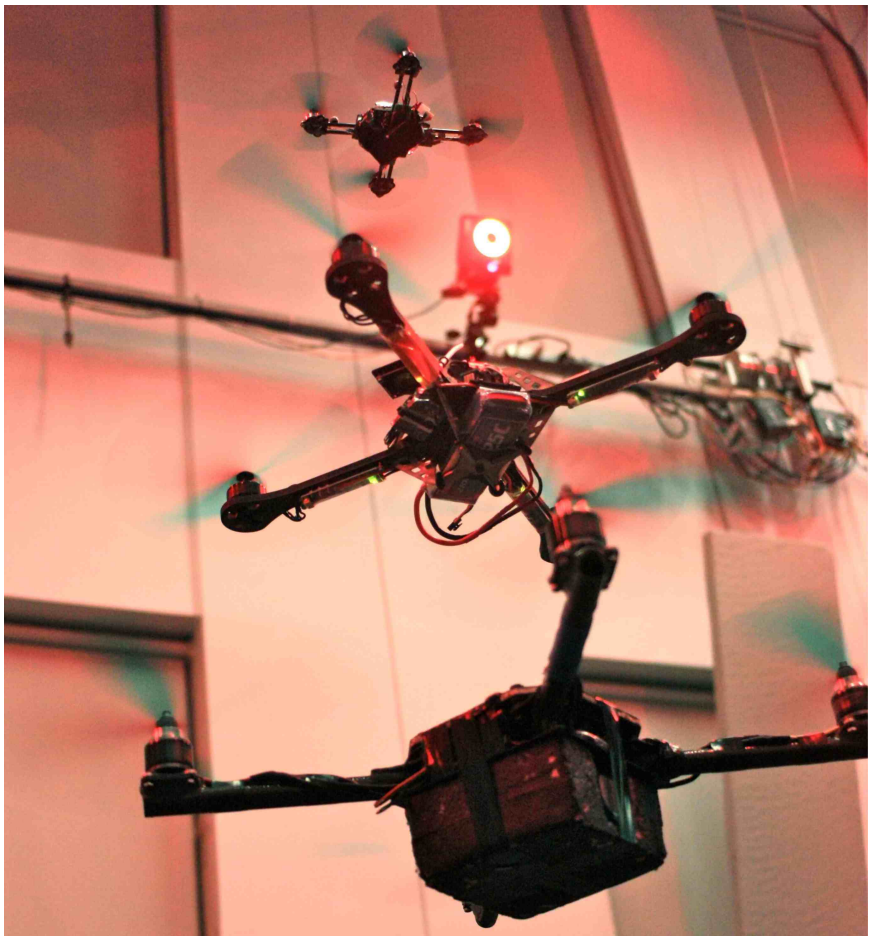


Figure 7.1: The kQuad65 (top), the Asctec Hummingbird [1] (middle), and the kQuad1000 (bottom).

of quadratic cost function resulting in a MIQP as opposed to a MILP.

The organization of the chapter is as follows. First, we present our trajectory generation method for a single quadrotor in Sec. 7.2 and its extension to heterogeneous quadrotor teams in Sec. 7.3. In Sec. 7.4, we present experimental results for teams of up to four quadrotors ranging from 65 to 962 grams and 21 to 67 cm in width shown in Fig. 7.1 following trajectories in three-dimensional environments with obstacles with accelerations approaching $1g$. Finally, in Sec.7.5, we offer some concluding remarks on this approach.

7.2 Single Quadrotor Trajectory Generation

In this section we first describe the basic quadrotor trajectory generation method using Legendre polynomial functions incorporating obstacles into the formulation. Specifically, we solve the problem of generating smooth, safe trajectories through known 3-D environments satisfying specifications on intermediate waypoints.

7.2.1 Basic Method

Consider the problem of navigating a vehicle through n_w waypoints at specified times. A trivial trajectory that satisfies these constraints is one that interpolates between waypoints using straight lines. However this trajectory is inefficient because it has infinite curvature at the waypoints which requires the quadrotor to come to a stop at each waypoint. Our method generates an optimal trajectory that smoothly transitions through the waypoints at the given times. The optimization program to solve this problem while minimizing the integral of the k_rth derivative of position squared is shown below.

$$
\begin{aligned}
\min \quad & \int_{t_0}^{t_{n_w}} \left|\left| \tfrac{d^{k_r}\mathbf{r_T}}{dt^{k_r}} \right|\right|^2 dt && (7.1)\\
\text{s.t.} \quad & \mathbf{r}_T(t_w) = \mathbf{r}_w, \qquad w = 0, ..., n_w \\
& \tfrac{d^j x_T}{dt^j}|_{t=t_w} = 0 \text{ or free}, w = 0, n_w;\ j = 1, ..., k_r \\
& \tfrac{d^j y_T}{dt^j}|_{t=t_w} = 0 \text{ or free}, w = 0, n_w;\ j = 1, ..., k_r \\
& \tfrac{d^j z_T}{dt^j}|_{t=t_w} = 0 \text{ or free}, w = 0, n_w;\ j = 1, ..., k_r
\end{aligned}
$$

Here $\mathbf{r}_T = [x_T, y_T, z_T]^T$ and $\mathbf{r}_i = [x_i, y_i, z_i]^T$. We enforce continuity of the first k_r derivatives of $\mathbf{r}_T$ at t_1,...,t_{n_w-1}. Next we write the trajectories as piecewise polynomial functions of order n_p over n_w time intervals using polynomial basis functions $P_{pw}(t)$:

$$\mathbf{r}_T(t) = \begin{cases} \sum_{p=0}^{n_p} \mathbf{r}_{Tp1} P_{p1}(t) & t_0 \leq t < t_1 \\ \sum_{p=0}^{n_p} \mathbf{r}_{Tp2} P_{p2}(t) & t_1 \leq t < t_2 \\ \vdots & \\ \sum_{p=0}^{n_p} \mathbf{r}_{Tpn_w} P_{pn_w}(t) & t_{n_w-1} \leq t \leq t_{n_w} \end{cases} \tag{7.2}$$

This allows us formulate the problem as a quadratic program (or QP) by writing the constants $\mathbf{r}_{Tpw} = [x_{Tpw}, y_{Tpw}, z_{Tpw}]^T$ as a $3n_w n_p \times 1$ decision variable vector c:

$$\begin{aligned} \min \quad & c^T H c + f^T c \\ \text{s.t.} \quad & Ac \leq b \\ & A_{eq} c = b_{eq} \end{aligned} \tag{7.3}$$

In our system, since the inputs u_2 and u_3 appear as functions of the fourth derivatives of the positions, we generate trajectories that minimize the integral of the square of the norm of the snap (the second derivative of acceleration, $k_r = 4$). The basis (7.2) allows us to go to higher order polynomials which allows us to satisfy such additional trajectory constraints as obstacle avoidance that are not explicitly specified by intermediate waypoints.

