Quadrotor Reorientation Control for Collision Recovery

Gareth Dicker

Masters

Mechanical Engineering

McGill University

Montreal, Quebec

December 15th, 2016

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Engineering

DEDICATION

To old and dead engineers who have never heard of drones.

To current engineers trying against the odds to use drones for good purposes.

To future engineers who will not recall a time when we did not own the sky.

May we take responsibility for the skyline we invent.

ACKNOWLEDGEMENTS

I would like to acknowledge my supervisor, Dr. Inna Sharf, for keeping my thinking sharp over the past few years. You successfully kept me in the field of engineering despite my occasional attempts to run off and do something impulsive like become a violinist, a monk or a high school teacher. Though I may eventually do all those things, I will probably do them better having first studied and researched under your guidance.

I would like to thank my 'dynamics duo' colleague Fiona Chui for making this masters degree fun. It would have been quite solemn without you. Also, I learned a lot from your PowerPoint, Latex, and general formatting skills. If I ever become a teacher, my students will enjoy good presentation slides.

Thanks to everyone in the Aerospace Mechatronics Laboratory. To the legends who threw down the gauntlet and showed me what good research looks like: Adam Harmat, Mingfeng Zhang, Mikkael Persson, Waqas Khan. To the ones who recently left, leaving me as the most senior lab member: Khoi Tran, Patrick Abouzakhm, Pierre-Yves Brèves, Thomas Fuehrer. To the current lab members, who are always helpful with experiments: Eitan Bulka, Adrian Battison, Bassam Haq. And to the office-room comedic relief: Luc Sagnières and Josh Levin. You are all great people.

Finally, I want to formally thank my partner Justine for a lot of breakfasts and dinners over the past two years. I'll do the dishes.

ABSTRACT

Small unmanned aerial vehicles (UAVs) are currently not safe enough to fly autonomously in public airspace. While research into obstacle avoidance for UAVs has progressed over the last decade, the question of how to recover from unanticipated collisions with objects has received little attention. This thesis will treat the question of flight control recovery for the case of a quadrotor UAV colliding with a wall. Such a scenario is relevant to UAVs flying in urban environment in proximity to buildings and structures. To motivate a collision recovery control solution, literature on aerobatic and aggressive quadrotor control theory was investigated. A 'reorientation' flight control law was selected from the literature, that was suitable for handling collision recovery situations. This control law was developed and also validated with Monte Carlo simulations. A custom quadrotor research platform with propeller protection was designed and built for experimental testing of the reorientation control law. Stabilization of the quadrotor from a wide range of unstable initial conditions was demonstrated outdoors using only onboard inertial sensing. Having demonstrated aggressive reorientation, a method for recovering from collisions was developed in simulation and validated, again with the Monte Carlo method. The collision recovery strategy was developed in conjunction with Ms. Chui, who provided methods for detection and characterization of wall collisions. The collision recovery strategy was tested on the experimental platform indoors under motion capture feedback. The experimental tests were largely successful and provide insight into the challenges inherent in recovering stable flight control of a quadrotor UAV following large disturbances and collisions.

ABRÉGÉ

Les petits véhicules aériens sans pilote (UAV) ne sont actuellement pas assez sécuritaires pour voler de façon autonome dans l'espace aérien public. Alors que la recherche sur l'évitement d'obstacles a progressé au cours de la dernière décennie, la question de savoir comment se remettre des collisions imprévues a reçu peu d'attention. Cette thèse traitera la question de la récupération du contrôle de vol pour le cas d'un quadcoptère sans pilote qui entre en collision avec un mur. Un tel scénario est pertinent pour les UAV volant en milieu urbain à proximité de bâtiments. Afin de développer une solution de contrôle récupération après une collision, nous avons étudié la littérature existante sur la théorie de contrôle acrobatique et agressif de quadcoptère. Nous avons choisi dans notre revue de littérature une loi de contrôle qui convient aux situations de récupération à la suite d'une collision. Cette loi de contrôle a été développée et validée par des simulations Monte Carlo. Nous avons conçu une plateforme de recherche personnalisée avec protection des hélices et l'avons construit pour des essais expérimentaux de la loi de contrôle de réorientation. Nous avons démontré la stabilisation du quadcoptère à partir d'une large gamme de conditions initiales instables à l'extérieur en utilisant uniquement la détection inertielle embarquée. Après avoir démontré une réorientation agressive, nous avons développé en simulation une méthode de récupération des collisions qui fut validée avec la méthode Monte Carlo. La stratégie de récupération des collisions a été élaborée en collaboration avec Mme Chui, qui a fourni des méthodes pour la détection et la caractérisation des collisions avec un mur. La stratégie de récupération de collision a été testée sur la plateforme expérimentale à l'intérieur grâce à la capture de mouvement. Les essais expérimentaux ont été largement réussis et donnent un aperçu des

défis inhérents à la récupération et à la régulation de vol d'un quadcoptère sans pilote suite à de grandes perturbations et collisions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

1.1 Background and Motivation

The micro UAV (drone) industry has grown steadily over the past decade, especially in the consumer market [1]. Maneuverable, hovering drones such as quadrotors, hexacopters and octocopters have come to the fore because stable aerial photography and videography is in demand. Improvements in areas such as micro-processing power, energy storage, inertial and visual sensing, and wireless communication, have made UAVs cheap and reliable enough for consumer purchases [2]. At the same time, businesses and organizations in fields such as agriculture, real estate, home insurance, crowd monitoring, film, humanitarian aid, infrastructure development and maintenance, and medical supply delivery, are beginning to exploit the higher level possibilities for UAVs as they gain in autonomy and reliability [3].

However, UAVs are not yet safe or reliable enough for autonomous flight in public airspace. Current FAA and Transport Canada airspace regulations do not allow drones to fly out of the line-of-sight of a trained pilot, and only one drone can be flown per pilot [4]. Autonomous flying of drones is not allowed under any circumstances. Wind gusts, loss of line-of-sight, loss of GPS satellite connection, piloting errors, and collisions with objects are all hazards which can cause UAVs to lose flight stability. UAV crashes endanger humans, creatures and objects around and below them.

A first safety measure is naturally to attempt to avoid colliding with obstacles. As computer vision algorithms improve, obstacle avoidance is becoming a safety feature on most larger quadrotor UAV platforms [5]. For instance, Intel has demonstrated a 360°

Snap Drone from Vantage Robotics | Racing Drone with Propeller Protection | AR Drone 2.0 from Parrot

Figure 1–1: Examples of propeller protected quadrotors

obstacle avoidance solution on quadrotors using multiple onboard Microsoft Kinect cameras [6]. However, obstacle avoidance is not fool proof. There are several situations which can cause it to fail. For instance, a wind gust destabilizes the cameras' field of view, an object moves too quickly toward the quadrotor for it to react in time, or multiple stationary or moving objects in proximity make collisions inevitable. Once obstacle avoidance fails, the UAV requires an additional safety measure to ensure recovery from the collision. This thesis will limit the discussion to the case of quadrotor collisions, ignoring fixed-wing and other types of UAVs.

When a quadrotor's propeller contacts another object it is immediately rendered defective and the flight controller, which requires all four propellers, will fail, unless the damaged propeller can be identified and the flight control law adjusted to allow for a stable emergency landing [7] [8]. In some cases, flight control will also fail if the quadrotor finds itself in an upside down orientation, depending on whether or not the control law is linearized about the hover orientation [9]. Since propeller collisions are the most common failure case for quadrotors, an increasing number of drone companies now offer propeller protection as a safety feature, as shown in Figure 1–1. Protection generally consists of thin and sturdy structural material either partially or fully enshrouding each of the propellers.

For low speed and level impacts, propeller protection can be sufficient to prevent failure of the flight controller. For instance, Galea et al. [10] performed repeated intentional collisions of a nano-scale quadrotor with a relatively elastic point of contact to a wall for the purpose of stippling. These small impacts did not destabilize position control of the quadrotor under external motion capture feedback. However, if a propeller protected quadrotor strikes an object rigidly at a non-level orientation, the collision can induce a large torquing moment and temporarily destabilize the flight controller. Moreover, a collision with a *stationary* object presents another challenge. For instance, if a quadrotor strikes a wall, torques into the wall and becomes vertically flush with the wall, it can become impossible to recover stable flight control away from the wall. To date, this mode of failure has not been examined in the literature, and a robust solution for recovery from such a collision remains to be developed.

1.2 Literature Review

Literature pertaining to the problem of recovering stable flight control for a quadrotor following a collision with a wall will now be reviewed in the following order: standard quadrotor dynamics and control methods; collision detection and characterization; examples of aerobatic and aggressive flight control; quadrotor state estimation; finally, the research paper from which was selected the control theory used in this thesis.

1.2.1 Standard Quadrotor Dynamics and Control

Conventional quadrotors consist of four uni-directional propellers in opposing pairs of counter-rotation. The symmetric geometry of quadrotors makes control of their Euler angles - roll, pitch and yaw - independent, having three moments and three rotational degrees of freedom [11]. Quadrotors can only remain stationary while in the orientation of hover due to the presence of gravity which the total propeller thrust must oppose in order to hold altitude.

Proportional-Integral-Derivative (PID) and Linear Quadratic Regulator (LQR) controllers are most commonly used for controlling attitudes near to hover [12]. If the quadrotor's position or velocity is to be controlled, then the higher level errors can be cascaded as set-points to the lower level attitude and angular rates, as demonstrated by Zhang et al. [13].

1.2.2 Collision Detection and Characterization

A key question for collision recovery control is how to detect that a collision has occurred. Using only onboard sensing, it can be challenging to distinguish between collisions with objects and wind gusts or intentional high accelerations. There is little literature on this subject. Assuming the collision has been detected, the next question is how to characterize the collision in a way that is informative for the flight control reaction. Tomic et al. [14] present a method for estimating the external wrench induced by contact between a quadrotor and a wall. This study only considers the case of sustained contact following a low velocity collision. In another study by McKinnon et al. [15], the external wrench is computed for the case of strong wind gusts. Neither of these estimators deals with the significant spikes in accelerometer and gyroscope measurements which result from sudden impacts, and so cannot be employed directly to solve our research problem. However, they incorporate estimates of the forces and moments generated by the propellers. This is important information in designing a recovery controller that can characterize scenarios such as the difference between a collision with a wall vs. a cable or pole.

The characterization of the intensity of a collision is a challenging estimation task, especially with the limited sensing available on most quadrotor platforms. Given that there are several possible measures of collision intensity, a characterization of intensity is a suitable problem for using fuzzy logic. The fuzzy logic process (FLP) presented by Chui [16] was used to characterize the collisions investigated in this thesis.

1.2.3 Aerobatic and Aggressive Control

A robust recovery controller should be able to handle the quadrotor being in an upside down orientation, so flight control laws which are linearized about the state of hover cannot be used [9]. Conventional flight controllers, especially for larger quadrotors that do not anticipate being upside down, generally are only stable for roll and pitch angles within approximately $\pm 45^{\circ}$. Often, greater stability and precision can be achieved if lower level control is also performed on the angular rates or body rates of the quadrotor. In order to extend the range of attitudes that the quadrotor can track, several control methods can be applied, such as sliding mode control. Sliding mode control continuously changes its control gains depending on the location of variables in the quadrotor's state space. Bouadi et al. [17] demonstrated a sliding mode control law for a quadrotor in simulation.

Non-linear controllers have been developed to track any attitude on $SO(3)$ while constraining the motion in $SE(3)$ [18]. Lee et al. [19] have shown in simulation that a quadrotor can perform fully upside down spiraling maneuvers while also translating along one Cartesian axis. Brescianini et al. [20] have demonstrated complex tasks such as flipping an inverted pole from one quadrotor to another have been demonstrated under motion capture feedback, where the quadrotor throwing the pole performs a full flip upon release of the pole. Nonlinear controllers such as these require a global description of attitude for which unit quaternions are most commonly used as they do not suffer from the gimbal lock problem of Euler angles [21].

A quadrotor, having only four actuators, can only track four degrees of freedom [22]. If control of the quadrotor's position is desired, the most common set-points are its Cartesian position as well as a heading angle. These four control set-points are sometimes called differentially flat parameters because the lower level control of attitude is differentiable from

the higher level control of position and heading [23]. Specifically, the time derivative of the horizontal position set-point is used to generate set-points for roll and pitch angles, whose time derivatives are, in turn, used to generate angular or body rate set-points. Lyapunov stability for the four levels of control of position, linear velocity, attitude, and body rates can be proven for the backstepping controller [24].

Employing the concept of differential flatness, aggressive trajectories have been generated using the 'minimum snap' method developed by Mellinger and Kumar [25] which allows for both positional and angular set-points to be specified at a point in time by following a particular path leading up to that moment. This method is useful, for instance, for passing through narrow gaps at steep angles. Using motion capture feedback, aerobatic motions such as double flips and upside down perching have been demonstrated by Lupashin et al. [26]. Gillula et al. [27] have achieved back-flips by dividing the state space into three separate modes and performing a different control task in each mode. Similar maneuvers have been shown in the work of Lee et al. [28], switching between attitude, position and velocity control at different moments during the maneuver.

1.2.4 State Estimation

The above aerobatic maneuvers and trajectory planning methods require precise position and velocity estimates. In recent years, position and velocity estimates have been developed using both single and multiple onboard cameras [29]. Optical flow has been widely implemented on single downward facing cameras for quadrotors hovering a few meters above the ground [30]. Optical flow provides a velocity estimate reliably in the horizontal direction, and somewhat less reliably in the vertical direction.

Generally, for precise position estimates, multiple cameras are needed. A good solution to this problem comes in the form of simultaneous localization and mapping (SLAM), which is

now widely implemented on quadrotors [31]. There are numerous variants to SLAM, but also numerous alternatives. For instance, a novel multi-camera position tracking and mapping (MCPTAM) solution was recently developed in the Aerospace Mechatronics Laboratory at McGill University using multiple wide-lens cameras placed arbitrarily on a quadrotor, in the work of Harmat et al. [32].

If a quadrotor is spinning in an uncontrolled manner, vision-based position estimation using methods such as SLAM and MCPTAM will fail [33]. GPS measurements are not precise enough for aerobatic position control, and motion capture systems such as Vicon or OptiTrack lose track of the vehicle if it rotates too quickly.

If a quadrotor does not have a position or velocity estimate available, it is impossible to perform high precision maneuvers. Without position and velocity measurements, state estimators still have access to onboard accelerometer, gyroscope, magnetometer and barometer measurements which they use to estimate the attitude and altitude [34] of the quadrotor. Complementary filters [35] and Kalman filters [36] are most commonly used for these estimates. Open source flight control units such as the Pixhawk and the Naze32 use these methods effectively [37]. Still, at higher angular velocities their performance can degrade [38] and quadrotor state estimation for aggressive maneuvering with onboard sensing only is an active research topic.

1.2.5 Reinitializing Positional Control

A recovery controller cannot rely on the availability of a position estimate; before it can achieve position control under computer vision, motion capture or GPS feedback, it must first stabilize its attitude to near hover. In particular, the roll and pitch angles must be stabilized, whereas hovering is invariant to the yaw angle [18]. The yaw rate, however, needs to be relatively stable, i.e. the quadrotor needs to not be spinning about its thrusting

axis. The control of roll and pitch angles depends on the thrust differentials generated by propellers while the control of yaw and yaw rate depends on the generated *moments*. Propeller moments in general are an order of magnitude weaker than propeller thrusts, and hence, the orientation control can be split into two parts: roll and pitch control, and yaw and yaw rate control.

This type of control was demonstrated for the case of reinitializing stable flight control in the work of Faessler et al. [39], which forms the basis for the control law presented in this thesis. The aforementioned paper presents a three stage approach to recovering full position control of a quadrotor using only onboard sensing. First, the roll and pitch angles are controlled to zero so that the quadrotor is stabilized in the hover orientation. Next, the vertical velocity is stabilized using an estimate provided by an onboard barometer and accelerometer. Finally, horizontal velocity is brought to zero using a velocity estimate from optical flow performed with a downward facing camera.

1.3 Research Objectives

This thesis addresses the problem of a propeller protected quadrotor colliding with a wall under a certain range of incoming velocities and orientations, and develops a flight control solution that attempts to recover from these collisions.

To limit the experimental complexity, the research assumes that the wall is flat, vertical and stationary, and that Coulomb friction exists between the quadrotor and the wall. The pre-collision velocities and orientations are limited to within some range that can be considered aggressive but not destructive to the propeller protection or propellers. It is assumed that all four propellers remain fully functional throughout the duration of the recovery period.

There are four research objectives, two of which are for simulation and two of which are for experimental implementations:

1) First, the control strategy shown in [39] will be tested in simulation using the dynamics model found in Chui et al. [40]. Using the Monte Carlo method on a wide range of initial conditions, it will be shown that the quadrotor can bring its attitude to hover and its linear velocities to zero following a stage-by-stage process of recovery.

2) Next, the attitude and altitude control stages from [39] will be implemented on an experimental quadrotor which is to be custom built with propeller protection. To validate the control law from the first objective, the quadrotor will be thrown into the air under a wide range of initial conditions. The quadrotor is required to detect that it has entered a state of free-fall, stabilize its attitude, and then stabilize its altitude using only onboard sensing.

3) Having validated the control law in objectives (1) and (2), a recovery control strategy will be developed in simulation using the contact model of [40] which includes detection and characterization of a collision with a wall. Monte Carlo simulations will demonstrate the range of initial conditions for which recovery from a collision is feasible using the proposed control strategy.

4) Finally, the recovery control strategy will be implemented on the same quadrotor used in objective (2) for a certain testable range of initial conditions. The extent to which this objective can be achieved depends on the success of the previous three objectives. In the ideal case, onboard sensing will be used for state estimation and a motion capture system will be used to measure the position and velocity of the quadrotor for experimental analysis.

1.4 Thesis Overview

The organization of the thesis is as follows. Chapter 2 presents the control theory which is used throughout the remainder of the thesis. In Chapter 3, preliminary objectives (1) and (2) are demonstrated in order with Monte Carlo simulation results, with the design of the custom quadrotor platform, and outdoor experimental testing. Chapter 4 develops the recovery control strategy, presents results from Monte Carlo simulation of collisions, and discusses unavoidable failure modes with suggestions for improvement. Chapter 5 covers the experimental portion of recovery control under motion capture, including analysis of successful and failed recoveries. Finally, Chapter 6 summarizes the research, draws several conclusions, and lists a few directions that could be taken up as future work.

CHAPTER 2
Reorientation Control Theory

This chapter focuses on the development of a control law that can recover from destabilization upon collision. First, the design requirements for such a controller are discussed in Section 2.1. Next, the quadrotor dynamics model used in simulation is presented in Section 2.2 along with a discussion of the concepts of underactuation and non-holonomy as they apply to quadrotors. In Section 2.3, we consider the feasibility of a control law that attempts to track a reference acceleration, which is the chosen setpoint for the recovery control law. Finally, in Section 2.4, the 'reorientation' control law is presented in full.

2.1 Controller Design Requirements

There are several requirements for a control law that can recover from collisions. The first requirement is that the controller must be able to handle any orientation of the quadrotor, including upside down orientations. For the duration of time that the quadrotor is upside down, significant altitude loss will be incurred because the quadrotor's propellers can only generate downward thrust. Altitude loss is obviously undesirable in proximity of the ground or other airspace zone boundaries, and so minimization of altitude loss constitutes a second requirement. Altitude loss will be minimized if the recovery time is short, which becomes a final requirement for recovery control. To achieve these three requirements, an aggressive attitude control law is needed. The control law should exploit all of the quadrotor's available thrusting force in its attempt to quickly regain stable flight control.

The physical presence of the collided obstacle presents a further constraint on the problem of recovery. In the case of a collision with a wall, the presence of the wall cuts the

'recoverable space' in half, as any unintelligent attempts to recover into the wall will fail. This motivates the need for an estimate of the wall's normal direction so that the quadrotor can be informed as to which way it will be able to recover. The case of a collision with a pole is somewhat less physically restrictive, and situations such as grazing collisions could complicate the estimate of a normal direction.

Due to the impulsive disturbance of the collision, onboard state estimates of position and linear velocity are rendered unreliable. With position and velocity set-point tracking impossible, the control law must initially track attitude alone. If the quadrotor is upright and stable, then altitude can be tracked as well. If altitude is also stable, then it becomes possible to reintroduce velocity and position control. This order of 'reinitialization' motivates a stage-by-stage process of recovery.

A suitable control law and stage-by-stage method has been developed by Faessler et al. [39], which satisfies all of the above requirements and constraints.

2.2 Review of Quadrotor Dynamics

Quadrotor dynamics have been thoroughly researched in the past decade [41] and we review the relevant aspects with the view to introducing the nomenclature for the control of a quadrotor, which will be used throughout the remainder of the thesis. Of particular interest are the quadrotor's under-actuated and non-holonomic nature which limit the quadrotor's possibilities for recovery control.