7.2.2 Choice of basis functions

Although this problem formulation is valid for any set of spanning polynomial basis functions, $P_{pw}(t)$, the choice does affect the numerical stability of the solver. A poor choice of basis functions can cause the matrix H in (7.3) to be ill-conditioned for large order polynomials. In order to diagonalize H and ensure that it is well-conditioned matrix we use Legendre polynomials as basis functions for the k_rth derivatives of our positions. Legendre polynomials are a spanning set of orthogonal polynomials on the interval from -1 to 1:

$$\int_{-1}^{1} \lambda_m(\tau)\lambda_n(\tau)d\tau = \frac{2}{2n+1}\delta_{nm}$$

where δ_{nm} is the Kronecker delta and τ is the non-dimensionalized time [10]. We then shift these Legendre polynomials to be orthogonal on the interval from t_{w-1} to t_w which we

call $\lambda_{pw}(t)$. We use these shifted Legendre polynomials to represent the k_rth derivatives of the first $n_p - k_r$ basis functions for our position functions, $P_{pw}(t)$. These first $n_p - k_r$ polynomials must satisfy

$$\frac{d^{k_r} P_{pw}(t)}{dt^{k_r}} = \lambda_{pw}(t) \quad p = 1, ..., (n_p - k_r)$$

We define the last k_r polynomial basis functions as

$$P_{pw}(t) = (t - t_{w-1})^{p-n_p+k_r-1} \quad p = (n_p - k_r + 1), ..., n_p$$

Note these last k_r polynomial basis functions have no effect on the cost function because their k_rth derivatives are zero. In our work we take $k_r = 4$ and n_p is generally between 9 and 15.

7.2.3 Integer Constraints for Obstacle Avoidance

For collision avoidance we model the quadrotor as a rectangular prism oriented with the world frame with side lengths l_x, l_y, and l_z. These lengths are large enough so that the quadrotor can roll, pitch, and yaw to any angle and stay within the prism. We consider navigating this prism through an environment with n_o convex obstacles. Each convex obstacle o can be represented by a convex region in configuration space with $n_f(o)$ faces. For each face f the condition that the quadrotor's desired position at time t_k, $\mathbf{r}_T(t_k)$, be outside of obstacle o can be written as

$$\mathbf{n}_{of} \cdot \mathbf{r}_T(t_k) \leq s_{of}, \tag{7.4}$$

where $\mathbf{n}_{of}$ is the normal vector to face f of obstacle o in configuration space and s_{of} is a scalar that determines the location of the plane. If (7.4) is satisfied for *at least* one of the faces then the rectangular prism, and hence the quadrotor, is not in collision with the obstacle. The condition that the prism does not collide with an obstacle o at time t_k can

be enforced with binary variables, b_{ofk}, as

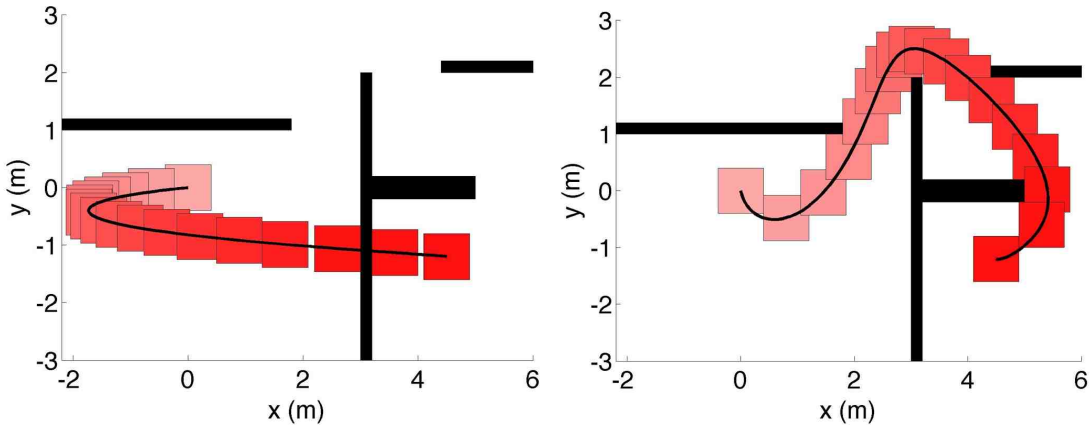
$$\begin{aligned} \mathbf{n}_{of} \cdot \mathbf{r}_T(t_k) &\leq s_{of} + M b_{ofk} \quad \forall f = 1, ..., n_f(o) \\ b_{ofk} &= 0 \text{ or } 1 \quad \forall f = 1, ..., n_f(o) \\ \sum_{f=1}^{n_f(o)} b_{ofk} &\leq n_f(o) - 1 \end{aligned} \tag{7.5}$$

where M is a large positive number [70]. Note that if b_{ofk} is 1 then the inequality for face f is always satisfied. The last inequality in (7.5) requires that the non-collision constraint be satisfied for at least one face of the obstacle which implies that the prism does not collide with the obstacle. We can then introduce (7.5) into (7.3) for all n_o obstacles at n_k intermediate time steps between waypoints. The addition of the integer variables into the quadratic program causes this optimization problem to become a mixed-integer quadratic program (MIQP).

Note that this formulation is valid for any convex obstacle but we only consider rectangular obstacles in this paper for simplicity. This formulation is easily extended to moving obstacles by simply replacing $\mathbf{n}_{of}$ with $\mathbf{n}_{of}(t_k)$ and s_{of} with $s_{of}(t_k)$ in (7.5). Non-convex obstacles can also be efficiently modeled in the this framework as discussed in [72].

7.2.4 Discretization in Time

Equation (7.3) represents a continuous time optimization. We discretize time and write the collision constraints in (7.5) for n_k time points. However, collision constraints at n_k discrete times do not guarantee that the trajectory will be collision-free between the time steps. For a thin obstacle the optimal trajectory may cause the quadrotor to travel quickly through the obstacle such that the collision constraints are satisfied just before passing through the obstacle and just after as shown in Fig. 7.2(a). This problem can be fixed by requiring that the rectangular prism for which collision checking is enforced at time step



(a) Time-step overlap constraints are not enforced (b) Time-step overlap constraints are enforced

Figure 7.2: Trajectories for a single quadrotor navigating an environment with four obstacles. Obstacles are the solid black boxes, the trajectory is shown as the black line, the position of the quadrotor at the n_k intermediate time steps for which collision checking is enforced is shown by the red boxes which grow darker with passing time. Note that for both of these trajectories $n_p = 15$ and $n_k = 16$

k has a finite intersection with the corresponding prism for time step $k + 1$:

$$\begin{aligned} |x_T(t_k) - x_T(t_k + 1)| &\leq l_x \quad \forall k = 0, ..., n_k \\ |y_T(t_k) - y_T(t_k + 1)| &\leq l_y \quad \forall k = 0, ..., n_k \\ |z_T(t_k) - z_T(t_k + 1)| &\leq l_z \quad \forall k = 0, ..., n_k \end{aligned} \tag{7.6}$$

These additional *time-step overlap constraints* prevent the trajectory from passing through obstacles as shown in Fig. 7.2(b). Enforcing time-step overlap is equivalent to enforcing an average velocity constraint between time steps. Of course, enough time steps must be used so that a solution is feasible. Note that the trajectory may still cut corners due to the time discretization. We address this by appropriately inflating the size of the obstacles and prisms for which collision checking is enforced. After the trajectory is found we perform a collision check to ensure that the actual quadrotor shape does not intersect with any of the obstacles over the entire trajectory.