2.2.1 Attitude Representation

A quadrotor's attitude can be described in several ways. Euler angles are most commonly used because they conveniently separate the attitude into three consecutive rotations. For example, first a yaw rotation, then a pitch rotation, then a roll rotation. However, Euler angles only uniquely describe the attitude when the quadrotor's orientation is such that all

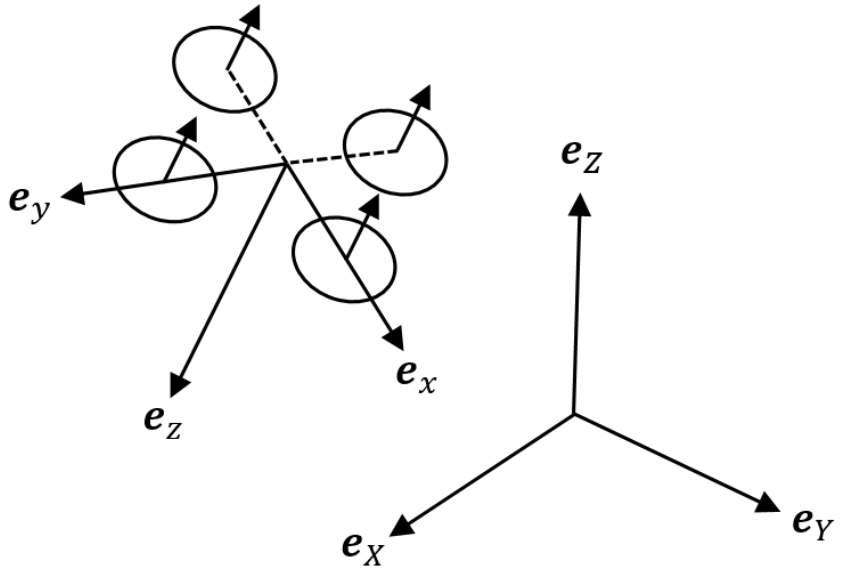


Figure 2–1: Body-fixed and inertial frames

four propellers are pointing upward, i.e., when it has not yet pitched or rolled up to 90°. If 90° rotations are allowed, Euler angles suffer from what is known as gimbal lock [42] which occurs whenever any two of the Euler angle rotations align along the same axis. For this reason, any attitude description using only three base elements is insufficient for a global description of the attitude. Instead, a description containing four elements, such as a unit quaternion, is required.

2.2.2 Quadrotor Dynamics Nomenclature

We begin by defining the inertial frame $\mathcal{F}_I = \{\mathbf{e}_X\ \mathbf{e}_Y\ \mathbf{e}_Z\}$ and body-fixed frame $\mathcal{F}_Q = \{\mathbf{e}_x\ \mathbf{e}_y\ \mathbf{e}_z\}$, where I denotes 'inertial' and Q denotes 'quadrotor'. In general, uppercase letters are used for components in $\mathcal{F}_I$ and lowercase for $\mathcal{F}_Q$.

The body-fixed frame is centered at the quadrotor's center of mass and is oriented such that $\mathbf{e}_x$ points outwards from the front of the vehicle and $\mathbf{e}_z$ points downwards when in the orientation of hover, following the North-East-Down (NED) convention.

The attitude of the quadrotor relative to $\mathcal{F}_I$ in quaternion form is given as $\mathbf{q} = [q^{(w)}\ q^{(x)}\ q^{(y)}\ q^{(z)}]^T$ where superscripts in parentheses represent elements of the quaternion. The attitude quaternion takes on the identity $[1\ 0\ 0\ 0]^T$ when the quadrotor is in the hover orientation. Vectors are rotated between $\mathcal{F}_Q$ and $\mathcal{F}_I$ via the quaternion rotation operator $\odot$[39].

Where Euler angles are used, we employ the conventional Greek letters $[\phi\ \theta\ \psi]^T$ for roll, pitch and yaw, in order. Although Euler angles cannot be used for control because of gimbal lock, they can always be computed directly from the quaternion form of the attitude using

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \tan^{-1}(2(q^{(w)}q^{(x)} + q^{(y)}q^{(z)})/(1 - 2(q^{(x)^2} + q^{(y)^2}))) \\ \sin^{-1}(2(q^{(w)}q^{(y)} - q^{(z)}q^{(x)})) \\ \tan^{-1}(2(q^{(w)}q^{(z)} + q^{(x)}q^{(y)})/(1 - 2(q^{(y)^2} + q^{(z)^2}))) \end{bmatrix}. \tag{2.1}$$

The relevant kinematic variables will now be defined. The position of the quadrotor in $\mathcal{F}_I$ is $\mathbf{p} = [X\ Y\ Z]^T$, its velocity in $\mathcal{F}_Q$ is $\mathbf{v} = [u\ v\ w]^T$, its velocity in $\mathcal{F}_I$ is $\dot{\mathbf{p}} = [\dot{X}\ \dot{Y}\ \dot{Z}]^T$, its acceleration in $\mathcal{F}_I$ is $\mathbf{a} = [\ddot{X}\ \ddot{Y}\ \ddot{Z}]^T$, and the body rates in $\mathcal{F}_Q$ are combined in $\boldsymbol{\omega} = [p\ q\ r]^T$.

Similar to the relationship between the quaternion and Euler angle representation of attitude, the relationship between Euler angle *rates* and body rates is given as [1]

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\cos\phi\sin\theta \\ 0 & 1 & \sin\phi \\ \sin\theta & 0 & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \tag{2.2}$$

There are four control outputs, the first of which is the desired thrust from the four propellers and the rest of which are the desired moments about the body-fixed axes in $\mathcal{F}_Q$. We name them $\mathbf{U} = [U_1\ U_2\ U_3\ U_4]$, where U_1 is the total thrust command and $U_{2..4}$ are the desired moments. The individual propeller speeds $\mathbf{\Omega} = [\Omega_1\ \Omega_2\ \Omega_3\ \Omega_4]$, typically measured in RPM, are mapped to these outputs via a matrix containing the thrust and torque coefficients of the propellers, k_t and k_d respectively. We denote with l the length of one of the quadrotor's arms, which are assumed to be of equal length. We assume a quadrotor in the 'x' configuration, where propellers 1 and 3 are rotating counterclockwise and propellers 2 and 4 are rotating clockwise. The transformation between control outputs and propeller speeds would be different for a quadrotor whose axes are aligned in a 't' configuration. The

[1] Although angular rates are distinct from body rates, when referring to the body rates we will allow the terms 'roll rate', 'pitch rate' and 'yaw rate' to be used loosely, for convenience.

transformation for the 'x' configuration is given as

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} k_t & k_t & k_t & k_t \\ -k_t l/\sqrt{2} & -k_t l/\sqrt{2} & k_t l/\sqrt{2} & k_t l/\sqrt{2} \\ k_t l/\sqrt{2} & k_t l/\sqrt{2} & -k_t l/\sqrt{2} & -k_t l/\sqrt{2} \\ -k_d & k_d & -k_d & k_d \end{bmatrix} \begin{bmatrix} {\Omega_1}^2 \\ {\Omega_2}^2 \\ {\Omega_3}^2 \\ {\Omega_4}^2 \end{bmatrix}. \tag{2.3}$$

On the experimental platform, this transformation from $\mathbf{U}$ to $\mathbf{\Omega}$ is handled by a mixer which sends PWM signals to motor controllers which in turn apply the correct voltage to each motor.

2.2.3 Equations of Motion

The Newton-Euler equations used to simulate the quadrotor under the reorientation controller presented in this thesis can now be formulated. First, we define the thrust vector in $\mathcal{F}_Q$ as

$$\mathbf{F}_T = [0 \ 0 \ -U_1]^T. \tag{2.4}$$

The force of gravity in $\mathcal{F}_Q$ is

$$\mathbf{F}_G = \mathbf{q}^{-1} \odot [0 \ 0 \ -g]^T, \tag{2.5}$$

and any contact forces between the quadrotor and external objects are $\mathbf{F}_C$. In the context of this research, these include both the normal and frictional forces between the quadrotor and the wall. Then we can write Newton's second law, expressed in $\mathcal{F}_Q$, as

$$m\dot{\mathbf{v}} + \boldsymbol{\omega}^{\times}\mathbf{v} = \mathbf{F}_T + \mathbf{F}_G + \mathbf{F}_C. \tag{2.6}$$

Here, $^{\times}$ denotes the cross product, and m is the quadrotor's mass.

For the Euler equation, we write

$$\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega}^{\times}\mathbf{I}\boldsymbol{\omega} = \mathbf{M}_T + \mathbf{M}_\Omega + \mathbf{M}_C. \tag{2.7}$$

Here, $\mathbf{I}$ is the inertia of the quadrotor about its center of mass and the moments from thrusting are

$$\mathbf{M}_T = [U_2\ U_3\ U_4]^T. \tag{2.8}$$

The term $\mathbf{M}_\Omega$ combines the gyroscopic moment resulting from propeller acceleration with the drag moment generated by the propellers, and $\mathbf{M}_C$ denotes the moments caused by contact with external objects, described in more detail in [40].

2.2.4 Underactuation and Non-Holonomy

A quadrotor has four control outputs $\mathbf{U}$ which generate four motor commands $\boldsymbol{\Omega}$. However, the quadrotor has a total of six degrees of freedom between the Euclidean linear space $SE(3)$ and rotational space $SO(3)$. Because the quadrotor has two fewer control outputs than its degrees of freedom, it is an under-actuated system. This means that it is impossible to control all six degrees of freedom simultaneously. Instead, a choice must be made for which four degrees of freedom should be controlled by the four control outputs. The most common choices are the quadrotor's position $\mathbf{p}$ and yaw angle ψ or yaw rate $\dot{\psi}$. When these are the degrees of freedom to be tracked, the quadrotor's other angles ϕ and θ are constrained.

When a system's state depends on the path taken in order to achieve it, the system is said to be non-holonomic. We find that this is the case for a quadrotor, whose angular state must pass through a particular path to achieve a certain positional state. For instance, to move horizontally, the quadrotor must first tilt forward, then tilt backward, and finally bring its roll and pitch angles to zero to remain stationary. This is due to the parallel alignment of all four propellers, whose thrusts point in the same direction. In addition to under-actuation,

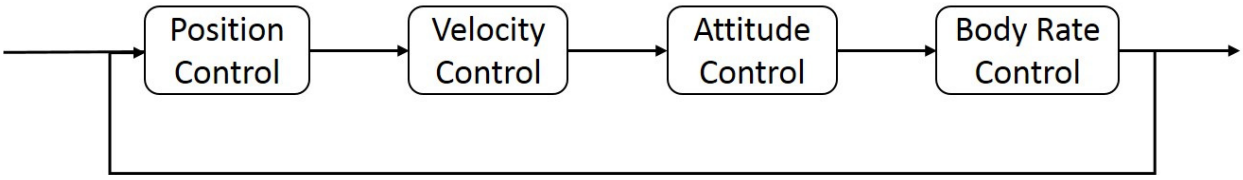


Figure 2–2: Standard position control

the quadrotor's non-holonomic nature forces the controller to either control $\mathbf{p}$ and ψ while constraining the remaining angles, or leave the position uncontrolled and control only $SO(3)$ and one other degree of freedom. The control law considered in this thesis follows the latter choice, where the additional degree of freedom is on $\mathbf{e}_z$.

Body rates $\boldsymbol{\omega}$ are measured by the on-board gyroscope, and their control is directly related to the moments generated by the propeller thrust differences. A standard flight controller usually has four levels of control. The highest level is position control, followed by control of linear velocity, followed by control of attitude, followed by body rate control. If position and velocity control are rendered impossible due to the loss of their state estimate, then only attitude and body rate control are possible. The body rate controller can receive an attitude set-point and generate desired body rates from attitude errors. This is the case for the reorientation control law presented in Section 2.4.

2.3 Tracking a Reference Acceleration

The control law in Section 2.4 is designed to track an acceleration for the center of mass of the quadrotor. The feasibility of such a control law will first be considered.

There are several limiting factors in a control law that attempts to track a reference acceleration. The physical parameters of a quadrotor, such as the ratio of its maximum available thrust and torque to its mass and moments of inertia, respectively, determine to

what degree tracking a reference acceleration is feasible. By analyzing the various influencing factors in detail, we can develop a practical understanding of this control task.

Since quadrotors are under-actuated systems, it is only possible to specify set-points for linear kinematics or for orientation, but not for both at the same time. The linear kinematic set-point can be either a position, a velocity or an acceleration of the center of mass of the quadrotor. In our case, it is an acceleration set-point. The orientation, which in this thesis will be taken to mean the roll and pitch components of the attitude, is then constrained to achieve the reference acceleration. The yaw component of the attitude does not affect the acceleration since the quadrotor's thrust is strictly along the $\mathbf{e}_z$ direction.[2] Unlike position and velocity set-points, acceleration set-points can be achieved as soon as the thrust is applied in the correct direction and magnitude. For this reason, tracking a reference acceleration can be thought of as an 'instantaneous' task: it will be achieved very quickly and may not need to be tracked for more than an instant. For any non-zero reference accelerations, the quadrotor will soon accumulate velocity and at a certain point the aerodynamic effects of high speed motion will become significant.

With these considerations in mind, the task of tracking a reference acceleration can be split into two separate tasks. The first task of the quadrotor is to achieve its orientation set-point so that thrust can be applied in the correct direction. The second task is then to apply the correct amount of thrust in that direction such that the quadrotor tracks its

[2] However, moment generation is affected by whether the quadrotor is rotating in an 'x' or 't' configuration, i.e., whether all four propellers are generating the moment or only two of the four, in the case of the 't' configuration.

reference acceleration. For clarity, we will refer to these tasks respectively as the *orientation task* and the *thrusting task*.

There are at least three complicating factors which create undesirable acceleration of the quadrotor. The first is the presence of gravity, which provides an inertially constant offset to any thrust generated by propellers. The second is the saturation limit of the thrust that can be generated by the propellers. The third is the uni-directionality of the propeller rotations. A fourth complication is the maximum value at which the propellers can accelerate, but for brevity, it will not be considered here. In summary, the three factors are:

- The presence of gravity,
- saturation limits of thrust and moments,
- and uni-directionality of the propellers.

During the orientation task, the quadrotor generates moments about its body-fixed frame axes $\mathbf{e}_x$ and $\mathbf{e}_y$ according to its control law. In the absence of saturation limits on the moments, the quadrotor can reorient itself instantaneously. If saturation limits on the thrust and moments are finite, then the orientation task will take some finite amount of time. If the propellers could spin in both directions, then moments would not induce any acceleration of the center of mass. Also in this case, the reorientation task would be faster for the same saturation limits.

If propellers are uni-directional, they will induce some amount of acceleration $\mathbf{a}$ of the center of mass, as depicted in Figure 2–3. If gravity exists, this will also cause undesirable downward acceleration during reorientation.

If gravity did not exist, then the first control output would be simply

$$U_1 = m \cdot ||\mathbf{a}_{ref}||. \tag{2.9}$$

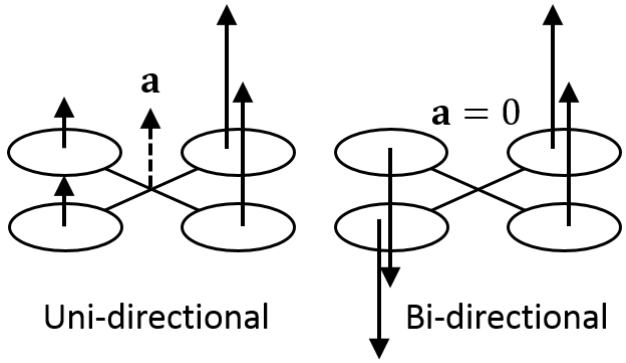


Figure 2–3: Uni-directional vs. bi-directional propellers

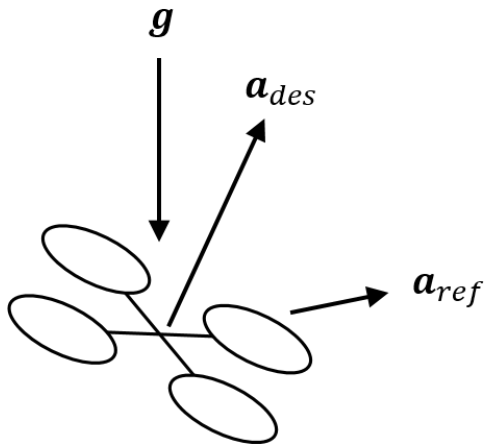


Figure 2–4: Desired and reference acceleration example

Here, $\mathbf{a}_{ref}$ is the reference acceleration to be tracked. In this case, the quadrotor would orient its $\mathbf{e}_z$ axis parallel to $\mathbf{a}_{ref}$. However, in the presence of gravity, the desired direction of thrust must be offset, as shown in Figure 2–4, such that its resultant with gravity produces the reference acceleration. We define the variable of desired acceleration $\mathbf{a}_{des}$ in $\mathcal{F}_I$ as distinct from the reference acceleration $\mathbf{a}_{ref}$ in that it always points in the desired direction of thrust. It is given by

$$U_1 = m \cdot ||\mathbf{a}_{des}|| = m \cdot ||\mathbf{a}_{ref} - \mathbf{g}||. \tag{2.10}$$

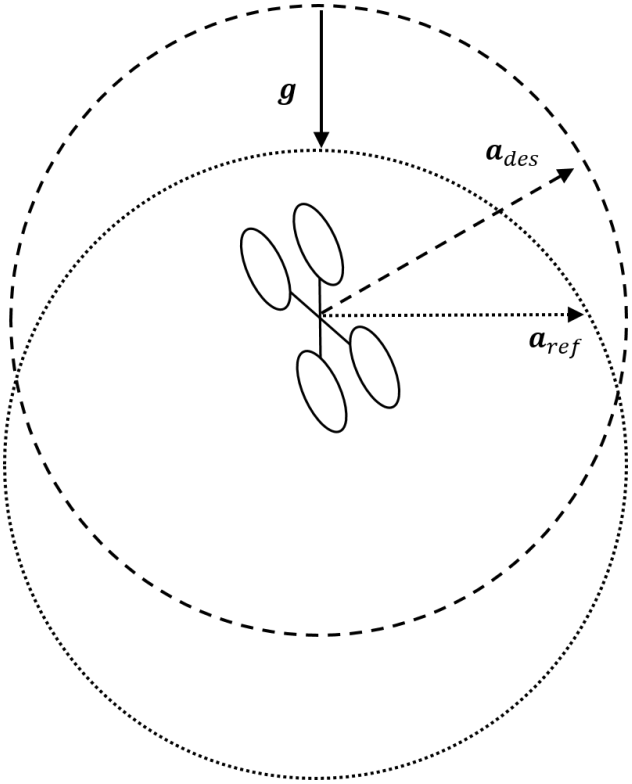


Figure 2–5: Desired and reference acceleration ranges

If there is a saturation limit on the thrust, this creates a boundary of possible accelerations that can be tracked. Figure 2–5 depicts this allowable range for the desired and the reference accelerations under the influence of gravity, where the dashed line demarcates the bound on $\mathbf{a}_{des}$ and the dotted line demarcates the resulting bound on $\mathbf{a}_{ref}$.

If the propellers were bi-directional the quadrotor could always obtain two possible orientations for any given reference acceleration, which are 180° different from each other. However, if the last constraint of uni-directional propellers is assumed, then there is only a single orientation that can achieve the desired thrust. In practice, this is not an issue as long as the reference acceleration does not change discontinuously. If it were, for example, to flip 180°, then the constraint of uni-directional propellers would force the quadrotor to re-orient

itself instead of more simply switching the direction of thrust. This consideration motivates tracking only continuous reference accelerations.

It is clear from the above considerations that a quadrotor that has the largest thrust and moments relative to its mass and inertia will minimize the undesirable acceleration induced during the orientation task and have the largest range of track-able reference accelerations for the thrusting task. Most quadrotor platforms have saturation limits on uni-directional propellers, so we can assume that no control law will be able to entirely eliminate undesirable acceleration during reorientation. Still, the minimum RPM of quadrotor propellers is usually significantly higher than zero, so minimizing this lower bound on propeller RPM will help reduce undesired acceleration.

It is important to stress that although the control is formulated to track a reference acceleration, the undesirable acceleration induced during the orientation task makes it difficult to confirm the precision with which the controller is actually able to move its center of mass at that desired acceleration. Moreover, the thrusting task occurs only for an brief period of time, so it is difficult to demonstrate that the reference acceleration has been precisely tracked. For this reason, other kinematic variables such as body rates and Euler angles serve better to demonstrate the performance of the orientation task.

A practical question for the thrusting task is when to begin applying the desired thrust? One option is to apply the correct thrust as soon as the orientation has come within certain bounds of its set-point. Another option is to scale the thrust with the dot product between the desired and actual $\mathbf{e}_z$ axes, as follows

$$U_1 = ||\mathbf{a}_{des}|| \ (\mathbf{e}_{z,des} \cdot \mathbf{e}_z). \tag{2.11}$$

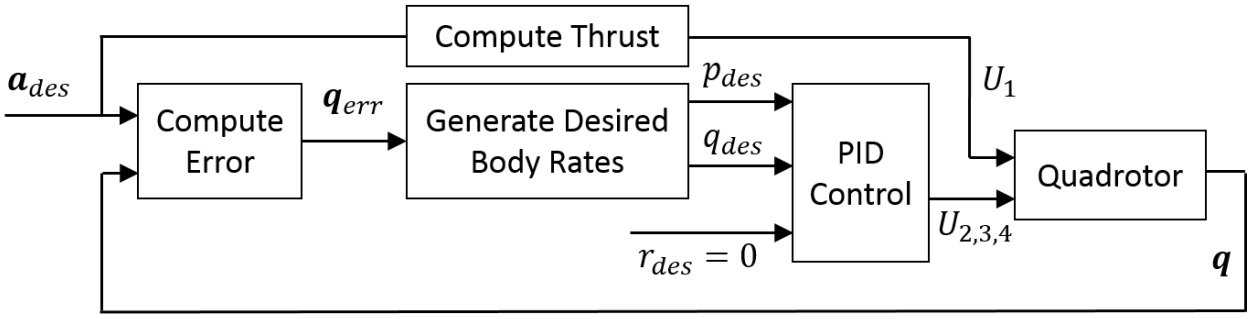


Figure 2–6: Reorientation control structure

Here, $\mathbf{e}_{z,des}$ is the desired direction of $\mathbf{e}_z$. The latter option was chosen for simulation, although modifications were made for implementation, which will be outlined in Chapter 3.