7.2.5 Temporal Scaling

As in [55] we can exploit temporal scaling to tradeoff between safety and aggressiveness. If we change the time to navigate the waypoints by a factor of α (e.g., $\alpha = 2$ allows the trajectory to be executed in twice as much time) the solution to the time-scaled problem is simply a time-scaled version of the original solution. Hence, we do not need to resolve the MIQP. As α is increased the plan takes longer to execute and becomes *safer*. As α goes to infinity all the derivatives of position and yaw angle as well as the angular velocity go to zero which leads, in the limit, to

$$\mathbf{u}(t) \rightarrow [mg, 0, 0, 0]^T.$$

By making α large enough we can satisfy any motion plan generated for a quadrotor with the assumption of small pitch and roll. Conversely, as α is decreased the trajectory takes less time to execute, the derivatives of position increase, and the trajectory becomes more aggressive leading to large excursions from the zero pitch and zero roll configuration.

7.3 Multiple Quadrotor Trajectory Generation

In this section we extend the method to include n_q heterogeneous quadrotors navigating in the same environment, often in close proximity, to designated goal positions, each with specified waypoints. This is done by solving a larger version of (7.3) where the decision variables are the trajectories coefficients of all n_q quadrotors. For collision avoidance constraints each quadrotor can be a different size as specified by unique values of l_x, l_y, l_z. We also consider heterogeneity terms with relative cost weighting and inter-quadrotor collision avoidance.

7.3.1 Relative Cost Weighting

A team of quadrotors navigating independently must resolve conflicts that lead to collisions and "share" the three-dimensional space. Thus they must modify their individual

trajectories to navigate an environment and avoid each other. If all quadrotors are of the same type then it makes sense for them to share the burden of conflict resolution equally. However, for a team of heterogeneous vehicles it may be desirable to allow some quadrotors to follow relatively easier trajectories than others, or to prioritize quadrotors based on user preferences. This can be accomplished by weighting their costs accordingly. If quadrotor q has relative cost μ_q then the quadratic cost matrix, H_m, in the multi-quadrotor version of (7.3) can be written

$$H_m = \text{diag}(\mu_1 H_1, \mu_2 H_2, ..., \mu_{n_q} H_{n_q}) \tag{7.7}$$

Applying a larger weighting factor to a quadrotor lets it take a more direct path between its start and goal. Applying a smaller weighting factor forces a quadrotor to modify its trajectory to yield to other quadrotors with larger weighting factors. This ability is particularly valuable for a team of both agile and slow quadrotors as a trajectory for a slow, large quadrotor can be assigned a higher cost than the same trajectory for a smaller and more agile quadrotor. A large quadrotor requires better tracking accuracy than a small quadrotor to fly through the same narrow gap so it is also useful to assign higher costs for larger quadrotors in those situations.

7.3.2 Inter-Quadrotor Collision Avoidance

Quadrotors must stay a safe distance away from each other. We enforce this constraint at n_k intermediate time steps between waypoints which can be represented mathematically

for quadrotors 1 and 2 by the following set of constraints:

$$\begin{aligned} \forall t_k: \quad & x_{1T}(t_k) - x_{2T}(t_k) \leq d_{x12} \\ \text{or} \quad & x_{2T}(t_k) - x_{1T}(t_k) \leq d_{x21} \\ \text{or} \quad & y_{1T}(t_k) - y_{2T}(t_k) \leq d_{y12} \\ \text{or} \quad & y_{2T}(t_k) - y_{1T}(t_k) \leq d_{y21} \\ \text{or} \quad & z_{1T}(t_k) - z_{2T}(t_k) \leq d_{z12} \\ \text{or} \quad & z_{2T}(t_k) - z_{1T}(t_k) \leq d_{z21} \end{aligned} \tag{7.8}$$

Here the d terms represent safety distances. For axially symmetric vehicles $d_{x12} = d_{x21} = d_{y12} = d_{y21}$. Experimentally we have found that quadrotors must avoid flying in the downwash of similar-sized or larger quadrotors because of a decrease in tracking performance and even instability in the worst cases. Larger quadrotors, however, can fly underneath smaller quadrotors. We have demonstrated that a larger quadrotor can even fly stably enough under a small quadrotor to serve as an aerial landing platform (see attached video). Therefore if quadrotor 1 and 2 are of the same type then $d_{z12} = d_{z21}$. However, if quadrotor 1 is much bigger than quadrotor 2 then quadrotor 2 must fly well below the larger quadrotor at some large distance d_{z12} while quadrotor 1 can fly much closer underneath quadrotor 2 represented by the smaller distance d_{z21}. The exact values of these safety distances can be found experimentally by measuring the tracking performance for different separation distances between quadrotor types. Finally, we incorporate constraints (7.8) between all n_q quadrotors in the same manner as in (7.5) into the multi-quadrotor version of (7.3).

7.3.3 Computational Complexity and Numerical Algorithm

Here we analyze the complexity of the MIQP generated by this formulation for a three-dimensional navigation problem formed by (7.3), (7.5), and (7.8). In this problem the

number of continuous variables, n_c, is at most

$$n_c = 3n_w n_p n_q. \tag{7.9}$$

Some continuous variables can be eliminated from the MIQP by removing the equality constraints. A strong factor that determines the computational time is the number of binary variables, n_b, that are introduced. The number of binary variables for a three-dimensional navigation problem is:

$$n_b = n_w n_k n_q \prod_{o=1}^{n_o} n_f(o) + n_w n_k \frac{n_q(n_q-1)}{2} 6 \tag{7.10}$$

The first term in (7.10) accounts for the obstacle avoidance constraints and the second term represents inter-quadrotor safety distance enforcement.

In this paper, we use a branch and bound solver [3] to solve the MIQP. At a worst case there are 2^{n_b} leaves of the tree to explore. Therefore, this is not a method that scales well for large number of robots but it can generate optimal trajectories for small teams (up to 4 quadrotors in this paper) and a few obstacles. Computational times for all scenarios presented in this paper are shown in Sec. 7.4.4. One advantage with this technique is that suboptimal, feasible solutions that guarantee safety and conflict resolution can be found very quickly (compare T_1 and T_{opt} in the Table 7.1) if the available computational budget is low.

7.4 Experimental Results

The experiments presented in this paper are conducted with Ascending Technologies Hummingbird quadrotors [1] as well as the kQuad65 and kQuad1000 quadrotors developed in-house which weigh 457, 65, and 962 grams and have blade tip to blade tip lengths of 55, 21, and 67 cm, respectively. We use a Vicon motion capture system [8] to estimate the position and velocity of the quadrotors and the onboard IMU to estimate the orientation and angular velocities. The software infrastructure is described in [62].

In previous work [55], the orientation error term was computed off-board the vehicle using the orientation as measured by the motion capture system. This off-board computation introduces a variable time delay in the control loop which is significant when using with multiple quadrotors. The time delay limits the performance of the attitude controller. We choose to instead use a stiff on-board linearized attitude controller as in [56] instead of the softer off-board nonlinear attitude controller as in [55].

We solve all problems with the MIQP solver in the CPLEX software package [3]. Computational times for all scenarios presented in this paper are shown in Sec. 7.4.4.

7.4.1 Three Quadrotors in Plane with Obstacles

This experiment demonstrates planning for three vehicles in a planar scenario with obstacles. Three homogeneous Hummingbird quadrotors start on one side of a narrow gap and must pass through to goal positions on the opposite side. The trajectories were found using the method described in Sec. 7.2 using 10th order polynomials and enforcing collision constraints at 11 intermediate time steps between the two waypoints ($n_p = 10$, $n_k = 11$, $n_w = 1$). The quadrotors were then commanded to follow these trajectories at various speeds for 30 trials with a hoop placed in the environment to represent the gap. Data and images for this experiment are shown in Figs. 7.3 and 7.4. Figure 7.3(a) shows the root-mean-square errors (RMSE) for each of these trials. While trajectories with larger acceleration, jerk, and snap do cause larger errors (as expected) the performance degrades quite gracefully. The data for a single run is presented in Figs. 7.3(b-d).

7.4.2 Two Heterogeneous Quadrotors through 3-D gap

The experiment demonstrates the navigation of a kQuad1000 (Quadrotor 1) and a Hummingbird (Quadrotor 2) from positions below a gap to positions on the opposite side of the room above the gap. This problem is formulated as a 3-D trajectory generation problem

using 13th order polynomials and enforcing collision constraints at 9 intermediate time steps between the two waypoints ($n_p = 13$, $n_k = 9$, $n_w = 1$). For the problem formulation four three-dimensional rectangular prism shaped obstacles are used to create a single 3-D gap which the quadrotors must pass through to get to their goals. Data and images for these experiments are shown Figs. 7.5 and 7.6. Since the bigger quadrotor has a tighter tolerance to pass through the gap we choose to weight its cost function 10 times more than the Hummingbird. This can be observed from the more indirect route taken by the quadrotor 2 in Fig. 7.5. Also, this can be observed by the larger error for quadrotor 2 since it is following a more difficult trajectory which requires larger velocities and accelerations. Finally, one should note that the larger quadrotor follows the smaller one up through the gap because it is allowed to fly underneath the smaller one but not vice versa as described in Sec. 7.3.2.

7.4.3 Formation Reconfiguration with Four Quadrotors

This experiment demonstrates reconfiguration for teams of four quadrotors. This problem is formulated as a 3-D trajectory generation problem using 9th order polynomials and enforcing collision constraints at 9 intermediate time steps between the two waypoints ($n_p = 9$, $n_k = 9$, $n_w = 1$). Trajectories are generated which transition quadrotors between arbitrary positions in a given three-dimensional formation or to a completely different formation smoothly and quickly. We present several reconfigurations in the attached video and a single reconfiguration within a line formation in Figs. 7.7 and 7.8. We ran the experiment with four Hummingbirds and a heterogeneous team consisting of two Hummingbirds, one kQuad65, and one kQuad1000. For the heterogeneous group we weight the cost of the kQuad65 10 times larger than the other quads because it is the least agile and can presently only follow moderately aggressive trajectories. Notice how the kQuad65 takes the most direct trajectory in 7.7(b). For the homogeneous experiment shown in Fig. 7.7(a) the quadrotors stay in the same plane because they are not allowed to fly underneath

Fig.	n_q	n_p	n_k	n_b	T_1 (s)	T_{opt} (s)
7.2(b)	1	15	16	208	0.42	35
7.5	2	13	9	270	0.62	1230
7.3	3	10	11	300	0.21	553
7.7(a)	4	9	9	324	0.11	39
7.7(b)	4	9	9	324	0.45	540

Table 7.1: T_1 is the time to find the first feasible solution and T_{opt} is the time to find the optimal solution and prove its optimality.

each other as described in Sec. 7.3.2 but in the heterogeneous experiment shown in Fig. 7.7(b) the optimal solution contains z components since larger quadrotors are allowed to fly under smaller ones.

7.4.4 Solver Details

We present problem details and computational times for each of the MIQPs solved in this paper in Table 7.1. All computation times are listed for a MacBook Pro laptop with a 2.66 GHz Intel Core 2 Duo processor using the CPLEX MIQP solver [3]. Note that while certain problems take a long time to find the optimal solution and prove optimality, a first solution is always found in less than a second. The solver can be stopped any time after the first feasible is found and return a sub-optimal solution.

7.5 Concluding Remarks

We presented an algorithm for generating optimal trajectories for multiple heterogeneous quadrotors in environments with obstacles. This method can enforce constraints on positions, velocities, accelerations, jerks and inputs and allows for different sizes, capabilities, and varying dynamic effects between different quadrotors. Collision avoidance is enforced

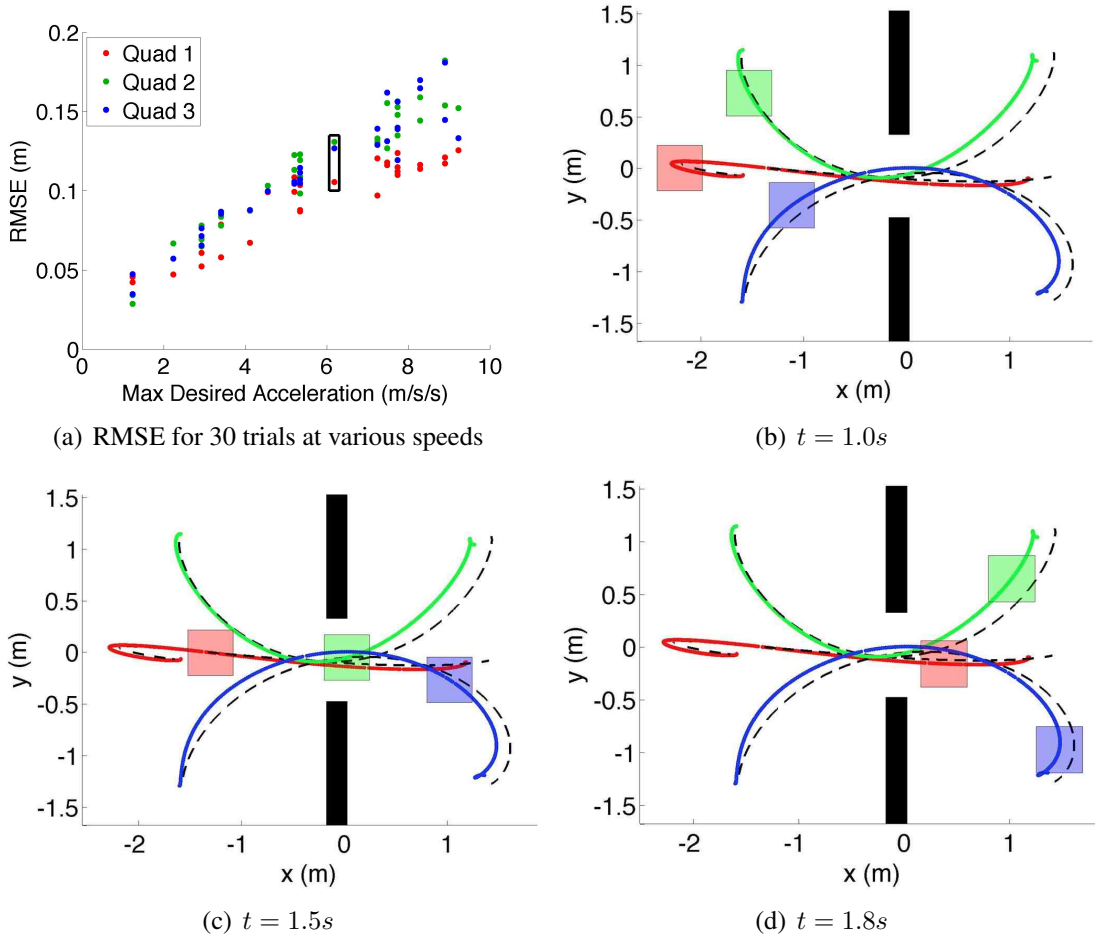
using integer constraints. The trajectories are optimal in the sense that they minimize cost functionals that are derived from the square of the norm of the snap (the fourth derivative of position). The time scaling property of this approach allows trajectories to be slowed down to be made *safer*. The method is complete in the sense that if a solution exists the method will find it.