2.4 Reorientation Control Law

In the context of the practical limitations outlined above, the reorientation control law will now be presented. The controller feeds a quaternion error to a body rate controller. Body rates are then controlled directly by the moments generated by propeller speed differences. Thus the control law is cascaded where the high level is reorientation control and low level is body rate control. The overall control block diagram is depicted in Figure 2–6.

The motivating requirement for the reorientation controller is to find the fastest way to achieve the desired orientation. We need to rotate about the body-fixed axes $\mathbf{e}_x$ and $\mathbf{e}_y$ as fast as possible. Also, we need to ensure that rotation about $\mathbf{e}_z$ is minimal. This is because rotations about $\mathbf{e}_x$ and $\mathbf{e}_y$ are controlled by thrust differentials between pairs of opposite propellers. If the locations of the propellers are changing quickly in the inertial frame, then the actual thrust each propeller needs to apply to generate the appropriate thrust differential will oscillate. The faster the rotation about $\mathbf{e}_z$, the faster this oscillation of thrust must be. This becomes a practical problem because propellers have acceleration limits. In fact, if

body rate r was extremely large, it would be sensible to first only control r until it reached some minimum threshold, and only then begin to control body rates p and q.

The error in orientation is scaled to generate desired body rates $\boldsymbol{\omega}_{des} = [p_{des}\ q_{des}\ r_{des}]^T$ which bring the quadrotor to the correct orientation very quickly. It focuses only on the error between the axis of the propeller thrusts, which is $\mathbf{e}_z$, and the direction they should be pointing, which is $\mathbf{e}_{z,des}$.

Beginning with a desired acceleration $\mathbf{a}_{des}$ from (2.10), $\mathbf{e}_{z,des}$ is computed in $\mathcal{F}_I$ as

$$\mathbf{e}_{z,des} = \frac{\mathbf{a}_{des}}{||\mathbf{a}_{des}||}. \tag{2.12}$$

The attitude quaternion $\mathbf{q}$ is used to compute $\mathbf{e}_z$ in $\mathcal{F}_I$ as

$$\mathbf{e}_z = \mathbf{q} \odot [0\ 0\ 1]^T. \tag{2.13}$$

Next, we calculate the difference between $\mathbf{e}_z$ and $\mathbf{e}_{z,des}$ as a quaternion error representing a rotation of α about a rotation axis $\mathbf{n}$. The angle α is given by

$$\alpha = \cos^{-1}(\mathbf{e}_z \cdot \mathbf{e}_{z,des}), \tag{2.14}$$

and the rotation axis $\mathbf{n}$ is calculated in $\mathcal{F}_Q$ as

$$\mathbf{n} = \mathbf{q}^{-1} \odot \frac{\mathbf{e}_z^{\times}\mathbf{e}_{z,des}}{||\mathbf{e}_z^{\times}\mathbf{e}_{z,des}||}. \tag{2.15}$$

Finally, the error quaternion $\mathbf{q}_{err}$ which encodes the rotation α about the axis $\mathbf{n}$ can be formed as[3]

$$\mathbf{q}_{err} = \begin{bmatrix} \cos(\alpha/2) \\ \mathbf{n}\sin(\alpha/2) \end{bmatrix} = \begin{bmatrix} q_{err}^{(w)} \\ q_{err}^{(x)} \\ q_{err}^{(y)} \\ q_{err}^{(z)} \end{bmatrix}. \tag{2.16}$$

The elements $q_{err}^{(x,y,z)}$ encode the axis of the rotation to correct the error in orientation about each of the axes in $\mathcal{F}_Q$; the elements $q_{err}^{(x)}$ and $q_{err}^{(y)}$ encode the error along the axes $\mathbf{e}_x$ and $\mathbf{e}_y$ respectively. If the element $q_{err}^{(w)}$ is less than zero, this means that α is greater than π and so the desired direction of rotation must be flipped to choose the shorter path. We control the rotation about $\mathbf{e}_z$ to ensure it is near zero so that the other two rotation axes can be controlled effectively.

The desired body rates p_{des} and q_{des} are thus computed directly from $q_{err}^{(x,y)}$ by scaling them and checking if the rotation should be in the opposite direction. This operation is given as[4]

$$\begin{bmatrix} p_{des} \\ q_{des} \end{bmatrix} = \text{sign}(q_{err}^{(w)})\ k_{err} \begin{bmatrix} q_{err}^{(x)} \\ q_{err}^{(y)} \end{bmatrix}. \tag{2.17}$$

[3] Alternately, it is possible to skip (2.12) to (2.16) and simply form the error quaternion by writing $\mathbf{q}_{err} = \mathbf{q}_{des} \otimes \mathbf{q}^*$ where $\mathbf{q}_{des}$ is the desired attitude extracted from the direction of $\mathbf{a}_{des}$, $\otimes$ denotes quaternion multiplication, and $\mathbf{q}^*$ is the complex conjugate of the current attitude, as is shown in the work of Fresk and Nikolakopoulos [43].

[4] Flipping the sign of these elements can alternately be thought of as swapping q_{err} for its conjugate q_{err}^*.

Since body rates are derivatives of the orientation of the quadrotor, only a proportional gain k_{err} is needed to scale between q_{err} and $\boldsymbol{\omega}_{des}$ in (2.17). The third desired body rate r_{des} is simply set to zero to ensure convergence about $\mathbf{e}_z$.

PID control is then applied to achieve $\boldsymbol{\omega}_{des}$. This generates the control outputs $U_{2..4}$, while the control output U_1 is computed using (2.11). The control outputs are mapped via (2.3) to propeller speeds.

Once the orientation task has been achieved, there are several options for how to complete the thrusting task. It is arguable that the orientation task is the more critical for control because the thrusting task depends upon its success. Assuming the quadrotor is pointing in a direction near to hover orientation, then thrust can be applied appropriately to control the altitude. Logical stages of recovery can be implemented to discretely specify what thrust should be applied, depending on the state of the quadrotor.

CHAPTER 3
Stabilization Using Reorientation Control

Before applying the reorientation control theory to the problem of collision recovery, it was first tested for the case of stabilizing a quadrotor to hover from a wide range of initial conditions. During this phase of the research, the aim was to stabilize the attitude and altitude of a quadrotor thrown into the air in any orientation and with high initial spin. The motivation for testing stabilization to hover was to thoroughly test and validate the reorientation control law first, before complicating the problem by introducing a collision obstacle.

The reorientation control law was coded into the quadrotor dynamics simulator described in [40]. Monte Carlo simulations were performed to ensure that the control law satisfied all of the requirements stated in the previous chapter. Simulation provided a framework for controller validation which would have been more challenging to test initially on a real quadrotor.

A custom propeller protected quadrotor was designed and manufactured for experimental testing. The control code was implemented on the quadrotor's onboard micro-controller with minimal modifications from simulation code, beside syntax. Extensive testing was performed where the quadrotor was thrown into the air, detected that it was undergoing free-fall, and stabilized its attitude and altitude. The throw tests were very successful, performing well under a wide range of challenging initial conditions.

In Section 3.1, the simulation environment and Monte Carlo simulations will be presented. In Section 3.2, the implementation and outdoor testing on a real quadrotor will be detailed. Finally, Section 3.3 summarizes the results of both simulation and implementation.

3.1 Simulation

3.1.1 Quadrotor Dynamics Simulator

All simulations were performed in MATLAB using its standard libraries as well as the MATLAB Aerospace Toolbox. Although an existing quadrotor simulation environment could have been used, quadrotor collision recovery modelling required a customized contact dynamics model in addition to quadrotor flight dynamics. Also, programming a custom quadrotor dynamics simulator allowed for straightforward modifications to a variety of parameters which will be discussed in this chapter. Working in conjunction with Fiona Chui who initially developed the simulator, we programmed the simulation environment using the MATLAB 'struct' data type to concisely log the relevant variables.

The dynamics model incorporates most important dynamic effects while omitting or simplifying others. Notably, the terms due to propeller acceleration as well as the saturation limits of this acceleration were modelled. The thrust and drag generated by the propellers were simply computed as a multiple of the square of the propeller speeds, neglecting all other aerodynamic effects.

The simulator propagates the state using ode45 with variable time step. The simulation parameters are initialized as follows. The time step is set to 200 Hz to match the update rate of the experimental flight controller. MATLAB 'struct' data types named Control, PropState, Pose, and Twist are initialized and are updated at each time step with the output of ode45 and the controller output. The values of Pose and Twist are updated directly from

the state of the quadrotor, which is

$$\mathbf{x} = [u \;\; v \;\; w \;\; p \;\; q \;\; r \;\; X \;\; Y \;\; Z \;\; q^{(w)} \;\; q^{(x)} \;\; q^{(y)} \;\; q^{(z)}]^T. \tag{3.1}$$

A visualization was developed by Fiona Chui, which animates the history of all of the above variables updated into the 'struct' data type Hist at each time step. Snapshots from an example visualization are shown in Fig 3–1.

The key simulation parameters are summarized in Table 3–1. The principle moments of inertia were obtained by weighing each of the individual components of the experimental platform and performing an automated computation in CAD, using the 3D locations of each of the components about the center of mass. The propeller moment of inertia about its rotation axis was obtained from a DJI propeller datasheet. The thrust and drag coefficients were estimated for these propellers experimentally using a force-torque sensor.

Table 3–1: Dynamics and Simulation Parameters

Parameter	Value	Units	Description
g	9.81	m/s^2	Gravitational acceleration
m	1.096	kg	Quadrotor mass
$\mathbf{I}$	diag [0.008 0.008 0.017]	$kg \cdot m^2$	Quadrotor moments of inertia
J_r	2.208e-5	$kg \cdot m^2$	Propeller moment of inertia
l	22	cm	Arm length
k_t	8.7e-8	N/RPM^2	Thrust coefficient
k_d	8.7e-9	Nm/RPM^2	Drag coefficient
dt	0.005	seconds	Simulation time step

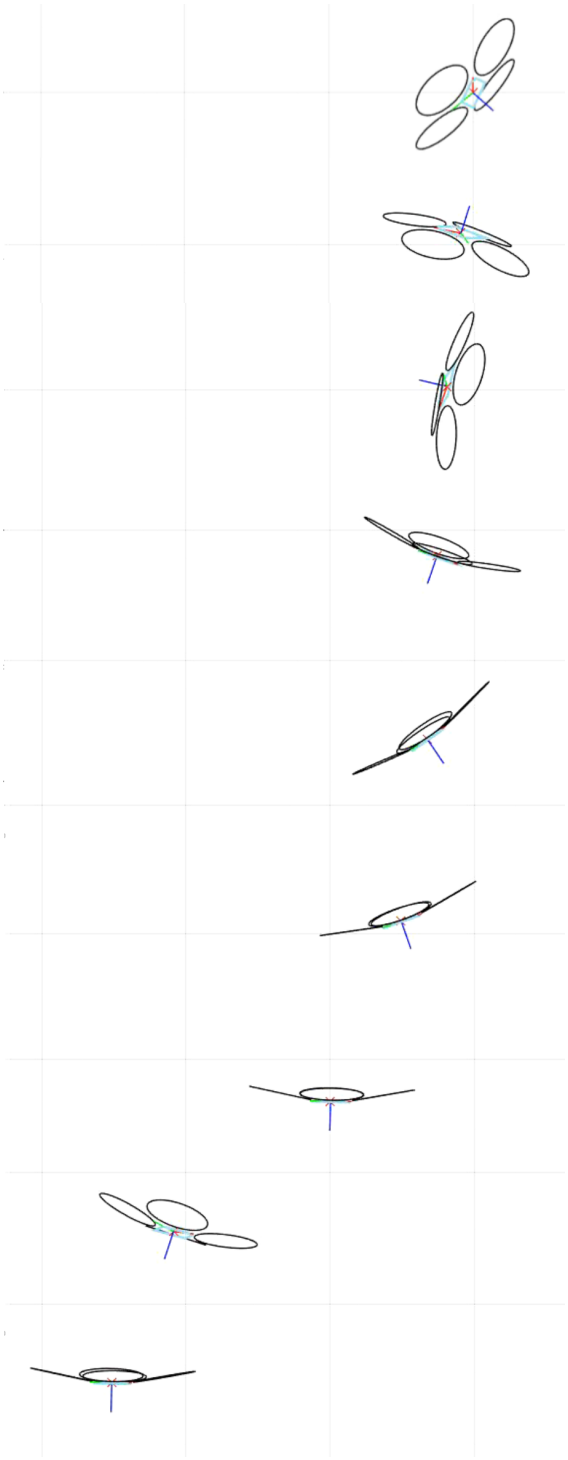


Figure 3–1: Snapshots of visualization from an example simulation trial

3.1.2 Control Structure

Stabilization to hover implies that the task for the quadrotor is to orient itself upright and apply thrust to bring its vertical velocity to zero. In addition, it may be desirable to also stabilize horizontal velocity. Although stabilizing horizontal velocity was deemed superfluous for recovery control, it was nevertheless tested in simulation to show that the control law could in fact achieve full positional stability. The three stages of stabilization in simulation, then, are as follows:

1. Reorient upright and bring angular velocity to zero,
2. stabilize vertical velocity,
3. stabilize horizontal velocity.

It was decided to execute these stages consecutively with logical switch conditions separating each stage from the next. Table 3–2 summarizes these stabilization stages and switch conditions.

Table 3–2: Stabilization Control Switch Conditions

Stage	Switch Condition
RS-1: Reorient, stabilize $\boldsymbol{\omega}$	$abs(\mathbf{q}_{e,rp}^{(x,y)}) < \mathbf{q}_{switch}^{(x,y)} \ \wedge \ \boldsymbol{\omega} < \boldsymbol{\omega}_{switch}$
RS-2: Stabilize $\dot{Z}$	$abs(\dot{Z}) < \dot{Z}_{switch}$
RS-3: Stabilize $\dot{X}$, $\dot{Y}$	$\|\|[\dot{X}\ \dot{Y}]^T\|\| < \dot{X}\dot{Y}_{switch}$

The reorientation controller begins immediately at the first simulation time step, unlike on a real quadrotor where it must be triggered by detecting free-fall. In each stage, $\mathbf{a}_{des}$ is set differently, as shown in Table 3–3. The proportional gains $[k_X\ k_Y\ k_Z]^T$ control the errors

on the velocities on each of the inertial axes. Note that only proportional gains are necessary for quadrotor velocity control.

Table 3–3: Desired Accelerations for Stabilization Stages

Stage	Desired Acceleration
RS-1: Reorient, stabilize $\boldsymbol{\omega}$	$\mathbf{a}_{des} = [0\ 0\ -g]^T$
RS-2: Stabilize $\dot{Z}$	$\mathbf{a}_{des} = [0\ 0\ (-k_Z\dot{Z} - g)]^T$
RS-3: Stabilize $\dot{X}$, $\dot{Y}$	$\mathbf{a}_{des} = [-k_X\dot{X} \quad -k_Y\dot{Y} \quad (-k_Z\dot{Z} - g)]^T$

The thrust is computed as

$$U_1 = \begin{cases} m\ (\mathbf{a}_{des}^T \mathbf{e}_z), \text{ if } (\mathbf{a}_{des}^T \mathbf{e}_z) < 0 \\ 0, \text{ otherwise.} \end{cases} \tag{3.2}$$

The moments $U_{2..4}$ are computed according to the control law described in the previous chapter. Finally, the propeller speeds are calculated and saturated. The propeller speeds, rather than $\mathbf{U}$, are sent to the dynamics model, which in turn converts them back to forces and moments according to their coefficients k_t and k_d for estimating their effective thrust and torque. The reason for sending propeller speeds instead of control outputs is to allow saturation limits to be applied directly on the propeller speeds and propeller acceleration.

Table 3–4 shows values and descriptions of the key parameters used in simulation. These values were all tuned by trial and error by examining the state variable profiles as well as the quadrotor visualization. The switch conditions between stages needed to be high enough to allow passing from one stage to the next without delay, but low enough to require the previous stage to reach a sufficient level of completion. Systematic tuning of these switch

Table 3–4: Stabilization Control Parameters

Parameter	Value	Units	Description
k_{err}	20	rad/s	Gain mapping error to desired $\boldsymbol{\omega}$
$\boldsymbol{\omega}_{switch}$	$[0.5\ 0.5\ 0.5]^T$	rad/s	Body rate switch threshold
$\dot{Z}_{switch}$	0.2	m/s	Vertical velocity switch threshold
$\dot{X}\dot{Y}_{switch}$	0.2	m/s	Horizontal velocity switch threshold
$\mathbf{q}_{switch}^{(x,y)}$	$[0.3\ 0.3]^T$	–	Quaternion elements switch threshold
$[k_X\ k_Y\ k_Z]^T$	$[3.0\ 3.0\ 5.0]^T$	s^{-1}	Proportional gains for velocity control

parameters seemed superfluous at that stage in the research, since the goal was simply to demonstrate that stage switching was a valid method.

3.1.3 Monte Carlo Validation

To evaluate the reorientation controller, Monte Carlo simulations were performed by randomizing the initial conditions of the simulator thousands of times and analyzing the statistical results. The Monte Carlo method ensures that any extreme or unstable edge cases are identified, while providing important information about averages and standard deviations for variables such as height loss, horizontal drift and stabilization time.

The range of randomized initial conditions was chosen to be quite large, especially on the initial body rates. The aim was to perform simulations that would cover the full range of possible real aggressive throws of a quadrotor. Therefore, any initial attitude was allowed. The body rates p and q were allowed to be as large as 4.2 rev/sec, while the yaw rate r was allowed to be as large as 1.6 rev/sec. These ranges were chosen based on logs of gyroscopic data generated while throwing the quadrotor aggressively in the air. The initial linear velocity of the quadrotor was also randomized to be up to 2 m/s in any direction. This was done to generate realistic throw scenarios, although linear velocity has no effect

on the stability of the control. The randomized initial condition ranges used in the Monte Carlo simulations are summarized in Table 3–5.[1]

Table 3–5: Initial Conditions for Monte Carlo Simulation

Range	Units	Description
$-\pi \leq \phi \leq \pi$	rad	Initial roll
$-\pi \leq \theta \leq \pi$	rad	Initial pitch
$\psi = 0$	rad	Initial yaw (zero b/c irrelevant)
$\|\|p\\|\|,\ \|\|q\\|\| \leq 27$	rad/s	Initial x and y body rates
$\|\|r\\|\| \leq 5$	rad/s	Initial z body rate
$\|\|\dot{\mathbf{p}}\\|\| \leq 2$	m/s	Initial linear velocities

Histograms of simulation results for stabilization times, height loss, and horizontal drift are presented in Figure 3–2. Simulations were run 10,000 times for 2 simulated seconds each. The overall result is that 99.5% of the simulations fully stabilized to RS-3 within the 2 second simulation time, which indicates that close to the full range of initial conditions can be handled by the controller. It is expected that the 0.5% that did not manage to stabilize within 2 sec simply required more simulation time. The average time it took to complete all stages was 0.958 seconds with a standard deviation of 0.443 seconds. The slight bump toward the left of the stabilization time histogram exists because if the initial orientation is not far from hover, then minimal height loss and horizontal drift is incurred and so RS-2 and RS-3 are also quicker to complete.

[1] It should be remarked that randomizing initial Euler angles does not correspondingly randomize $SO(3)$. However, this randomization choice was sufficient for the purpose of Monte Carlo simulations.

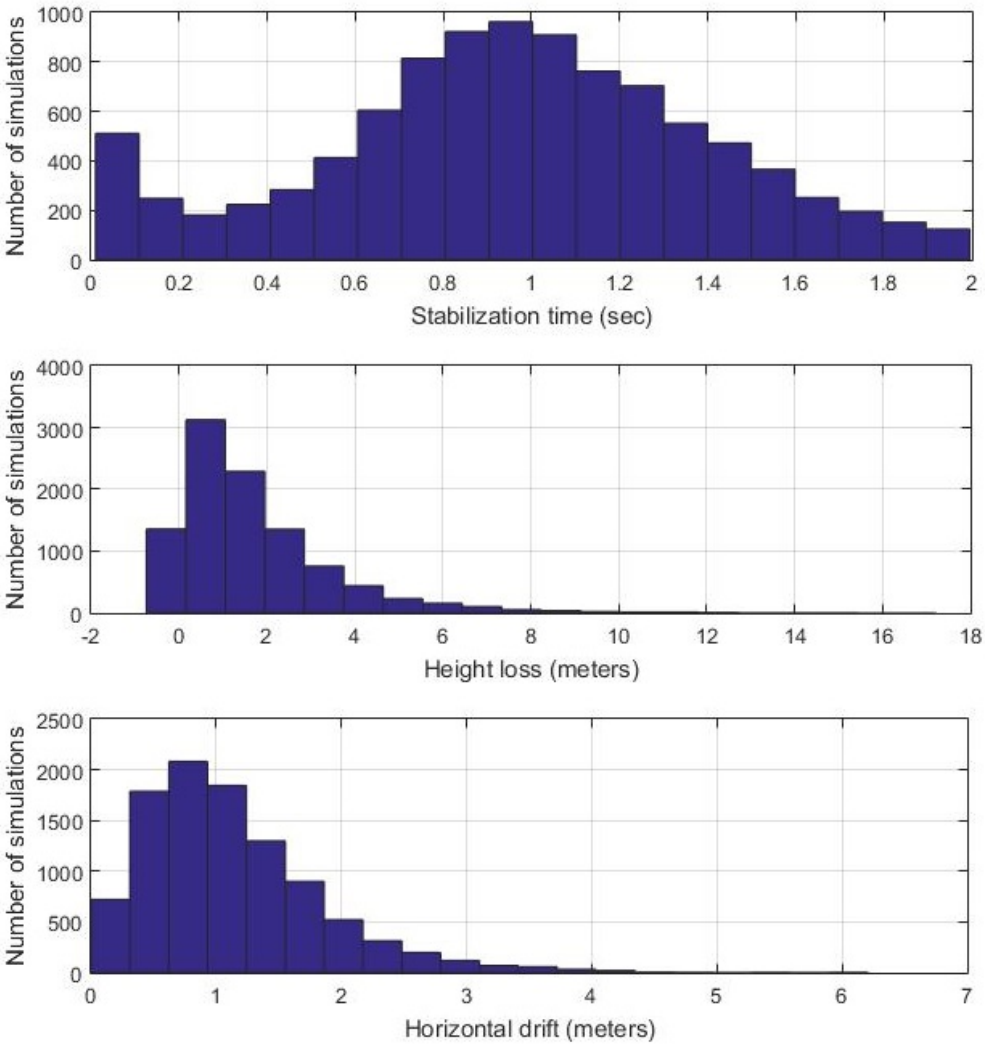


Figure 3–2: Histograms of results from Monte Carlo Simulation

The average height loss was 1.71 m, though for trials with high initial yaw rate the height loss was a large as 18 m. This indicates that there is some threshold for the maximum initial yaw rate that can be stabilized. The average horizontal drift was 1.14 m, although again, a few of the simulations with large initial yaw rate drifted as far as 6m horizontally before stabilization was achieved.

3.2 Implementation

Once the reorientation control law was tested in simulation, it was programmed onto a custom built propeller protected quadrotor. Live tests were performed outdoors by throwing the quadrotor into the air with similar randomized initial conditions to those in simulation. There were two main differences between simulation and the live tests. The first is that in the live throws, reorientation control was triggered by the detection of free-fall. The second difference is that the final stage of stabilizing horizontal velocity autonomously was not attempted because it was superfluous to the research task, would have required additional sensing, and because it has already been demonstrated by [39] using optical flow on an onboard, downward facing camera. Thus, the final horizontal motion stabilization and landing were performed under manual piloting.

Results for the implementation were excellent. The only two crashes that occurred were due to poor manual control after stabilization had occurred, or due to striking the ground because of a delayed detection of free-fall. The reorientation controller, as long as it was engaged, managed to stabilize the quadrotor in every one of over 150 trials.

3.2.1 Experimental Platform

The experimental platform shown in Figure 3–3 was designed, built and programmed by four people: the author, Fiona Chui, Sebastian Schell and Linus Lenhert. Fiona Chui designed the CAD model for 3D printing, Sebastian Schell set up the communication protocols

Figure 3–3: CAD model of 'Navi' experimental platform

and assembled the electronics, and Linus Lenhert provided the CAD model for the first few iterations of propeller protection. The author implemented the flight control code as well as consulted on all stages of the development.

The experimental platform, which we named Navi, was designed on a carbon fiber frame from a previous quadrotor called Spiri from Pleiades Inc. Propeller protection consists of 3D printed nylon alloy bumpers which extend around each of the propellers, and two carbon fiber tubes connecting each bumper to its adjacent bumper. The four 1400 kV motors were outfitted with self-tightening 8" propellers from DJI. Four 20 A Afro Slim electronic speed controllers (ESCs) take PWM outputs from the microcontroller and drive the motor voltage. A four cell 1300 mAh LiPo battery powers hovering flight for about 5 minutes. The total weight of the platform is approximately 1.1 kg, and the arm diameter is 44 cm.

A Pixhawk from 3D Robotics was chosen as the microcontroller, shown in Figure 3–4. The Pixhawk can be used as a controller for a wide variety of aerial, underwater and ground

Figure 3–4: Pixhawk microcontroller and Odroid X4U Linux computer

vehicles. It comes with open-source software which includes a state estimator and a flight controller. The state estimate makes use of an embedded 3-axis accelerometer, gyroscope, barometer, and magnetometer. It fuses barometer and accelerometer measurements to provide an estimate of vertical velocity. A sonar and high-precision infrared sensor were added to the bottom of Navi in the expectation that a better estimate of vertical velocity would be needed for holding altitude, but these sensors eventually proved redundant.

The Pixhawk has ports to connect to other devices such as an onboard computer which can extend the capability of the quadrotor in various ways. We decided to integrate an Odroid X4U onboard computer with the Pixhawk because of the additional functionality that it provides as a Linux operating system. The Odroid provides a wireless entry to log into the Pixhawk's shell during flight, which aids in command line debugging. The Odroid was also used to run ROS with the ROS package 'mavros' which allows for communication to the Pixhawk for autonomous control using external sensing. This functionality was used to test collision recovery in the following chapter.

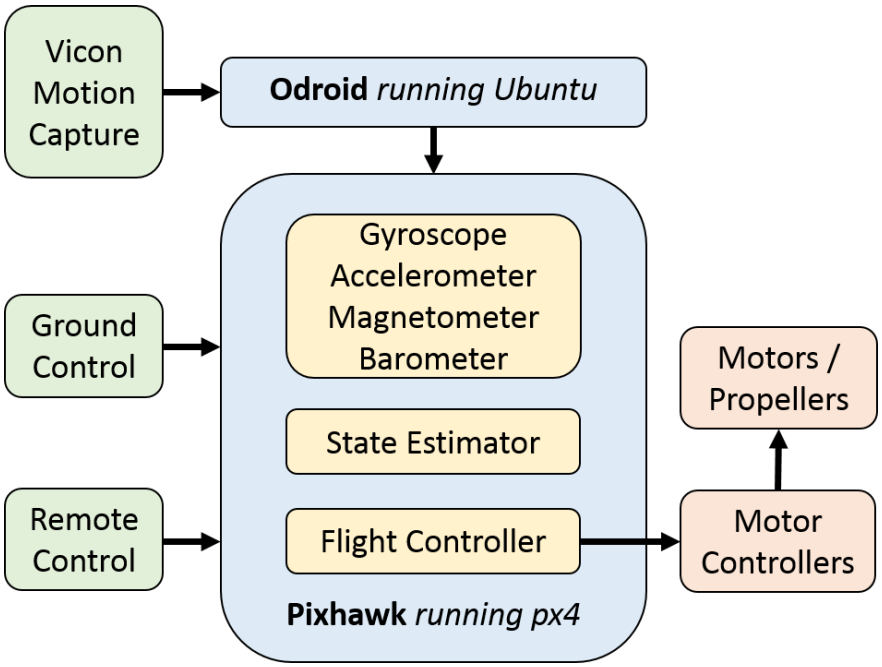


Figure 3–5: Software and hardware components of flight controller

The firmware flight stack which runs the state estimation, flight control, and all middleware functions, is called 'px4'. The px4 flight stack is the most well maintained option for developers. It uses the Mavlink protocol to communicate with other devices. A ground station desktop application called QGroundControl, which is also based on Mavlink, was used for parameter setting, calibration and flight time debugging via telemetry and microUSB cable. The software and hardware connections are depicted in Figure 3–5.

3.2.2 Setup and Procedure

The px4 flight controller module called 'mc_att_control' was modified to run the reorientation control code in place of standard attitude control. It is triggered upon the detection of free-fall. Free-fall was flagged True when the norm of the accelerometer measurement went

below a certain threshold, as

$$\text{Free-fall detection flag} = \begin{cases} 1, \ ||\mathbf{a}_{acc}|| < a_f \ [m/s^2] \\ 0, \ \text{otherwise.} \end{cases} \tag{3.3}$$

Here, $\mathbf{a}_{acc}$ is the accelerometer measurement and a_f is the threshold free-fall acceleration magnitude.

The accelerometer measures the gravity vector when at rest, and during pure free-fall it measures zero. However, once the quadrotor is in an armed state, it spins its motors at their minimum speed, which generates a significant amount of base thrust. Due to this base thrust, the vehicle does not experience pure free-fall and the accelerometer norm will not drop to zero. Instead, a threshold was chosen which is low enough to not trigger free-fall detection accidentally, but high enough to account for the effect of the base thrust.

Initial limited testing was carried out in an indoor space with the vehicle tethered from the ceiling for safety. The controller gain k_{err} was tuned by trial and error to produce the highest possible body rates that could be tracked by the body rate controller without causing instability.

Extensive outdoor tests were conducted in Sudbury Ontario during the NCFRN yearly field trials in June, 2016. Experiments were performed on an empty soccer field on the Laurentian University campus. A meshed net was erected above ground level to catch the quadrotor should the controller fail. In each experiment, the quadrotor was manually thrown into the air. When free-fall was detected, the quadrotor attempted to recover its attitude and altitude under the closed-loop reorientation control and was then manually landed. The main steps of the experimental procedure are:

1. Pilot turns on Navi, turns off safety, and arms motors.

2. Pilot switches to the px4 altitude control mode and sets mid-throttle.
3. Thrower executes pre-designated throw.
4. Navi detects free-fall and enters reorientation control mode.
5. Navi detects reorientation has occurred and begins to stabilize altitude.
6. Navi detects altitude is stable and returns manual control to the pilot.
7. Pilot lands Navi manually.

Figure 3–6: Navi flown at the NCFRN field trials in Sudbury Ontario

3.2.3 Experimental Results

The first set of experiments was performed to test free-fall detection and the overall functioning of the reorientation controller in an outdoor setting. In the course of these experiments the free-fall detection threshold a_f was tuned to about 4.5 m/s^2, as lower values

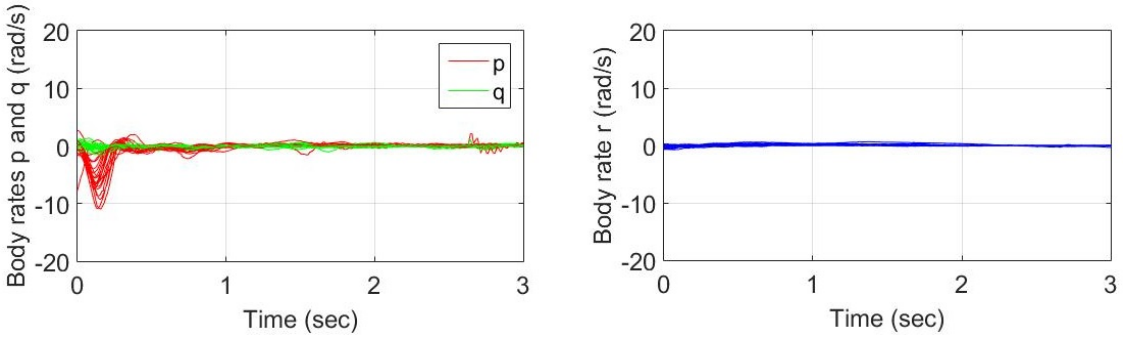
resulted in the reorientation controller not engaging. This value could have been significantly lower had the minimum RPM of the propellers been lower as well.

To thoroughly evaluate the performance of the reorientation control, several formal sets of experiments were performed. In each of the sets of throws, a range of initial conditions was specified for the thrower who attempted to randomize their throws within that range. The initial conditions for the first four sets are summarized in Table 3–6. For these sets, the aim was to gauge the relative significance of two factors: the affect of the quadrotor being initially upside down, and the initial yaw rate, on the reorientation control performance. Twenty randomized throws were performed for each set, totalling 80 trials for this first group of experiments.

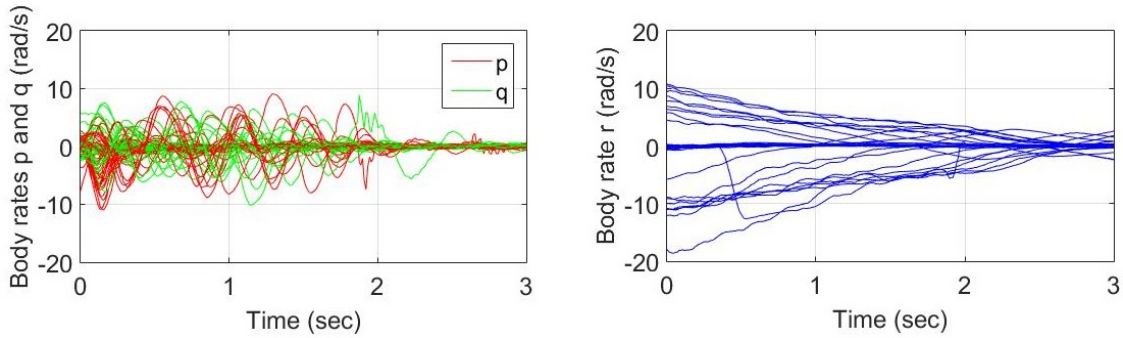
Table 3–6: Initial Conditions for First Group of Trials

Set	Orientation [rad]	Yaw Rate [rad/s]	Description
1	$0 < [\phi\ \theta]^T < \pi$	0	Upright, no yaw rate
2	$0 < [\phi\ \theta]^T < \pi$	$-15 < r < 15$	Upright, with yaw rate
3	$-\pi < [\phi\ \theta]^T < 0$	0	Upside down, no yaw rate
4	$-\pi < [\phi\ \theta]^T < 0$	$-10 < r < 10$	Upside down, with yaw rate

Figure 3–7 shows the convergence of body rates for Set 1 and Set 2, where the quadrotor was initially upright. Stabilization with *no* yaw rate, as shown in Figure 3–7a, was always achieved in 0.4 sec. However, with some initial yaw rate, significant oscillations were observed in body rates p and q. As soon as body rate r became small enough, p and q also converged to zero. As shown in Figure 3–7b, r converges linearly at a maximum rate of about 6 rad/s^2, and oscillations in p and q generally stabilize by about 2 sec, when r has become less than 5 rad/s.

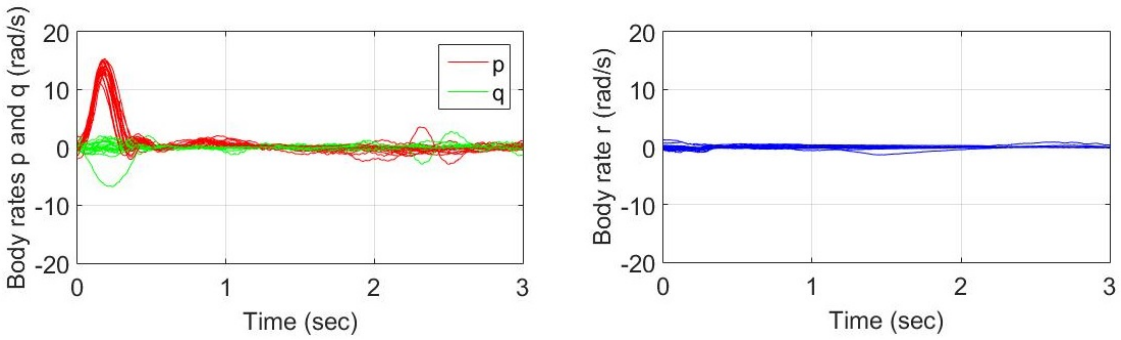


(a) Set 1: Upright, no yaw rate

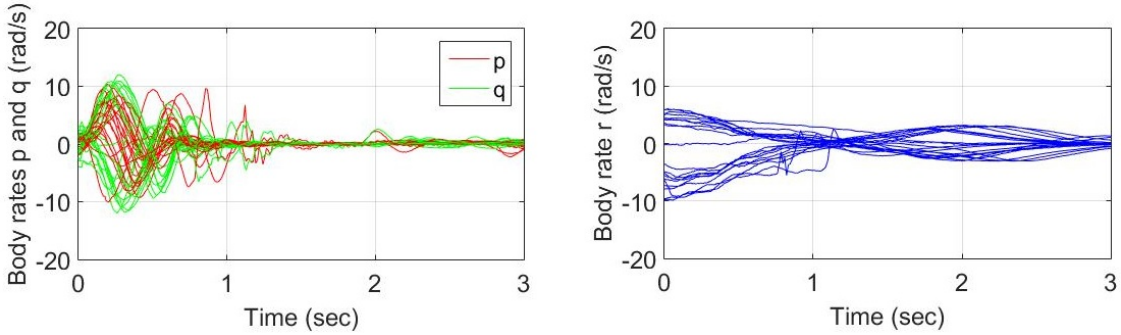


(b) Set 2: Upside down, with yaw rate

Figure 3–7: Body rates for sets 1 and 2



(a) Set 3: Upright, no yaw rate



(b) Set 4: Upside down, with yaw rate

Figure 3–8: Body rates for sets 3 and 4

Figure 3–8a shows that for Set 3, p reached as high as 15 rad/s, whereas in Set 1, p reached a maximum of 10 rad/s. Settling time for Set 3 was about 0.5 sec as compared to 0.4 sec in Set 1. It was not ergonomically possible for the human thrower to give as much initial yaw rate when the quadrotor was initially upside down. In Figure 3–9b, therefore, we see p and q converging in about 1 s for Set 4 as opposed to 2 sec, which was the case in Set 2.

A second group of four sets - 4 sets of 10 throws each - was executed to test the influence of various other factors on the control performance. In Set 5, the quadrotor was released with no initial body rates but in the 't' configuration, meaning that it had to perform a rotation about both the $\mathbf{e}_x$ and $\mathbf{e}_y$ axes. In Set 6, the quadrotor was thrown 'casually' and

'naturally' into the air as if launching it. In Set 7, the quadrotor was given high initial p and q. Finally, in Set 8, the quadrotor was 'aggressively' flung into the air with high body rates on all three axes and large initial velocity. For each of these four sets the orientation was not restricted. The initial conditions for these sets were not precise, but their descriptions are summarized in Table 3–7.

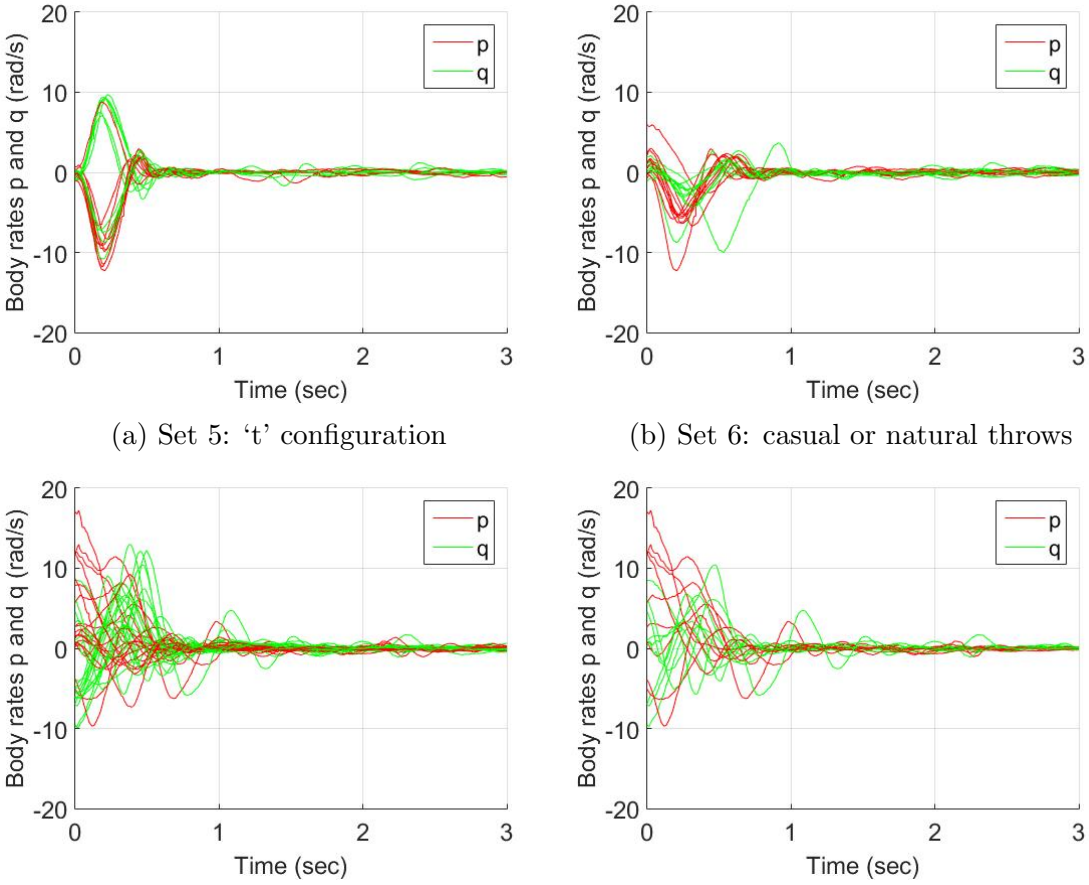
Table 3–7: Descriptions of Second Group of Trials

Set	Description
5	't' configuration
6	Casual or natural throws
7	High initial p and q
8	Aggressive throws

In Figure 3–9a, the release in the 't' configuration demonstrates that the quadrotor was able to stabilize even though only two of its four propellers were generating the moment that controlled the body rates. Its time to stabilization was marginally slower than in the 'x' configuration, as expected.