Of course, this method is not without its limitations. We acknowledge that this is a centralized approach that requires knowledge of the start and goal positions for all agents. The computational complexity of the MIQP limits the application of this method to small teams with a small number of obstacles. Complexity also increases with the number of time steps for which collision avoidance is enforced so there is a tradeoff between plan fidelity and planning time. However, as shown in Table 7.1 suboptimal solutions are often available orders of magnitude faster than the optimal solution. Nonetheless, large teams and more complex environments will require a different planning paradigm.

It is difficult to find globally optimal solutions for large teams of vehicles. If one is willing to sacrifice global optimality then alternative approaches are available. A great deal of work has been done on methods that use local rules in attempt to reach a global goal. These methods sacrifice global optimality but gain tractability for dealing with large teams of vehicles since the computations are only done locally for individual vehicles. A disadvantage of current approaches is that they generally do not generate trajectories that are continuous up to high derivatives as is desired for quadrotors. Some examples of this type of approach are methods that use reciprocal velocity obstacles [76]. In this approach individual agents choose a velocity that is as close possible to a desired velocity and avoids collisions with other agents in the environment. This approach has been demonstrated on large numbers of agents in simulation and experimentally on large teams of differential drive robots [12]. Approaches such as this may have a lot to offer toward the problem of planning for large teams of aerial vehicles.

Many planning techniques suffer from dimensional explosion when planning for large teams of vehicles. Search based planning techniques like ARA* [52] suffer from this

problem since the dimension of the space which must be searched grows linearly with the number of vehicles in the environment. One approach to limiting the dimension of the search space is to group individual vehicles into formations. When planning, the formation can be treated as a single entity [49]. Of course, with this approach a rigid formation of vehicles cannot fit through small gaps which individual vehicles can so completeness is sacrificed. However, this approach scales to extremely large teams of vehicles since a formation can have any number of vehicles.



(a) RMSE for 30 trials at various speeds

(b) $t = 1.0s$

(c) $t = 1.5s$

(d) $t = 1.8s$

Figure 7.3: Images (b-d) show data for a single run (the boxed data in (a)). The colored boxes represent the quadrotor positions at specified times during the experiments corresponding to the snapshots in Fig. 7.4. The colored lines represent the actual quadrotor trajectories for this run while the dotted black lines represent the desired trajectories.

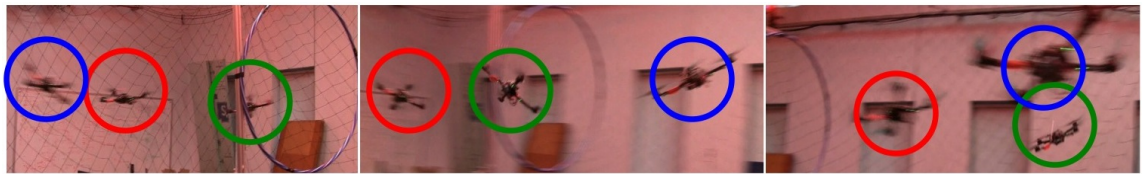
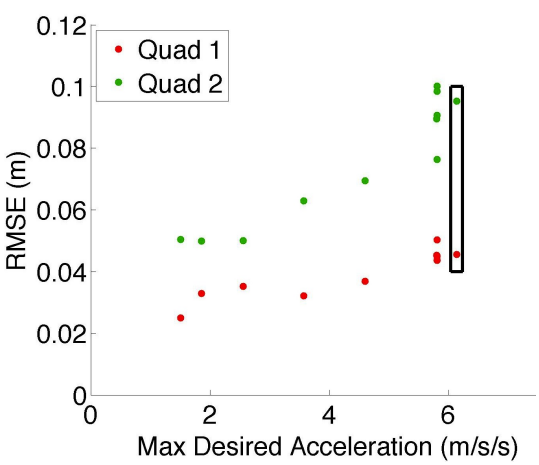
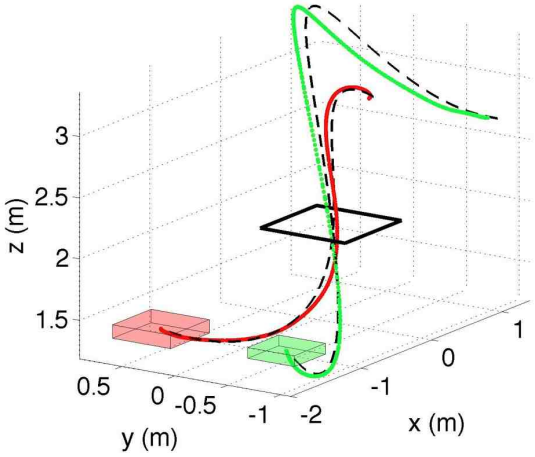


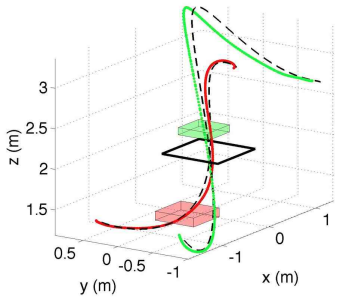
Figure 7.4: Snapshots of the three quadrotor experiment in which the hoop represents the gap. See the attached video or http://tinyurl.com/multiquad.



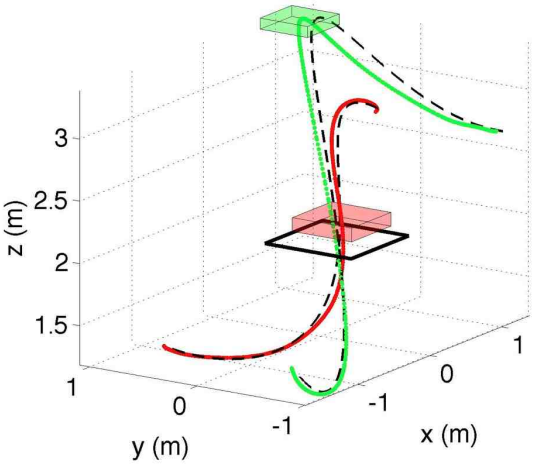
(a) RMSE for 11 trials at various speeds



(b) $t = 0.0s$



(c) $t = 1.9s$



(d) $t = 2.3s$

Figure 7.5: Images (b-d) show data for a single run (the boxed data in (a)). The colored boxes represent the quadrotor positions at specified times during the experiment corresponding to the snapshots in Fig. 7.6. The colored lines represent the actual quadrotor trajectories for this run while the dotted black lines represent the desired trajectories.