In Figure 3–9b, it is clear that casual or natural throws present little difficulty to the controller, especially because initial r was not varied for these trials. This indicates that stabilization control would serve well as a launching method for a commercial quadrotor.

Figure 3–9c shows that high initial p and q do not significantly increase the settling time from that of a static body rate throw. Stabilization always occurred within 1 s. This indicates that the control of these body rates is very aggressive, much more so than the control of r, which depends on the drag coefficient rather than the thrust differentials.



(a) Set 5: 't' configuration

(b) Set 6: casual or natural throws

(c) Set 7: high initial p and q

(d) Set 8: aggressive throws

Figure 3–9: Body rates p and q for 2nd group of trials

Finally, Figure 3–9d shows very similar results to those in Set 7, which simply implies that the most aggressive throw a human could generate for a 1 kg quadrotor saturates at about this range of initial conditions. It was ergonomically challenging to produce both large p and q, and large r. An example of one of the most aggressive throws is shown in Figure 3–10.

Figure 3–10: Example of an aggressive throw of the quadrotor

3.3 Summary

The Monte Carlo simulations show that reorientation control manages to stabilize, in order, the quadrotor's attitude, altitude, linear velocity and position, from a wide range of initial conditions. Simulations show that higher initial yaw rate increases stabilization time significantly, whereas large initial roll and pitch rates do not.

The real-time aggressiveness and robustness of the controller is evident from numerous experimental trials. The quadrotor managed to stabilize from being initially upside down and spinning, as well as thrown with high initial body rates on all axes. To the best of the author's knowledge, this is the most thoroughly tested implementation of a global attitude stabilization controller for a quadrotor of mass greater than one kilogram.

CHAPTER 4
Simulation of Collision Recovery Using Reorientation Control

Having implemented quadrotor stabilization control as described in the previous chapter, the reorientation controller was applied to the central research objective of recovering from a collision with a vertical wall. A Monte Carlo simulation was performed to validate the control strategy for a wide range of initial conditions, and the experimental implementation was performed for a narrower range of initial conditions using the same custom quadrotor platform as in the previous chapter.

A strategy, or pipeline, was developed to recover from a collision with a wall and is presented in Section 4.1. The strategy consists of three phases, the first two of which are treated in [16]. This thesis will mention the first two phases in brief, and cover in detail the third phase, which is the execution of the recovery maneuver.

The contact dynamics model between the quadrotor's propeller protection and a wall as described in [16] in the existing MATLAB simulator. This simulator was employed to conduct a Monte Carlo simulation to identify the most challenging collision scenarios for the recovery control strategy. The setup for this simulation is described in Section 4.2. As detailed in Section 4.3, there were some collisions that could not be recovered from, but the large majority of collisions were recoverable in simulation.

4.1 Collision Recovery Pipeline

The collision recovery pipeline consists of three consecutive phases, as shown in Figure 4–1. The first phase detects that a collision has occurred. The intensity of the collision is

characterized using a fuzzy logic process (FLP). Finally, the output of the collision characterization is sent as input to the recovery controller, which executes the recovery maneuver.

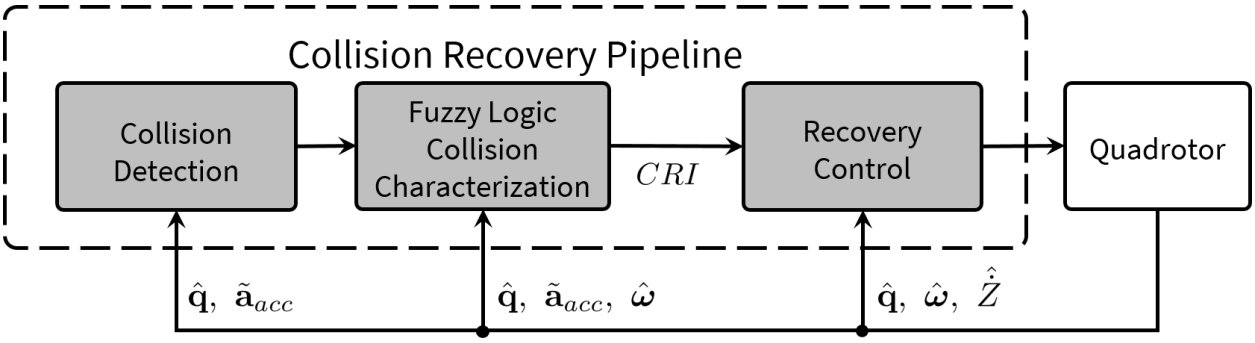


Figure 4–1: Collision recovery pipeline

4.1.1 Collision Detection

Collisions are assumed to be only with vertical walls, which can be detected as a spike in horizontal acceleration. In simulation, this is implemented directly by checking the known state of the center of mass acceleration $\mathbf{a}$. On a real quadrotor, the collision detection phase must estimate this spike from accelerometer measurements. To estimate the horizontal acceleration, the accelerometer measurement $\tilde{\mathbf{a}}_{acc}$ is rotated into the inertial frame and compensated with gravity $\mathbf{g}$ as

$$\hat{\mathbf{a}} = \hat{\mathbf{q}} \odot \tilde{\mathbf{a}}_{acc} + \mathbf{g}. \tag{4.1}$$

Here, hats denote estimated values and the tilde denotes a measured value. A collision is detected when the magnitude of the inertial acceleration horizontal components reaches a threshold given by

$$\text{Collision detection flag} = \begin{cases} 1, & \left|\left|\left[\hat{a}_X,\ \hat{a}_Y\right]^T\right|\right| > k_{det}\ [g] \\ 0, & \text{otherwise} \end{cases}. \tag{4.2}$$

At the time of collision detection, the wall normal $\hat{\mathbf{e}}_N$ is also estimated for use in the remaining two pipeline phases as

$$\hat{\mathbf{e}}_N = [\hat{a}_X,\ \hat{a}_Y]^T \,/\, \left\|[\hat{a}_X,\ \hat{a}_Y]^T\right\|. \tag{4.3}$$

In simulation, the wall normal is known perfectly, whereas in experimentation this normal direction must be estimated from measured acceleration components. Using (4.3), we found that the wall normal estimated in experiments had an error of as much as 15° away from the true direction because the measured components of acceleration are affected by the quadrotor's sudden increase of rotation at the moment of impact.

4.1.2 Collision Characterization

Once a collision is detected, the characterization phase begins. A collision characterization method using fuzzy logic as presented in detail in [16], where the intensity of the collision is characterized by a single FLP output: Collision Response Intensity (CRI), ranging from -1 to 1. As shown in Figure 4–2, the fuzzy logic outputs are named 'AB' for 'Away, Big', 'AS' for 'Away, Small', 'L' for 'Level', 'TS' for 'Toward, Small' and 'TB' for 'Toward, Big'. They indicate that the quadrotor is either undergoing a big or small collision and is inclined either toward or away from the wall at the time of collision. This allows for an intelligent simplification of the complex types of collisions that a quadrotor can undergo with a wall. By mapping them to the CRI value, we obtain a useful indicator of how aggressive the recovery control should be.

4.1.3 Recovery Control Stages

The CRI as well as $\hat{\mathbf{e}}_N$ are sent to the final recovery pipeline phase where they are mapped to a reference acceleration of the center of mass of the quadrotor, $\mathbf{a}_{ref}$. The recovery control passes through three stages, the first of which tracks this reference acceleration. In

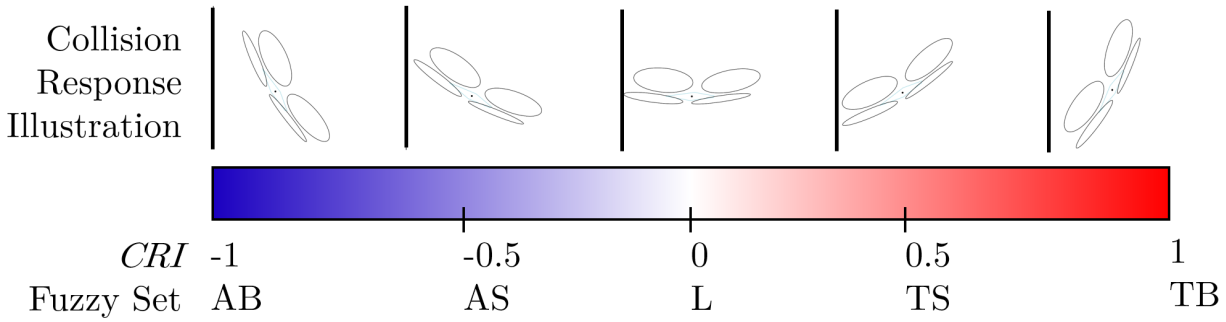


Figure 4–2: Collision Response Intensity scale

the second and third stages, the quadrotor attempts to achieve an upright attitude and zero vertical velocity, respectively. Once this has occurred, recovery is complete. Note that horizontal velocity control is not included in the recovery control phase because it is not critical to the recovery task and because it would follow the aforementioned three stages which complete the actual recovery from the collision. Furthermore, horizontal velocity control and stabilization has been demonstrated previously with onboard sensing in [39].

Figure 4–3 shows the logic of the recovery control phase, and the corresponding desired acceleration and switch conditions for each stage are summarized in Table 4–1.

Table 4–1: Recovery Control Switch Conditions

Stage	$\mathbf{a}_{des}$	Switch Condition
RS-1	$\mathbf{a}_{ref} - \mathbf{g}$	$abs(\mathbf{q}_{e,rp}^{(x,y)}) < \mathbf{q}_{switch}^{(x,y)} \ \wedge \ \boldsymbol{\omega} < \boldsymbol{\omega}_{switch}$
RS-2	$-\mathbf{g}$	$abs(\mathbf{q}_{e,rp}^{(x,y)}) < \mathbf{q}_{switch}^{(x,y)} \ \wedge \ \boldsymbol{\omega} < \boldsymbol{\omega}_{switch}$
RS-3	$\mathbf{a}_{alt} - \mathbf{g}$	$abs(\dot{Z}) < \dot{Z}_{switch}$

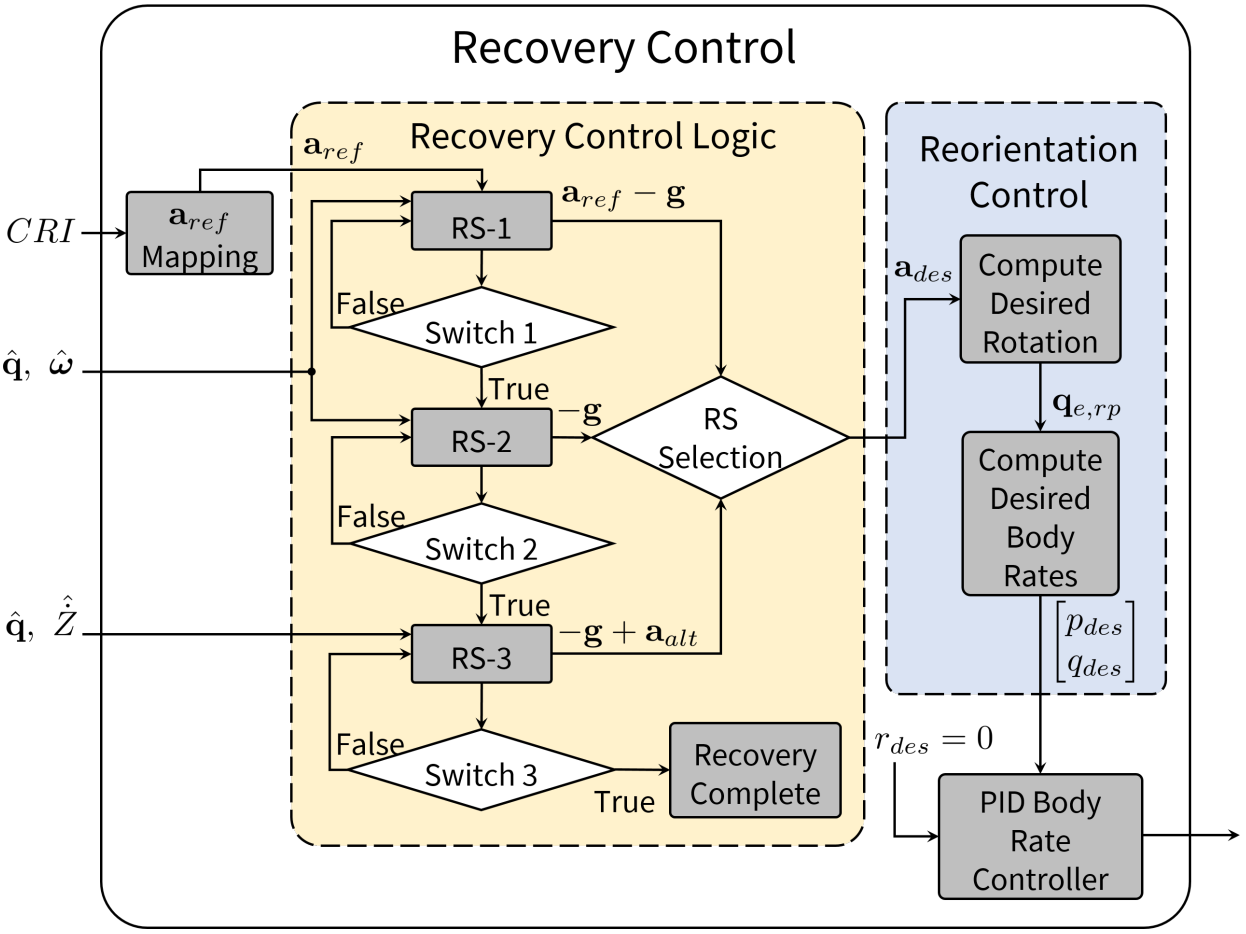


Figure 4–3: Recovery stage logic and control flow

For Recovery Stage 1 (RS-1), the magnitude of $\mathbf{a}_{ref}$ is mapped from the *CRI*. By trial and error, it was determined that scaling from *CRI* to $\mathbf{a}_{ref}$ should be treated differently if the *CRI* was positive or negative. In simulation and experimentation, the scaling rule was set as

$$||\mathbf{a}_{ref}|| = \begin{cases} k_a\ CRI,\ CRI > 0 \\ 1/2\ k_a\ CRI,\ \text{otherwise} \end{cases} . \tag{4.4}$$

It should be noted that this logic is suggested simply as a first attempt at generating appropriate $\mathbf{a}_{ref}$ values.

With the magnitude of $\mathbf{a}_{ref}$ obtained, we orient $\mathbf{a}_{ref}$ along the wall normal as

$$\mathbf{a}_{ref} = \|\mathbf{a}_{ref}\| \ \hat{\mathbf{e}}_N. \tag{4.5}$$

Finally, $\mathbf{a}_{des}$ is computed as

$$\mathbf{a}_{des} = \mathbf{a}_{ref} - \mathbf{g}. \tag{4.6}$$

The controller attempts to track this $\mathbf{a}_{des}$ until the Switch condition at RS-1 is met (see Table 4–1). In RS-2, the quadrotor attempts to achieve an orientation of hover by setting $\mathbf{a}_{ref}$ to zero. Finally, the third stage (RS-3) stabilizes the quadrotor's altitude, so $\mathbf{a}_{des}$ is offset by the output $\mathbf{a}_{alt}$ of the altitude controller, expressed as an acceleration for simplicity. Once Switch 3 evaluates True, the recovery pipeline terminates. In experimentation, RS-3 was not included as a recovery stage for reasons discussed in Section 4.4.

4.2 Monte Carlo Simulation

The collision recovery pipeline was validated by running 10,000 Monte Carlo simulations through a range of incoming collision conditions.

4.2.1 Contact Dynamics Simulator

The wall in simulation is defined as a vertical plane which can come in contact with the quadrotor's propeller protection. The propeller protection is modelled as four circular bumpers concentric with the quadrotor's propellers. When any of these bumpers overlaps geometrically with the wall plane, the magnitude of overlap is used to compute a deflection which in turn generates a normal force on the bumper. A frictional force is also generated proportional to the normal force,[1] scaled by the variable friction coefficient μ, up to its

[1] The static and kinetic values of coefficient of friction are assumed to be the same.

kinetic value, which was obtained by a wall friction experiment. As few as zero or as many as four bumpers can be simultaneously in contact with the wall. All contact forces are lumped into $\mathbf{F}_C$ which is included in the quadrotor's dynamics equations as defined in (2.6). More detail on this aspect of the simulator can be found in [40]. It should be noted that the wall effect, which is the change in thrust and drag generated by propellers in proximity of a wall, was not included in the simulated dynamics.

4.2.2 Initial Conditions

The range of incoming conditions was wider than those that would be possible to generate in experiments. The incoming speed and angle into the wall were identified as having the most effect on the post-collision motion, so these were most widely varied. The incoming yaw angle also has a significant effect on the response because it changes whether one or two bumpers strike the wall. The angle of the quadrotor about the wall normal was also slightly varied. Incoming angular and body rates were not randomized because a quadrotor flying into a wall generally does not have large body rates.

The direction of velocity into the wall was along the world frame $\mathbf{e}_X$ axis. Since the quadrotor is symmetric in its roll and pitch axes, it was decided to arbitrarily orient the quadrotor pre-collision such that it *pitched* into or away from the wall, with a roll angle tilted about the wall normal. However, for non-zero roll and yaw angles, the pitch angle is no longer a measure of inclination into or away from the wall. We therefore define the true inclination angle ζ as the angle between the projection of the body-fixed $-z$ axis onto the vertical plane normal to the wall, and the inertial Z axis. If we define the wall tangent direction $\hat{\mathbf{e}}_T = \mathbf{e}_Z{}^{\times}\hat{\mathbf{e}}_N$ in $\mathcal{F}_I$, and the body-fixed $-z$ axis rotated into $\mathcal{F}_I$ as $\mathbf{e}_z^I = \mathbf{q} \odot (-\mathbf{e}_z)$,

Table 4–2: Parameters and Ranges for Recovery Control Simulation

Value/Range	Units	Description
$\mu = 0.3$	–	Wall/bumper kinetic friction coefficient
$k_a = 9.81$	m/s^2	Scaling factor between CRI and $\mathbf{a}_{ref}$
$k_{det} = 1.0$	g	Acceleration threshold for collision detection
$-15 < \phi < 15$	$^\circ$	Initial roll
$-45 < \theta < 45$	$^\circ$	Initial pitch
$-45 < \psi < 45$	$^\circ$	Initial yaw
$\boldsymbol{\omega} = \mathbf{0}$	$^\circ$/s	Initial body rates
$0.5 < \dot{X} < 2.5$	m/s	Incoming velocity along $\mathbf{e}_X$ axis

then ζ is computed as

$$\zeta = \text{sign}(\zeta) \cdot \cos^{-1}\left(\frac{\left(\mathbf{e}_z^I - \left((\mathbf{e}_z^I)^T\hat{\mathbf{e}}_T\right)\hat{\mathbf{e}}_T\right)^T\mathbf{e}_Z}{\left\|\mathbf{e}_z^I - \left(\left(\mathbf{e}_z^I\right)^T\hat{\mathbf{e}}_T\right)\hat{\mathbf{e}}_T\right\|}\right), \tag{4.7}$$

which is positive when the quadrotor is inclined into the wall. The inclination angle was used for collision characterization and also in the analysis of simulated and experimental results.

The Monte Carlo simulation parameters and initial conditions are summarized in Table 4–2, and the parameter values for switch conditions shown in Table 4–1 are the same as for stabilization control stated in Chapter 3 in Table 3–4.

It should be repeated that randomizing initial Euler angles does not correspondingly randomize inclination angles. The subset of of the attitudes in $SO(3)$ that were allowed by the specified ranges of Euler angles generated more inclination angles near to zero than near to 45°.

At the start of simulation, all four motors speeds were initially set to their hover thrust and the quadrotor was placed directly next to the wall. The collision occurred within a

few milliseconds of the start of simulation. The exact time of collision depended on the initial attitude and incoming velocity. During this short pre-collision period, the static hover thrusts of the propellers, pointing in a random direction, did have an effect on the velocity into the wall at the moment of collision. However, this had little effect on the randomization of incoming velocity because the direction of thrust was also random. It was decided that these drawbacks on the randomization of initial conditions were not significant enough to warrant investigating the issue further.

4.3 Results and Analysis

4.3.1 Recovery Failure Rate Correlations

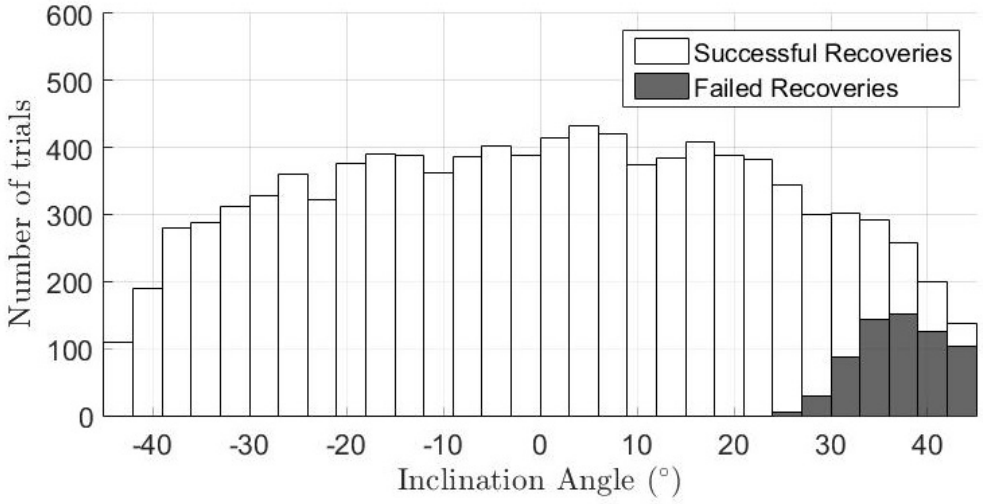


Figure 4–4: Dependency of failure on inclination angle

The Monte Carlo simulations showed successful recoveries for 91.87% of the trials, each of which ran for three simulated seconds. It was found that 93.47% completed RS-1, and all trials which completed RS-2 also completed RS-3. The 1.74% of outlier trials which

completed RS-1 but did not complete RS-3 simply ran out of the allocated three-second simulation time frame. When these outlier trials were individually re-run with longer simulation time frames, they all completed to successful recovery at the end of RS-3.

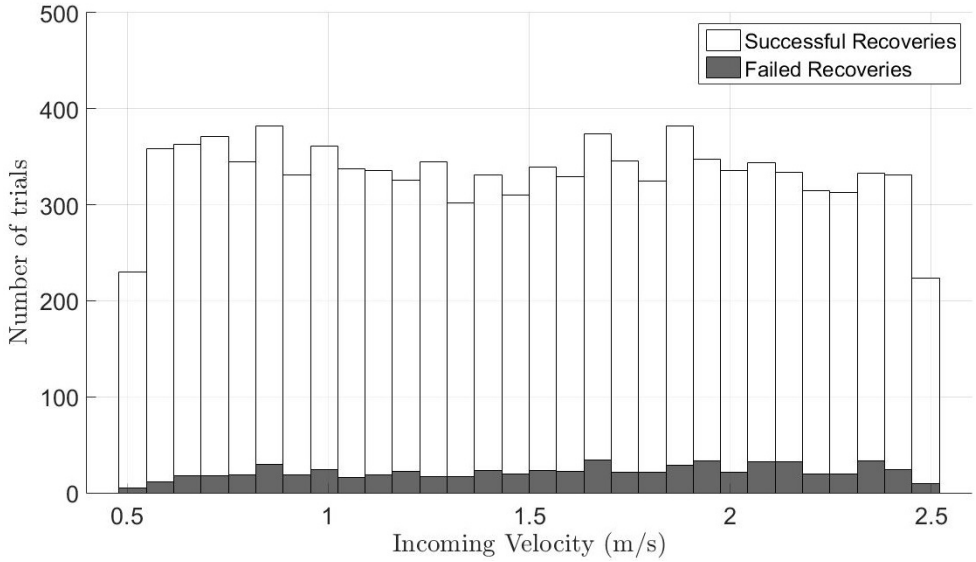


Figure 4–5: Simulated successes and failures vs. incoming velocity

The most important question for the Monte Carlo simulation is which incoming conditions are the most challenging to recover from? The two incoming conditions that were expected to have the largest effect on failure rate were inclination angle and incoming velocity. It was found that failure rate depends heavily on inclination angle, as seen in Figure 4–4, where failures begin to occur when ζ is higher than about 25°. On the other hand, no strong correlation could be found between incoming velocity and failure rate, as seen in

Figure 4–5.[2] Neither were there correlations between failure rate and varying initial roll or yaw angles.

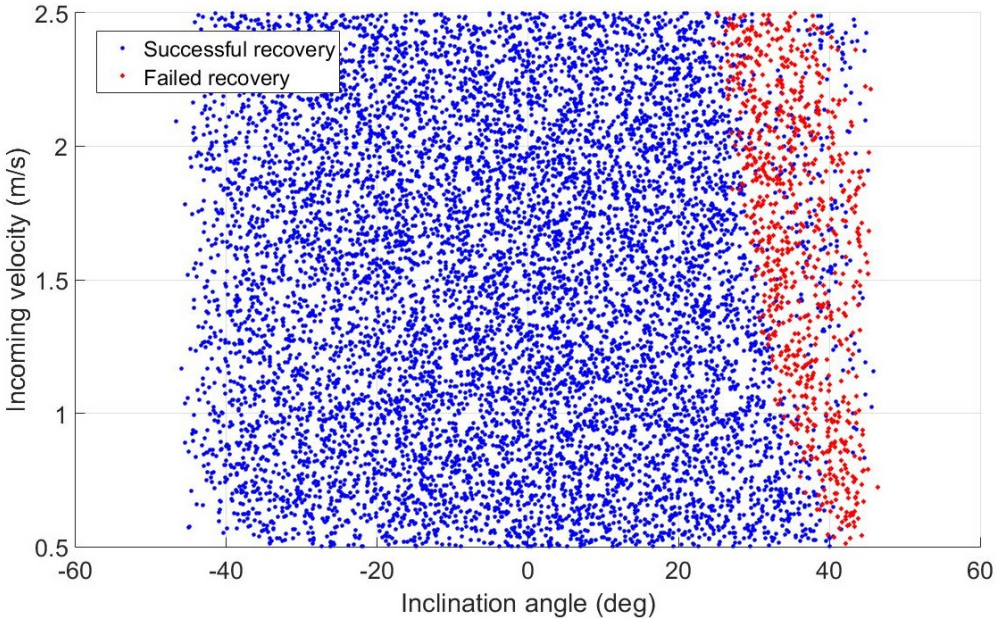


Figure 4–6: Simulated failures vs. inclination and velocity

Interestingly, although there is no correlation between incoming velocity and failure rate, the variance of inclination angles that cause failures increases with incoming velocity. At low incoming velocities, only inclination angles higher than 40° cause failures, whereas at incoming velocities around 2.5 m/s, failures occur at inclination angles as low as 25°. As Figure 4–6 shows, there are a number of successful recoveries at higher incoming velocities and inclination angles, whereas this number decreases with decreased velocity. At higher

[2] The fewer number of trials in Figure 4–5 at the lower and upper bounds of velocity is due to the effect mentioned at the end of Section 4.3

incoming velocities there is a larger rebound from the wall. This rebound gives the recovery controller more time to reorient the quadrotor away from the wall.

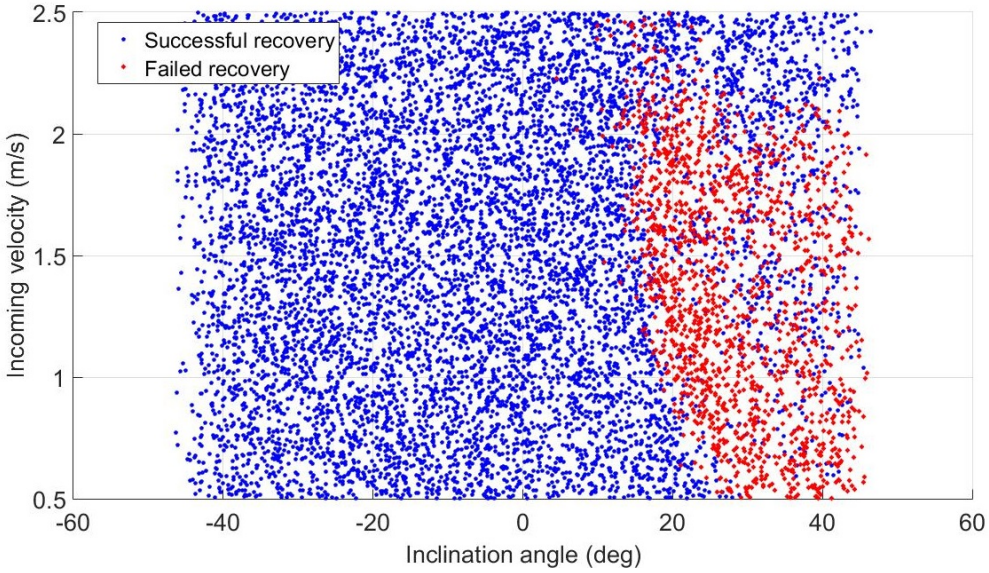
4.3.2 The Effect of Friction

The above Monte Carlo simulation results were obtained with the kinetic friction coefficient $\mu = 0.3$ which is a reasonable value for friction between the bumper material and a plastered, painted wall. The Monte Carlo simulations were re-run with $\mu = 0.0$ and $\mu = 0.6$ to test the effect of friction on the failure rate. It was found that increasing μ decreases the failure rate significantly. Figure 4–7 compares the failure rates vs. inclination angles and incoming velocities for the same range of initial conditions as the Monte Carlo simulations shown in Figure 4–6.

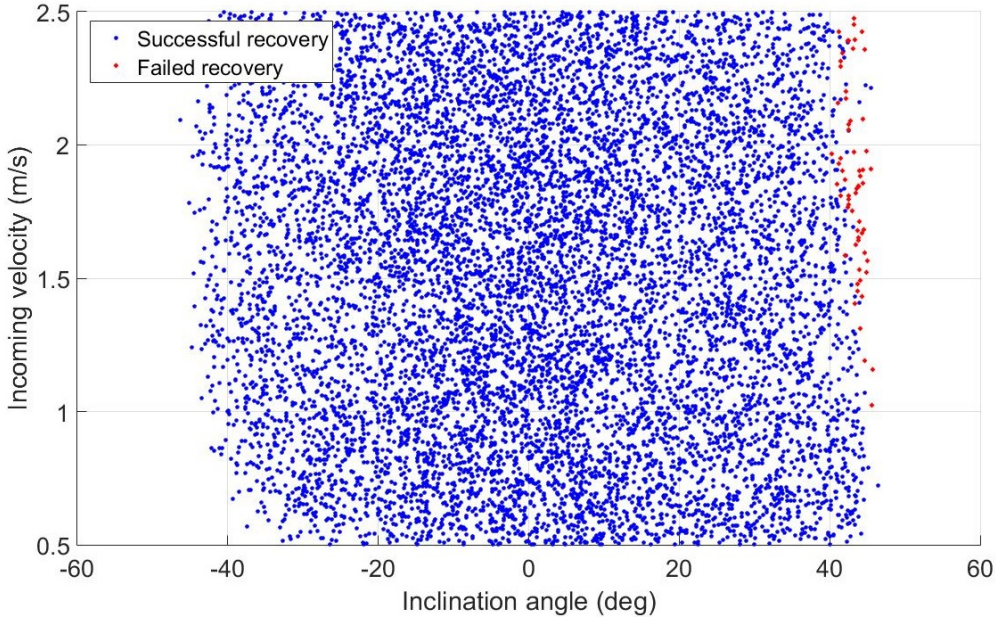
In Figure 4–7a, with no friction between the quadrotor and the wall, the failure rate increased to 16% from 8% with $\mu = 0.3$. With $\mu = 0.0$, failures occur at lower inclination angles, but also more frequently at lower incoming velocities. In Figure 4–7b, with high friction ($\mu = 0.6$) between the quadrotor and the wall, the failure rate decreased to less than 1% for the same range of initial conditions. In short, the higher the friction between the quadrotor and the wall, the easier it is to recover from a collision.

4.3.3 Height Loss and Horizontal Drift

Height loss and horizontal drift were also investigated. The average height loss from the moment of collision to the completion of recovery was -0.82 m, as seen in Figure 4–8. The vast majority of collision trials dropped no more than 4 m in height, although this is a significant amount. The largest height loss naturally occurred in trials which only managed to recover after the quadrotor slid down the wall for a significant period of time. Positional changes for trials which did not succeed at recovery are not included in the histograms.



(a) $\mu = 0.0$



(b) $\mu = 0.6$

Figure 4–7: Effect of μ on failure rate vs. inclination and velocity

Change in horizontal position, shown in Figure 4–9, is largely a function of the *CRI value, since this was used to generate* $\mathbf{a}_{des}$ *which in turn determined the orientation the*

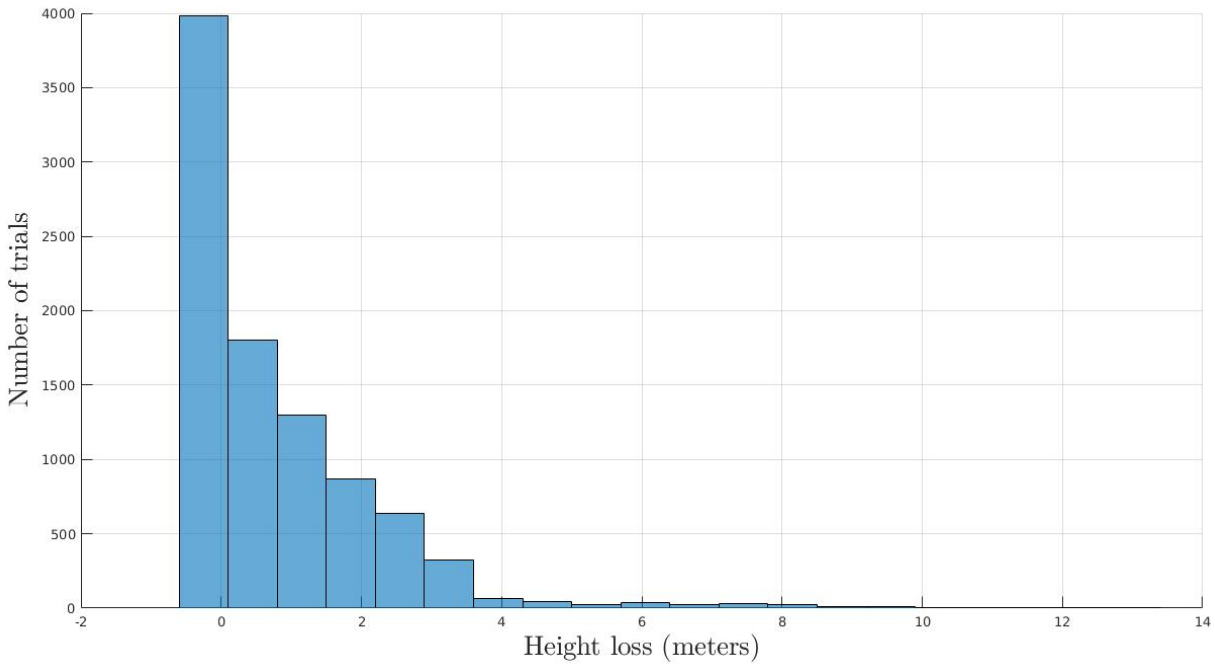


Figure 4–8: Height loss for simulations which recovered

quadrotor should attempt to achieve in RS-1. If the desired orientation was further from hover, which was the case for more intense collisions, then the quadrotor would displace more in the horizontal direction. The majority of trials displaced less than 0.5 m horizontally, with a plateau of trials between 0.5 and 3.0 m.

Horizontal drift could be decreased by re-thinking various aspects of the collision recovery pipeline. A first step at improvement could be to tune the gain k_a which scales CRI *to $\mathbf{a}_{ref}$, as shown in (4.4), or to change the scaling logic of (4.4). Another option would be to change the logical switch conditions between recovery control stages, or even to merge the stages of recovery into a single stage. For instance, Switch 1 waits until the quadrotor is actually tracking $\mathbf{a}_{ref}$ to evaluate True, but this could be modified to switch as soon as the quadrotor is pointing at all away from the wall, by checking $\zeta < 0°$.*

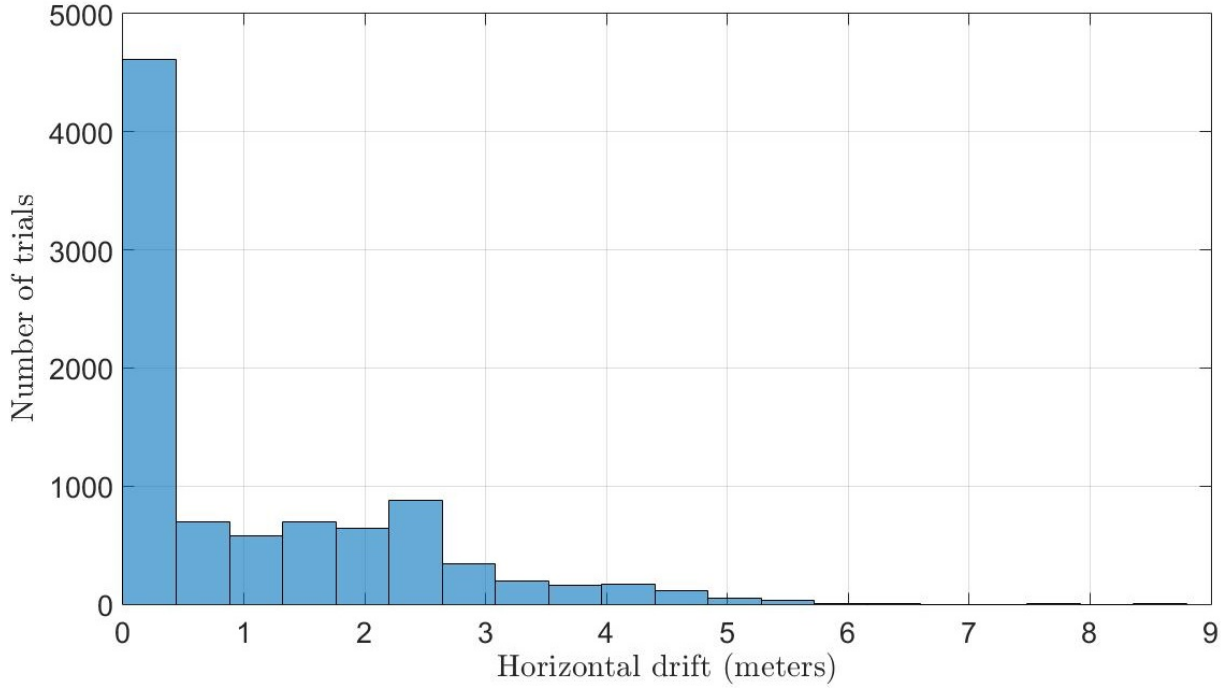
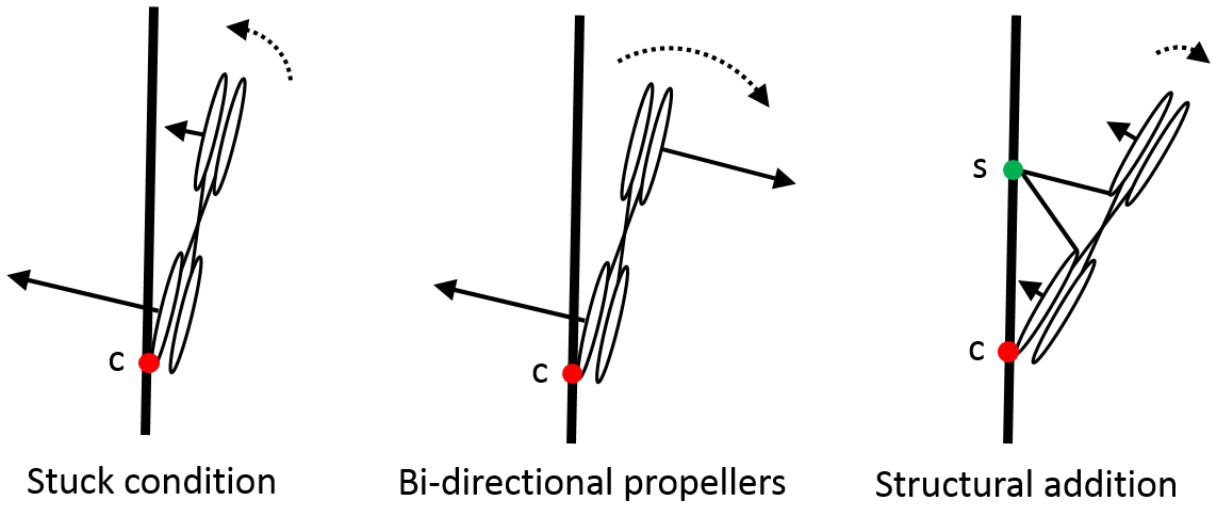


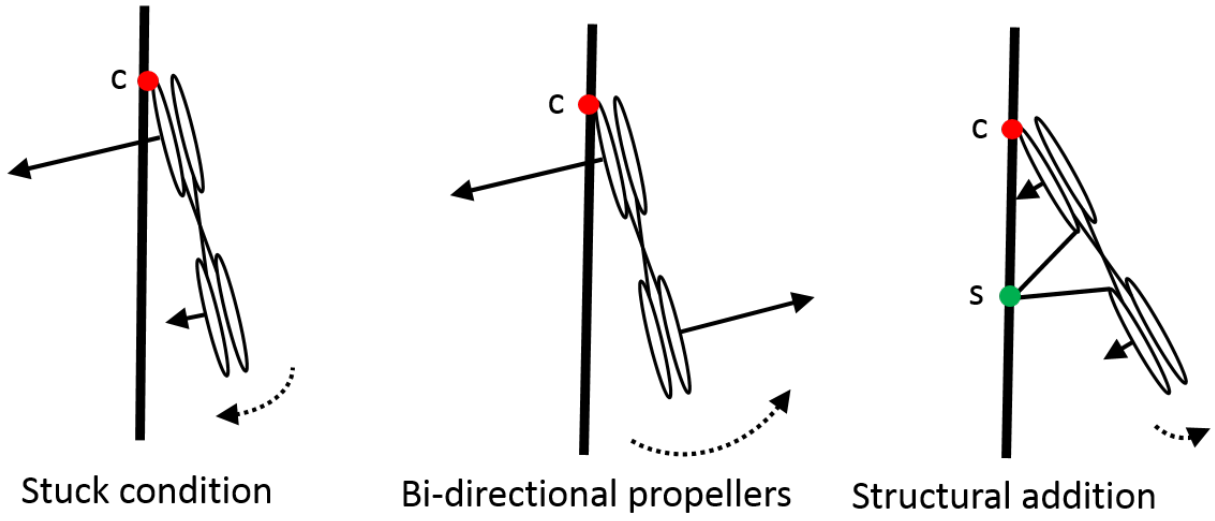
Figure 4–9: Horizontal drift for simulations which recovered

4.3.4 Failed Recoveries

Failures only occurred when the quadrotor became vertically 'stuck' against the wall. In general, when a quadrotor is nearly vertically stuck to the wall, it is impossible to recover using only uni-directional propellers. The quadrotor can either become vertical while initially pointing upwards or pointing downwards. In either case, the quadrotor will generate torque into the wall about the some point c as depicted in the left images of Figure 4–10. This moment is counteracted by the moment produced by gravity in the case that the quadrotor is still pointing upwards. If it is pointing downwards, gravity pulls the quadrotor into the wall. Once the quadrotor reaches some critically vertical inclination angle, it can no longer escape the stuck condition.