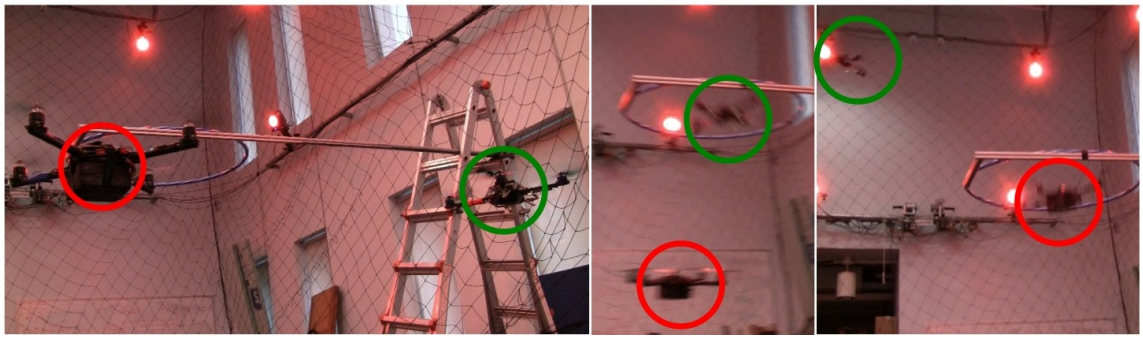
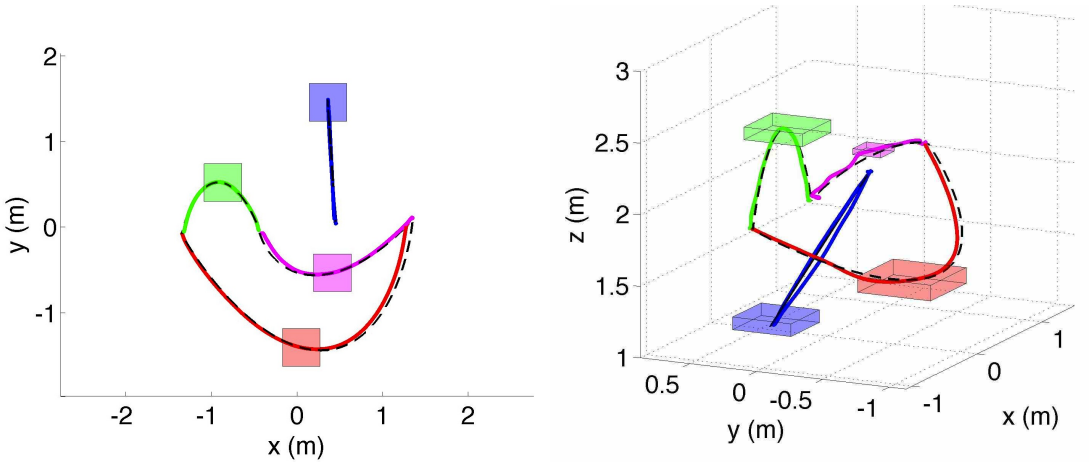


Figure 7.6: Snapshots of an experiment with the kQuad1000 (quadrotor 1, red) and the AscTec Hummingbird (quadrotor 2, green) in which the hoop represents the horizontal gap. See the attached video or http://tinyurl.com/multiquad.



(a) Four Hummingbirds - Top View

(b) kQuad1000 (red), kQuad65 (magenta), and two Hummingbirds (green and blue) - Perspective View

Figure 7.7: Trajectories for formation reconfigurations for homogeneous (a) and heterogeneous (b) quadrotor teams. The colored boxes represent the quadrotor positions at an intermediate time during the trajectories. The colored lines represent the actual quadrotor trajectories while the dotted black lines represent the desired trajectories.

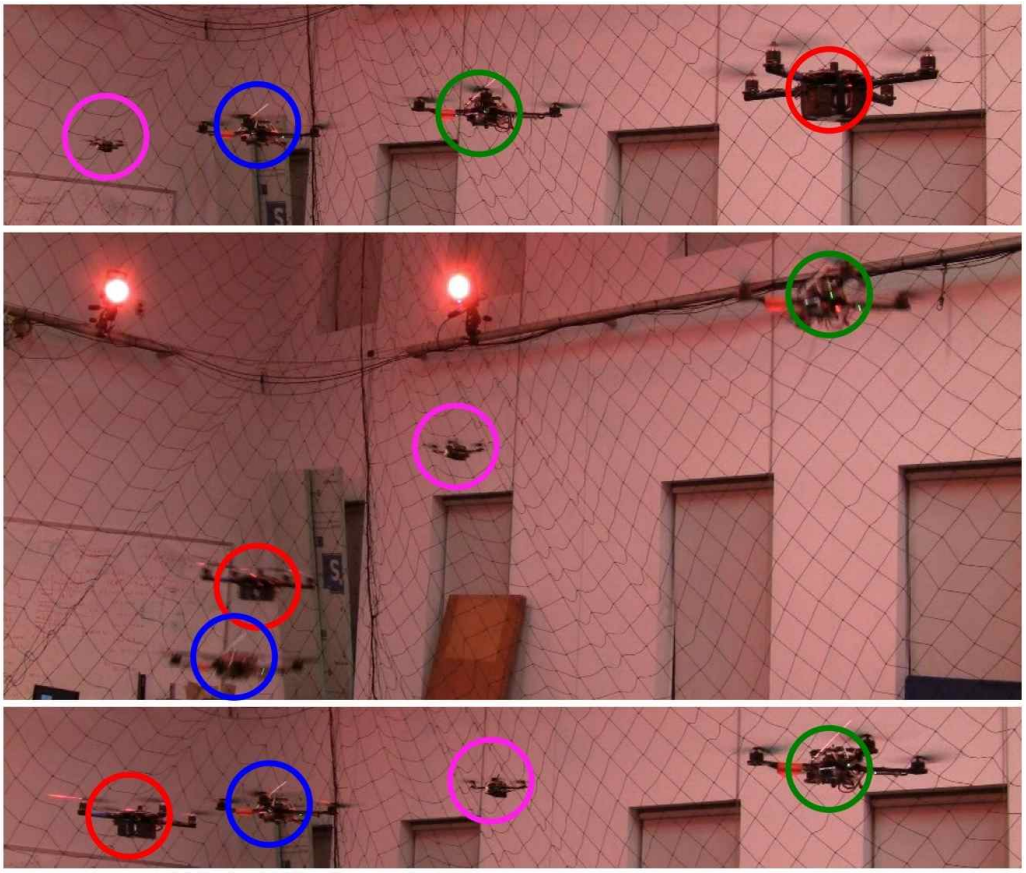


Figure 7.8: Snapshots of a four quadrotor reconfiguration within a line formation at the beginning (top), an intermediate time (middle), and the final time (bottom). The four quadrotors used are the kQuad1000 (red), the kQuad65 (magenta), and two Hummingbirds (green and blue). See the attached video or http://tinyurl.com/multiquad.