(a) Pointing upwards



(b) Pointing downwards

Figure 4–10: Vertical 'stuck' condition with two solutions

To deal with this mode of failure, there are two solutions that could be implemented, which are depicted the center and right schematics of Figure 4–10. One option is to use bi-directional instead of uni-directional propellers to generate a moment away from the wall. Alternately or additionally, a structural component could be added to the top of the quadrotor which would prevent it from passing its critical angle by physically contacting the wall at some point s.

4.4 Summary

In summary, a collision recovery strategy was developed and tested in simulation using a contact dynamics simulator between a propeller protected quadrotor and a wall plane. Monte Carlo simulations demonstrate a very high success rate of recovery but also that not all types of collisions can be recovered from. The range of recoverable collisions can be parametrized in terms of inclination angle and incoming velocity. The rate of failure to recover is strongly influenced by the magnitude of friction between the quadrotor's bumpers and the wall. Height loss can be significant for collisions which almost became stuck to the wall but which manage to rebound away, and horizontal drift is greatest for collisions which are characterized with large CRI *values. The quadrotor only fails to recover if it enters the stuck condition, and this singular mode of failure can only be entirely avoided if hardware changes are made.*

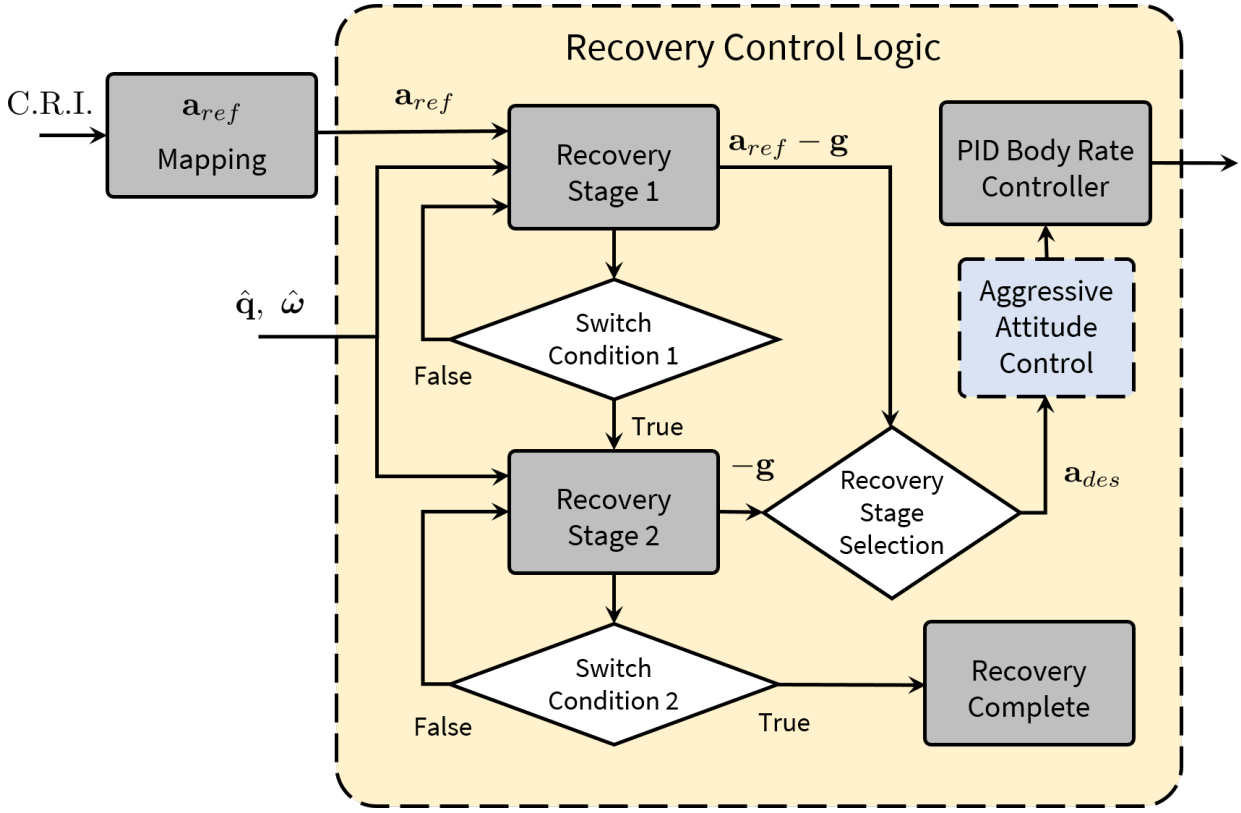
CHAPTER 5
Experiments on Collision Recovery

5.1 Overview

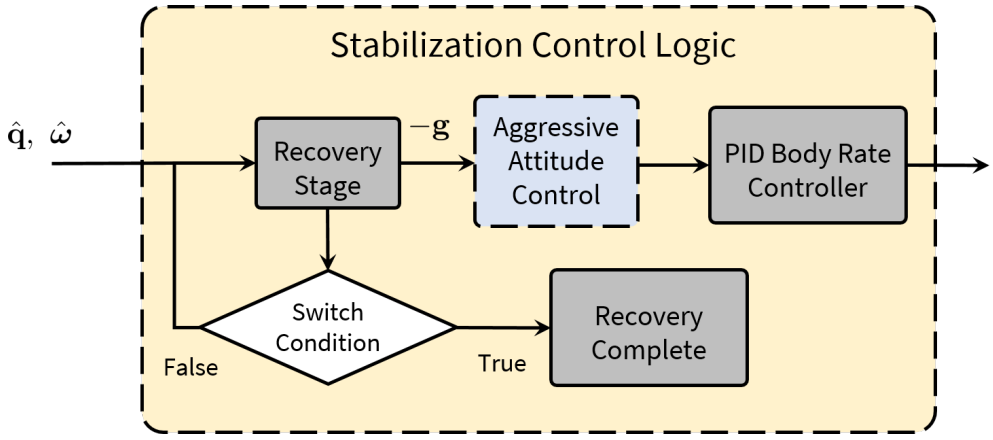
The recovery pipeline was implemented on the same Navi quadrotor that was used to test stabilization control in Chapter 3. The number and testable range of experiments was limited as compared to the initial conditions that could be generated in simulation, due to several technical issues. A total of 51 trials were performed using the collision recovery pipeline, and to assess the need for the full recovery pipeline, an additional 42 trials were performed where upon collision detection, stabilization control alone attempted to recover the quadrotor to a hover orientation, i.e., the fuzzy logic collision characterization block in Figure 4–3 was omitted. We will refer to the first set of 51 trials as the *recovery control set*, and the second set of 42 trials as the *stabilization control set*.

5.2 Differences from Simulation

The collision detection condition was the same for both sets of trials. For stabilization control set trials, $\mathbf{a}_{ref}$ was always set to zero and there was only a single recovery stage, which was RS-2. For both sets, RS-3 was removed from the recovery pipeline for reasons discussed below. These modifications resulted in the 'reduced' pipelines for recovery and stabilization control, as illustrated in Figure 5–1.



(a) Recovery control logic



(b) Stabilization control logic

Figure 5–1: Logic for recovery and stabilization control pipelines

For the recovery control set, $\mathbf{a}_{ref}$ was computed differently from simulation. The mapping from CRI to $\mathbf{a}_{ref}$ was modified from that of simulation as

$$||\mathbf{a}_{ref}|| = \begin{cases} g/2\ CRI,\ CRI > 0 \\ 0,\ \text{otherwise} \end{cases} . \tag{5.1}$$

The motivation for this modification was to ensure that quadrotors colliding with the wall at negative inclinations would not attempt in RS-1 to track an $\mathbf{a}_{ref}$ pointing into the wall. Although simulation did generate $\mathbf{a}_{ref}$ pointing into the wall for negative CRI values, in experiments the mapping shown in (4.5) seemed to produce an overreaction into the wall. Also, the scaling factor from CRI to $\mathbf{a}_{ref}$ was cut in half from that of simulation, in order to produce less aggressive orientations away from the wall and reduce the horizontal drift discussed in Section 4.4.

5.3 Motion Capture State Estimate

All trials were performed using a Vicon external motion capture system. The estimate of the quadrotor's state under Vicon motion capture was performed by the px4 flight stack which uses a complementary filter for state estimation. The px4 state estimator contains a number of gain parameters to weigh the reliability and noise of the various sensor measurements. As illustrated in Figure 5–2, the altitude estimate is fed barometer, accelerometer, and motion capture data; the attitude is estimated from IMU and motion capture data; the horizontal position and velocity is estimated from motion capture data fused with accelerometer measurements.

Due to the software architecture of the px4 flight stack, we were forced to tune the gains such that Vicon data was weighted heavily in both the attitude and altitude estimates. Unfortunately, we were unable to tune the state estimate parameters in such a way that when

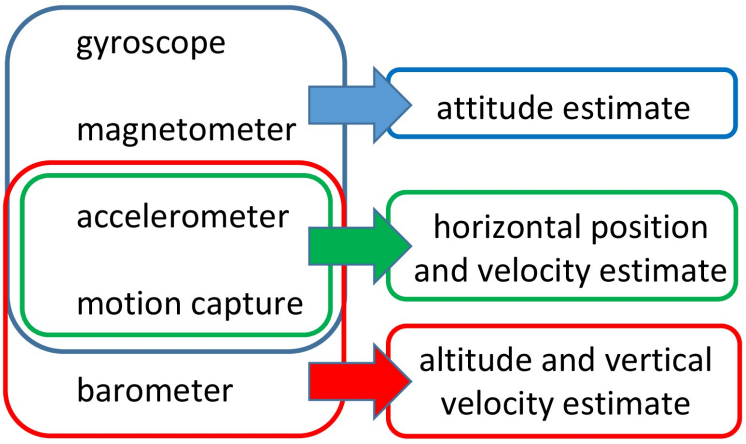


Figure 5–2: px4 estimates from sensor measurements

Vicon motion capture data cut out, the attitude and altitude estimates remained stable. As a consequence, the recovery procedure was *not* performed using only onboard sensing, which was one of the research objectives. Furthermore, due to the possibility of Vicon cut-outs, for safety reasons it was decided to drop stage RS-3 in which vertical velocity is meant to be autonomously stabilized. The risk was that the altitude control with a faulty altitude estimate would cause the quadrotor to shoot up into the ceiling or drop to the floor. Thus, although altitude was quickly stabilized once manual control was returned to the pilot, the research objective of autonomous altitude stabilization was not met in experiments.

5.4 Setup and Procedure

A safety net was erected above the laboratory floor to protect the quadrotor from breakage if it were to strike the ground. In each trial, the quadrotor was manually armed, taken off, and flown to a location near the wall. It was then switched into the px4's 'offboard mode' which specified a positional set-point some distance away from the wall and a yaw angle. At this point, the operator changed the positional setpoint to be 'inside' the wall, causing the quadrotor to fly into the wall at varying speeds and inclinations. Upon collision

detection, recovery or stabilization control was executed. When recovery was complete, manual control was returned to the pilot who landed the quadrotor, or if the quadrotor failed to recover, the pilot killed power to the motors to avoid further damage.

Software was developed on the Odroid in ROS to pass the Vicon data to the Pixhawk and to send position set-point commands to the Pixhawk. The ROS package 'mavros' was used to send and receive Mavlink data between the Odroid and the Pixhawk. Figure 5–3 summarizes these nodes, which can be found along with documentation on Github in the private repository *McGill-AML*.

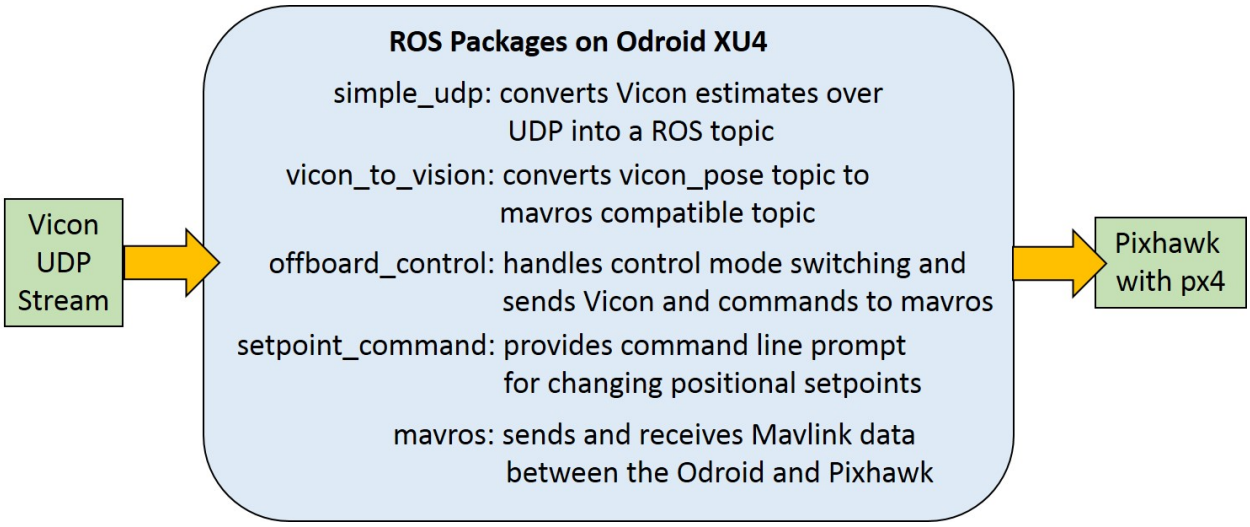


Figure 5–3: ROS nodes for position control

The experimental trials were executed for the range of initial conditions summarized in Table 5–1. The inclination angles and incoming velocities were varied incrementally, with the highest inclination angle at 21° into the wall and the highest incoming velocity at 3.3 m/s. Inclination angles away from the wall were also tested, up to 12°. The initial yaw angle was set discretely to be either 0° or 45°, in order to generate initial impacts on a single bumper as well as two bumpers. However, the further effect of initial yaw angle was not investigated

Table 5–1: Parameters and Ranges for Recovery Control

Value/Range	Units	Description
$-10.5 < \zeta < 21.5$	$^\circ$	Inclination angle
$\psi = 0 \ \vee \ \psi = 45$	$^\circ$	Initial yaw
$0.5 < \dot{X} < 3.3$	m/s	Incoming velocity

systematically, as no significant differences could be found between trials impacting on one bumper vs. two bumpers.

5.5 Results

Both recovery and stabilization control performed well for the tested range of initial conditions. The range of testable initial conditions was limited by the insufficiently strong and rigid bumper material, which often cracked and required repairs. Figure 5–4 shows the inclination angles and incoming velocities for the complete set of 93 trials conducted, out of which there were three failure cases, all due to identifiable technical problems.

5.5.1 Failure Cases

Two of the three failure cases were due to Vicon cutting out which corrupted the attitude estimate. The other failure was due to a bug in the code which had caused some collision parameters to not reset correctly. One of the trials for which Vicon cut out entered the 'stuck condition' described in the previous chapter. This failure occurred in the recovery control set at the highest tested inclination angle (21°). Had Vicon continued to provide correct measurements, this inclination should have been recovered from according to the results of Monte Carlo simulations. Besides these three failure cases, all other 90 trials successfully recovered from their collisions.

Snapshots from the trial which became stuck against the wall are shown in Figure 5–5. In sub-image a) we see the quadrotor impacting at an inclination angle of 21° and 1.1 m/s

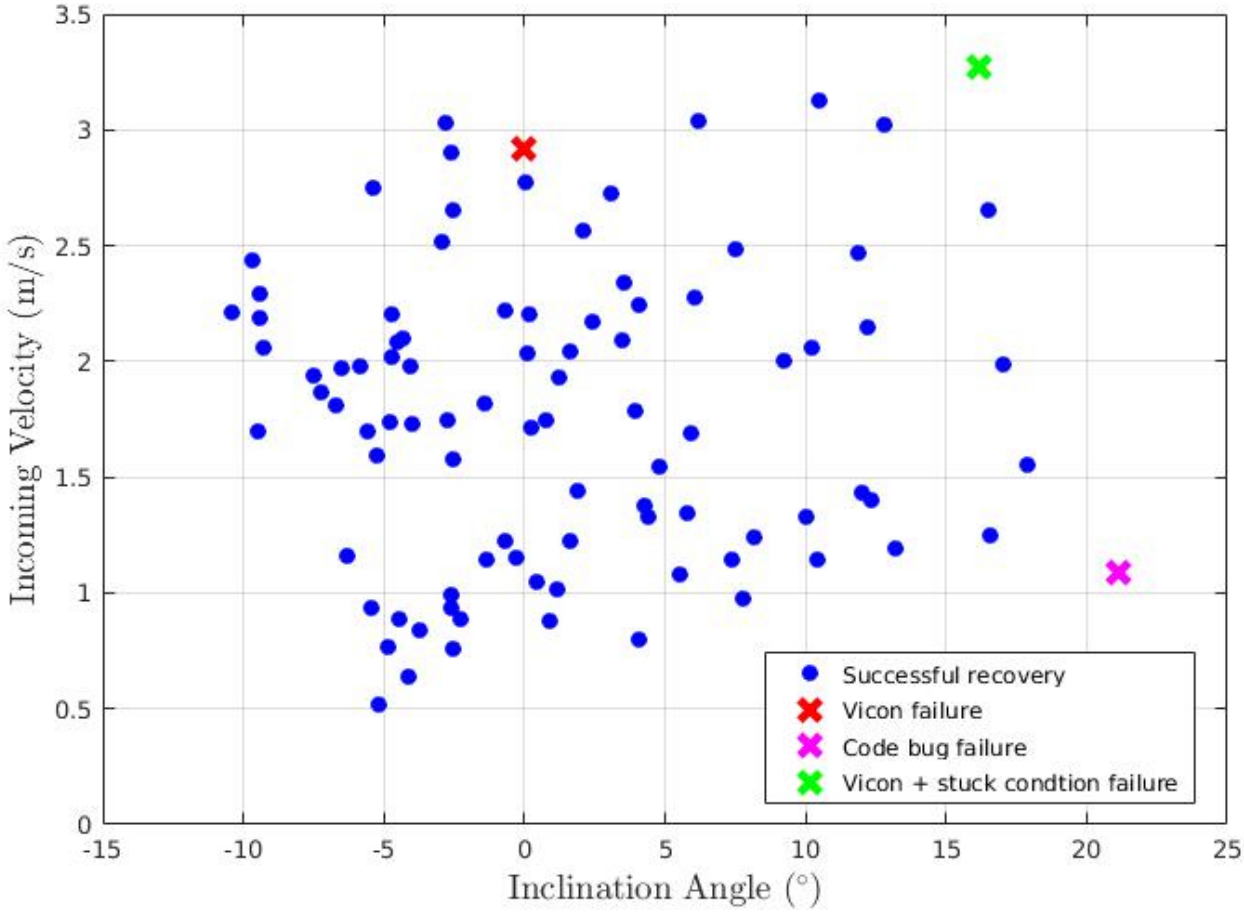


Figure 5–4: Experimental successes and failures

and rotating vertically in b). Vicon measurements were lost at 0.2 seconds after the impact, at about the same time as a second collision occurred in c). This second collision generates a strong moment in d) and e) and without a functioning recovery controller, it pulled the quadrotor vertically flush against the wall. Finally in f) we see the inevitable loss in height as the quadrotor is unable to remove itself from the wall. At this point, even if Vicon had not cut out, the quadrotor would have been unable to recover.