Chapter 8

Concluding Remarks

8.1 Summary of Contributions

This thesis has presented contributions to the state-of-the-art in quadrotor control, payload transportation with single and multiple quadrotors, and trajectory generation for single and multiple quadrotors. In Ch. 2 we described a controller capable of handling large roll and pitch angles that enables a quadrotor to follow trajectories requiring large accelerations and also recover from extreme initial conditions. In Ch. 3 we described a method that allows teams of quadrotors to work together to carry payloads that they could not carry individually. In Ch. 4 we discussed an online parameter estimation method for quadrotors transporting payloads which enables a quadrotor to use its dynamics in order to learn about the payload it is carrying and also adapt its control law in order to improve tracking performance.

In Ch. 5 we presented a trajectory generation method that enabled quadrotors to fly through narrow gaps at various orientations and perch on inclined surfaces. Chapter 6 discussed a method for generating dynamically optimal trajectories through a series of predefined waypoints and safe corridors and Ch. 7 extended that method to enable heterogeneous quadrotor teams to quickly rearrange formations and avoid a small number of obstacles.

8.2 Future Work

The extension of the methods described here to work outside of the controlled lab environment will make quadrotors useful in many practical scenarios. Quadrotors are beginning to be used in commercial applications for surveillance and aerial videography by simply using GPS to sense the position of the vehicle. While these vehicles probably shouldn't fly at large roll and pitch angles often they could certainly benefit from the *Large-Angle* controller described in this thesis. This controller would make the vehicles more robust to disturbances from wind and also collisions.

GPS alone is not enough to precisely position a quadrotor to pick up a payload. However, with the addition of an onboard camera the quadrotor could precisely position itself relative to objects on the ground and pick them up. This could be used for activities like roadside litter pickup, soil sampling for agriculture applications, and deployment and retrieval of lightweight sensors. A camera could also aid in landing the vehicle on small targets for perch and stare missions.

The trajectory generation methods described here are primarily for generating short time-span, fast motions where the largest constraints come from the dynamic constraints of the vehicles. One could accomplish a great deal of practical tasks without these type of maneuvers as many basic tasks require only near-hover flight. However, trajectory generation methods like the ones described here will be required in situations where fast maneuvers are required such as quickly avoiding dynamic obstacles (or other vehicles) or flying through gaps that the near-hover state will not allow. In order to fully realize this goal outside of a motion capture setup, methods for precise position and velocity estimation that allow for large roll and pitch angles and fast linear and angular velocities will be required.

All these challenges are not impossible, they just require more work. It shouldn't be too long before teams of quadrotors can fly into burning buildings through narrow gaps in walls, dodge falling debris, and cooperatively use their onboard grippers to lift babies and carry them to safety.

Bibliography

[1] Ascending Technologies, GmbH. http://www.asctec.de.

[2] Draganfly Innovations, Inc. http://www.draganfly.com.

[3] IBM ILOG CPLEX V12.1: Users manual for CPLEX, International Business Machines Corporation, 2009.

[4] MiKroKopter. http://www.mikrokopter.us.

[5] Parrot AR.Drone. http://ardrone.parrot.com.

[6] Robot Operating System (ROS). http://www.ros.org.

[7] ROS-Matlab Bridge. http://github.com/nmichael/ipc-bridge.

[8] Vicon Motion Systems, Inc. http://www.vicon.com.

[9] P. Abbeel. *Apprenticeship Learning and Reinforcement Learning with Application to Robotic Control*. PhD thesis, Stanford University, Stanford, CA, August 2008.

[10] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions*. Dover, first edition, 1964.

[11] Brian MacAllister Aleksandr Kushleyev and Maxim Likhachev. Planning for landing site selection in the aerial supply delivery. In *Proc. of the Int. Conf. on Intelligent Robots and Systems*, 2011.

[12] J Alonso-Mora, A Breitenmoser, M Rufli, P Beardsley, and R Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In A Martinoli and F Mondada, editors, *Proc. of the 10th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, Berlin, November 2010. Springer Press.

[13] E. Altug, J. Ostrowski, and C. Taylor. Control of quadrotor helicopter using dual camera visual feedback. *The Int. Journal of Robotics Research*, 24(5):329–341, May 2005.

[14] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3 – 34, 1995. Hybrid Systems.

[15] Suguru Arimoto, Sadao Kawamura, and Fumio Miyazaki. Bettering operation of robots by learning. *Journal of Robotic Systems*, 1(2):123–140, 1984.

[16] A. T. Asbeck, S. Kim, and M. R. Cutkosky. Scaling hard vertical surfaces with compliant microspine arrays. In *Proc. of Robotics: Science and Systems*, Cambridge, MA, June 2005.

[17] A. Bachrach, R. He, and N. Roy. Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 1(4):217–228, Dec 2009.

[18] S. Bayraktar and E. Feron. Experiments with small helicopter automated landings at unusual attitudes. *arXiv*, 2007.

[19] M. Bernard and K. Kondak. Generic slung load transportation system using small size helicopters. In *In Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3258–3264, 2009.

[20] A. Bicchi and V. Kumar. Robotic grasping and contact. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 348–353, San Francisco, CA, April 2000.

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

[22] M. Caccia, G. Indiveri, and G. Veruggio. Modeling and identification of open-frame variable configuration unmanned underwater vehicles. *Oceanic Engineering, IEEE Journal of*, 25(2):227 –240, April 2000.

[23] Guowei Cai, Ben Chen, and Tong Lee. An overview on development of miniature unmanned rotorcraft systems. *Frontiers of Electrical and Electronic Engineering in China*, 5:1–14, 2010. 10.1007/s11460-009-0065-3.

[24] G. Calafiore, M. Indri, and B. Bona. Robot dynamic calibration: Optimal excitation trajectories and experimental parameter estimation. *Journal of Robotic Systems*, 18(2):55–68, 2001.

[25] R. Cory and R. Tedrake. Experiments in fixed-wing uav perching. In *AIAA Guidance, Navigation, and Control Conference*, 2008.

[26] I.D. Cowling, O.A. Yakimenko, J.F. Whidborne, and A.K. Cooke. A prototype of an autonomous controller for a quadrotor uav. In *European Control Conference*, 2007.

[27] A. L. Desbiens and M. R. Cutkosky. Landing and perching on vertical surfaces with microspines for small unmanned air vehicles. *J. Intell. Robotic Syst.*, 57:313–327, January 2010.

[28] J. Fink, N. Michael, S. Kim, and V. Kumar. Planning and control for cooperative manipulation and transportation with aerial robots. In *Proc. of the Int. Symposium of Robotics Research*, Luzern, Switzerland, August 2009.

[29] T. Flash and N. Hogan. The coordination of arm movements: An experimentally confirmed mathematical model. *The Journal of Neuroscience*, 5:1688–1703, 1985.

[30] M. Gerig. *Modeling, Guidance, and Control of Aerobatic Maneuvers of an Autonomous Helicopter*. PhD thesis, ETH Zurich, 2008.

[31] Vaibhav Ghadiok, Jeremy Goldin, and Wei Ren. Autonomous indoor aerial gripping using a quadrotor. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4645 –4651, sept. 2011.

[32] J. H. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin. Design and analysis of hybrid systems, with applications to robotic aerial vehicles. In *Proc. of the Int. Symposium of Robotics Research*, Luzern, Switzerland, September 2009.

[33] 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. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1649–1654, Anchorage, AK, May 2010.

[34] D. Gurdan, J. Stumpf, M. Achtelik, K. Doth, G. Hirzinger, and D. Rus. Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Roma, Italy, April 2007.

[35] R. He, A. Bachrach, M. Achtelik, A. Geramifard, D. Gurdan, S. Prentice, J. Stumpf, and N. Roy. On the design and use of a micro air vehicle to track and avoid adversaries. *The Int. Journal of Robotics Research*, 29:529–546, 2010.

[36] R. He, S. Prentice, and N. Roy. Planning in information space for a quadrotor helicopter in a GPS-denied environment. *International Conference on Robotics and Automation*, 2008.

[37] Markus Hehn and Raffaello D'Andrea. Quadrocopter trajectory generation and control. In *IFAC World Congress*, 2011.

[38] B. Hein and I. Chopra. Hover performance of a micro air vehicle: Rotor at low reynolds number. *Journal of the American Helicopter Society*, 52(3):254–262, July 2007.

[39] Lionel Heng, Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Autonomous obstacle avoidance and maneuvering on a vision-guided mav using on-board processing. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2472 –2477, may 2011.

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

[41] P. Ioannou and B. Fidan. *Adaptive Control Tutorial*. SIAM: Society for Industrial and Applied Mathematics, 2006.

[42] P. Ioannou and J. Sun. *Robust Adaptive Control*. Prentice-Hall, first edition, 1996.

[43] A. Jadbabaie and J. Hauser. On the stability of receding horizon control with a general terminal cost. *Automatic Control, IEEE Transactions on*, 50(5):674 – 678, may. 2005.

[44] M. Kawato, Y. Maeda, Y. Uno, and R. Suzuki. Trajectory formation of arm movement by cascade neural network model based on minimum torque-change criterion. *Biological Cybernetics*, 62:275–288, 1990.

[45] H. Khalil. *Nonlinear Systems*. Prentice Hall, 1996.

[46] H.J. Kim, D.H. Shim, and S. Sastry. Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles. volume 5, pages 3576 – 3581, 2002.

[47] I. Kroo, F. Prinz, M. Shantz, P. Kunz, G. Fay, S. Cheng, T. Fabian, and C. Partridge. The mesicopter: A miniature rotorcraft concept, phase ii interim report. 2000.

[48] V. Kumar and K. J. Waldron. Analysis of omnidirectional gaits for walking machines for operation on uneven terrain. In *Proc. of Sym. on Theory and Practice of Robots and Manipulators*, pages 37–62, Udine, Italy, September 1988.

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

[50] I. D. Landau, R. Lozano, and M. M'Saad. *Adaptive Control*. Springer, first edition, 1998.

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

[52] Maxim Likhachev, Geoff Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. *Advances in Neural Information Processing Systems*, 16, 2003.

[53] Quentin Lindsey, Daniel Mellinger, and Vijay Kumar. Construction of cubic structures with quadrotor teams. *Robotics: Science and Systems*, 2011.

[54] S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1642–1648, Anchorage, AK, May 2010.

[55] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, May 2011.

[56] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers. In *Int. Symposium on Experimental Robotics*, December 2010.

[57] D. Mellinger, M. Shomin, and V. Kumar. Control of quadrotors for robust perching and landing. In *Proceedings of the International Powered Lift Conference*, Oct 2010.

[58] D. Mellinger, M. Shomin, N. Michael, and V. Kumar. Cooperative grasping and transport using multiple quadrotors. In *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems*, Nov 2010.

[59] Daniel Mellinger, Quentin Lindsey, Michael Shomin, and Vijay Kumar. Design, modeling, estimation and control for aerial grasping and manipulation. In *Proc. of the Int. Conf. on Intelligent Robots and Systems*, 2011.

[60] N. Michael, J. Fink, and V. Kumar. Cooperative manipulation and transportation with aerial robots. In *Proc. of Robotics: Science and Systems*, Seattle, WA, June 2009.

[61] N. Michael, J. Fink, and V. Kumar. Cooperative manipulation and transportation with aerial robots. *Autonomous Robots*, 30:73–86(14), January 2011.

[62] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The grasp multiple micro uav testbed. *IEEE Robotics and Automation Magazine*, September 2010.

[63] G.J. Monkman, S. Hesse, R. Steinmann, and H. Schunk. *Robot Grippers*. Wiley, Berlin, 2007.

[64] R. Oung, F. Bourgault, M. Donovan, and R. D'Andrea. The distributed flight array. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 601–607, Anchorage, AK, May 2010.

[65] D. Pines and F. Bohorquez. Challenges facing future micro air vehicle development. *AIAA Journal of Aircraft*, 43(2):290–305, 2006.

[66] Paul Pounds and Aaron Dollar. Hovering stability of helicopters with elastic constraints. In *ASME Dynamic Systems and Control Conference*, 2010.

[67] Paul Pounds, Robert Mahony, and Peter Corke. Modelling and control of a quadrotor robot. In *Australasian Conference on Robotics and Automation*, 2006.

[68] Arthur Richards, Jonathan How, Tom Schouwenaars, and Eric Feron. Plume avoidance maneuver planning using mixed integer linear programming. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2001.

[69] K. Salisbury and B. Roth. Kinematics and force analysis of articulated mechanical hands. *J. Mechanisms, Transmissions, and Automation in Design*, 105:35–41, December 1983.

[70] Tom Schouwenaars, Bart DeMoor, Eric Feron, and Jonathan How. Mixed integer programming for multi-vehicle path planning. In *European Control Conference*, pages 2603–2608, 2001.

[71] Tom Schouwenaars, Jonathan How, and Eric Feron. Receding horizon path planning with implicit safety guarantees. In *American Control Conference*, pages 5576–5581, 2004.

[72] Tom Schouwenaars, Andrew Stubbs, James Paduano, and Eric Feron. Multi-vehicle path planning for non-line of sight communication. In *American Control Conference*, 2006.

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

[74] J. Tang, A. Singh, N. Goehausen, and P. Abbeel. Parameterized maneuver learning for autonomous helicopter flight. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1142–1148, Anchorage, AK, May 2010.

[75] R. Tedrake. LQR-Trees: Feedback motion planning on sparse randomized trees. In *Proc. of Robotics: Science and Systems*, Seattle, WA, June 2009.

[76] Jur van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2008.

[77] M. J. Van Nieuwstadt and R. M. Murray. Real-time trajectory generation for differentially flat systems. *International Journal of Robust and Nonlinear Control*, 8:995–1020, 1998.

[78] J.T.-Y. Wen and K. Kreutz-Delgado. The attitude control problem. *Automatic Control, IEEE Transactions on*, 36(10):1148 –1162, oct. 1991.

[79] Jie Yu, Ali Jadbabaie, James Primbs, and Yun Huang. Comparison of nonlinear control design techniques on a model of the caltech ducted fan. In *IFAC World Congress, IFAC-2c-112*, pages 53–58, 1999.