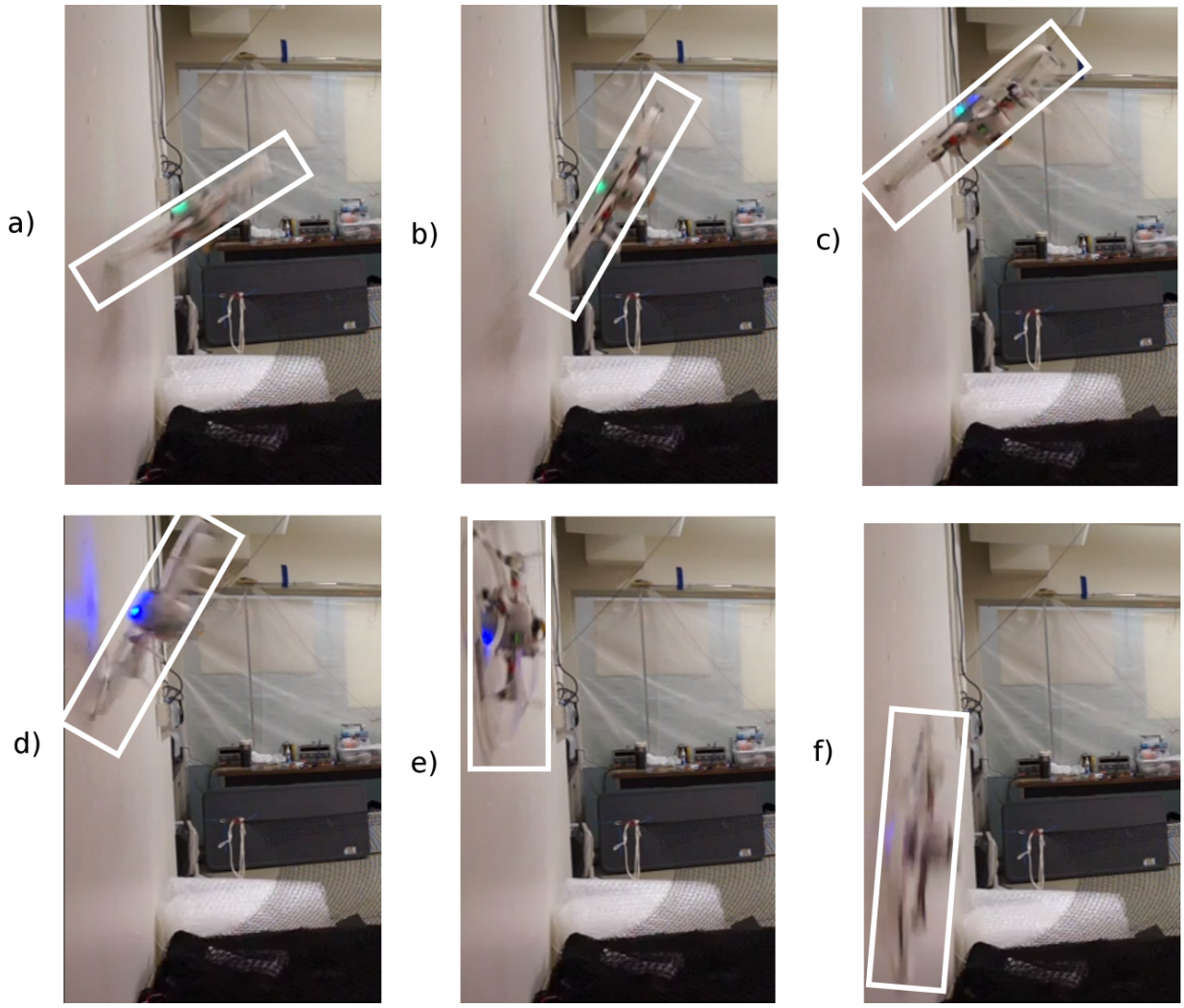


Figure 5–5: Snapshots of trial demonstrating the stuck condition

5.5.2 Example of Successful Recovery

An example of a successful recovery from the recovery control set is shown in Figure 5–6. The initial collision in sub-image a) shows a relatively low inclination angle (12.7°) which increases to an almost vertical orientation in b) and c), during which time the recovery control is attempting to counteract the moment induced by the collision. In d) we see the quadrotor becoming more upright, and by e) it has reached its desired orientation pointing

away from the wall. Having completed RS-1, the quadrotor returns to hover in f) at which time recovery is complete.

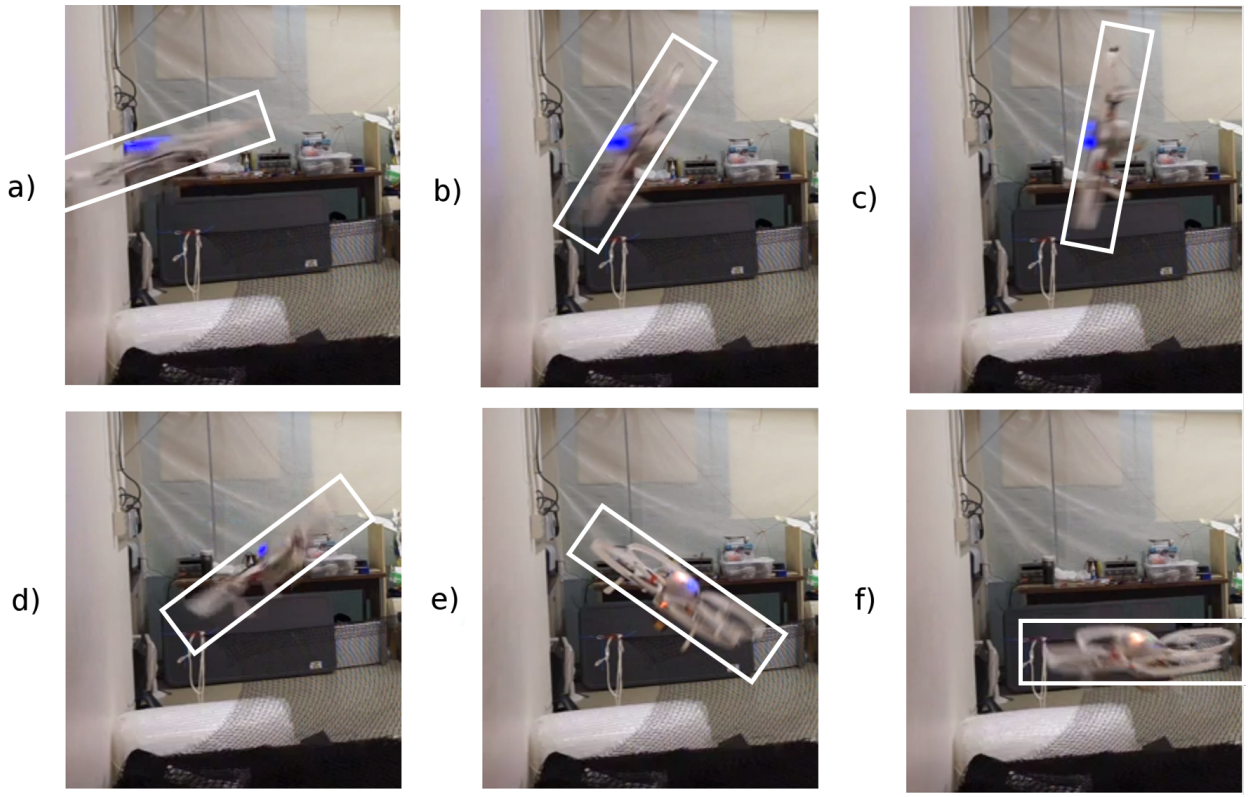


Figure 5–6: Snapshots of trial demonstrating successful recovery

Some time histories are shown for this example trial in Figure 5–7, where the solid vertical line denotes the time of collision detection and the dashed vertical line denotes the time of recovery at 0.77 sec. The pitch angle travels between -75°, its maximum negative, and 45°, its maximum positive value, in 0.3 seconds, at a maximum body rate of over 600 °/sec. The roll angle is minimally affected by the collision. The incoming velocity, $\dot{X}$, linearly increases to 3.3 m/s at the time of collision, drops immediately to near zero, quickly increases toward the wall again while the quadrotor is inclined toward the wall, and finally becomes negative as the quadrotor points away from the wall. The velocity along $\dot{Y}$ remains almost

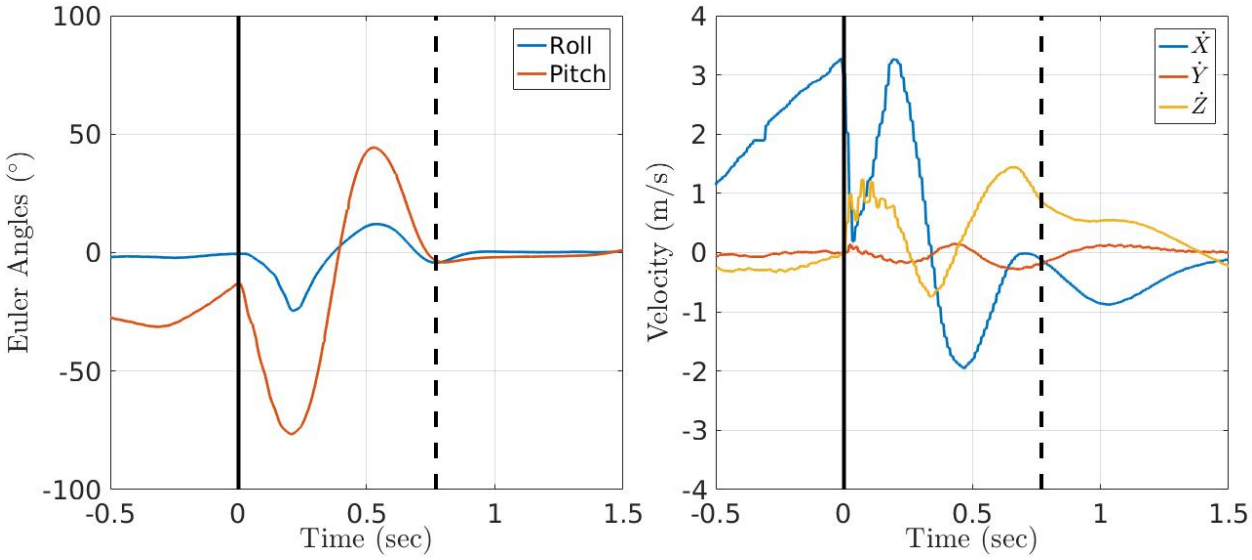


Figure 5–7: Time histories for successful recovery example

unchanged. The vertical velocity, $\dot{Z}$, jumps upward upon collision as the quadrotor rotates into the wall, becomes slightly negative while the quadrotor rotates away from the wall, and becomes positive again once the orientation is near hover at about 0.5 sec.

5.5.3 Comparison of Recovery and Stabilization Control Sets

The height loss, horizontal drift, and recovery time for the stabilization control set were all lower than for recovery control set.

Figure 5–8 compares recovery control on the upper plots to stabilization control on the lower plots for the measured variables of height loss, horizontal drift and recovery time. Each of these was measured from the time of collision detection to the time that RS-2 was completed. On the left hand side of Figure 5–8, a positive value implies a loss in height, so between recovery control and stabilization control, we see that stabilization control had less instances of significant height loss. In the middle column of Figure 5–8, we clearly see that there was no horizontal drift for the stabilization control set, whereas the recovery control set had an even distribution for trials which generated non-zero reference accelerations away

from the wall. Horizontal drift is inevitable using RS-1, since the quadrotor intentionally points away from the wall for some period of time before returning to hover, which induces horizontal velocity. Finally, the right hand side of Figure 5–8 shows a significant reduction in recovery time when RS-1 is omitted, with the largest recovery time at about 0.6 s vs. 1.5 s using RS-1. This result is reasonable and is confirmed by simulation, since the requirement for the quadrotor to point away from the wall before returning to hover should add time to the recovery procedure.

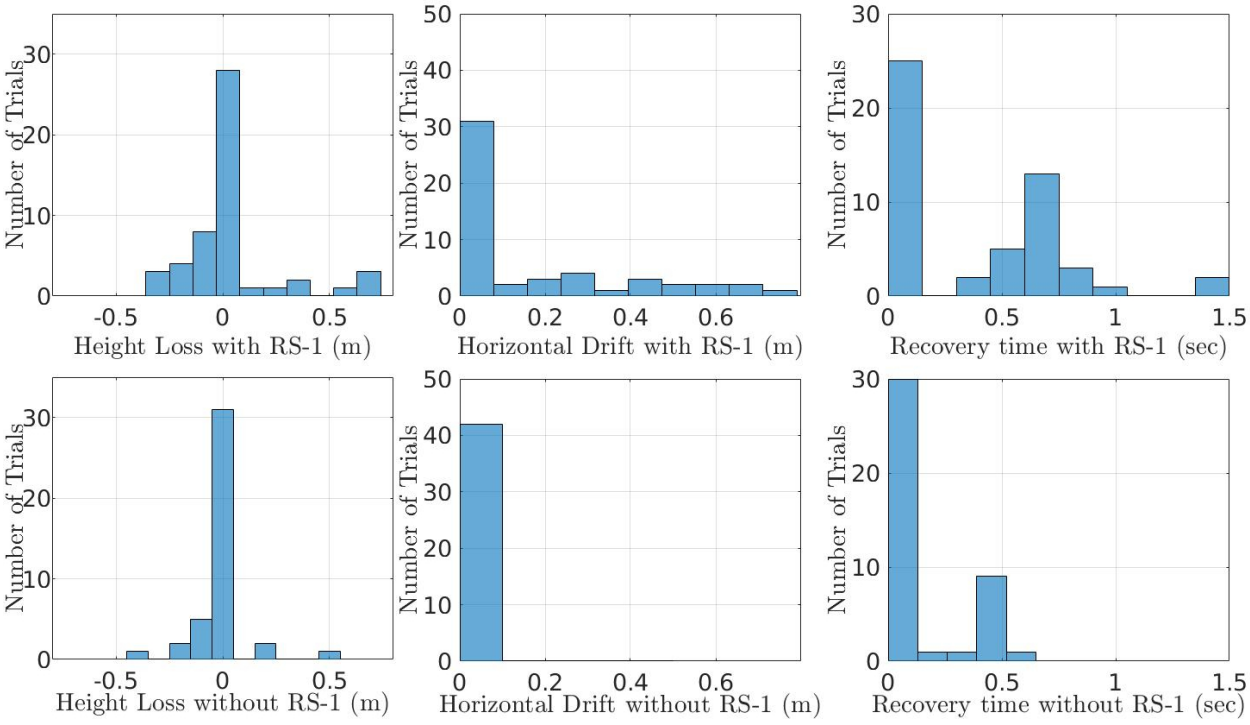


Figure 5–8: Comparison between recovery control and stabilization control

It is reasonable to conclude that stabilization control is a good solution for collision recovery at least for the tested range, and full recovery control pipeline is unnecessary for this range.

5.6 Summary

The range of initial conditions for experiments was relatively limited, due to insufficiently strong and rigid bumper material. It was necessary for the state estimator to make use of Vicon motion capture measurements which failed to meet the research objective of using only onboard sensing. Still, the results of these experiments do demonstrate that recovery control behaves as expected from simulation when technical issues do not arise. This control law can be implemented in real-time on any quadrotor with propeller protection.

CHAPTER 6
Conclusion

6.1 Summary of Research

The reorientation control law investigated in this thesis served as the basis for a collision recovery strategy. As a first attempt at collision recovery, overall the control law performed well. Simulations demonstrated that the reorientation control law could be used to stabilize a quadrotor from a wide range of initial conditions to a state of hover. This stabilization control was implemented on a custom built 1.1 kg quadrotor which was thrown into the air, detected free-fall, and stabilized its attitude and altitude. The reorientation control law was not able to recover flight control from all types of wall collisions, but the reasons for failures were investigated and improvements proposed. In experiments, almost a hundred collision trials demonstrated that the reorientation control law almost always succeeded for the range of initial conditions that were successful in simulation. Both a stabilization control strategy and a full collision recovery control strategy were implemented, and their performance was comparable.

6.2 Main Conclusions

Several conclusions can be drawn from the various stages of this work, which will now be listed.

The control law of [39] attempts to control the acceleration of the center of mass of a quadrotor. This task is only feasible for a certain range of reference accelerations which is limited by the thrusting acceleration the quadrotor's propellers can generate. A prerequisite for tracking any acceleration is that the quadrotor is able to orient its propellers in any

direction. This requires a globally valid control law which implies a global description of attitude. This can be achieved by generating body rate set-points from errors on individual elements of a unit quaternion description of attitude, and performing PID body rate control.

In stabilizing a quadrotor from a random throw into the air, the most challenging scenarios were those which began with large yaw rates. The yaw rate should be stabilized first so that the quadrotor is no longer spinning wildly. As long as this is achieved, reorienting the quadrotor to the orientation of hover is relatively trivial and can be performed in under half a second. Height loss can reach several meters if the yaw rate is initially high, but theoretically a quadrotor should eventually stabilize to hover as long as its attitude estimate remains accurate and its propeller speeds can change quickly enough.

The motion that a propeller protected quadrotor colliding with a wall undergoes is challenging to model and characterize accurately, and it is challenging to react to intelligently. However, there is only a single situation which is unrecoverable from, which is when the quadrotor becomes vertically stuck to the wall. The main objective for recovery is therefore to avoid this stuck condition. If the quadrotor begins to rotate into the wall following the collision, it must quickly ensure that it counters this rotation, making use of its full range of propeller thrust differences.

To achieve this away-from-the-wall rotation, only the roll and pitch of the quadrotor are important to control. The yaw angle is irrelevant during recovery. The reorientation control law is well suited to this requirement. It is formulated to control the orientation of the quadrotor's thrusters in a particular direction and to apply thrust in that direction.

In experiments, two different strategies were implemented. The trials in the recovery set brought the quadrotor to point away from the wall before achieving hover, whereas the stabilization trials immediately brought the quadrotor to hover. It was found that it is

unnecessary to point the quadrotor away from the wall to achieve collision recovery. This implies that an estimate of the wall normal is not necessary, nor is a characterization of the intensity of the collision with the wall. It is possible that there are circumstances where these quantities are still useful, such as for grazing collisions with walls, which were not explored.

6.3 Suggestions for Future Work

There are several directions that could be pursued for future work.

Stabilization control could add the stage of stabilizing horizontal velocity. The switch conditions between stages could be improved to transition more seamlessly. Downward facing height sensors could be integrated to bring the quadrotor to a particular altitude following a throw. Under different lighting and terrain conditions, infrared and sonar sensors suffer from noise which could be investigated in more detail. Optical flow could be tested on a downward facing camera and full stabilization of position shown for a wide range of outdoor conditions. Stabilization under strong wind gusts could also be investigated.

One direction that was not pursued for collision recovery was to drop the thrust of the quadrotor very low before attempting reorientation. This would of course induce height loss, but it could be an appropriate reaction if the quadrotor is in danger of entering the stuck condition. The idea would be to allow the quadrotor to rebound naturally from the collision with the wall without continuing to provide thrust into the wall, which may play a large role in creating the stuck condition initially.

Another interesting area of research is the detection of collisions. Little literature has addressed this problem. Without knowledge of the forces being generated by the propellers, it is challenging to estimate the external wrench on the quadrotor. However, a purely kinematic solution using an IMU could be developed where the body rate errors and spikes in the accelerometer are checked together to determine if a collision has occurred and of what

kind. For instance, if the body rate error suddenly spikes but the accelerometer only slightly spikes, the collision is probably a flip over a cable or an edge. However, if the accelerometer does spike but the body rate errors do not, then it is probably a head on collision. A heuristic or statistical model could be developed to distinguish between these types of collisions.

The experiments for collision recovery were unfortunately performed using motion capture feedback in the state estimate. This was only done to generate repeatable position set-points into the wall, and is otherwise unnecessary. The experimental set should be repeated without using motion capture feedback, to demonstrate that only onboard sensing is necessary to achieve collision recovery.

A wider range of incoming conditions could be tested for impacts with the wall so that more failure cases could be demonstrated experimentally. Systematically varying single parameters such as the incoming yaw angle would be informative to see if failures occur more readily when the collision is on one vs. two bumpers. If a pilot were to manually fly the quadrotor into the wall, they could randomly vary the incoming angles, body rates, and speeds. Grazing collisions could be tested, as well as collisions accelerating upward and/or downward into the wall. There is a large space of collisions types, each of which could be investigated in more detail.

Considering that the reorientation control law is the most aggressive possible attitude controller, little can be done, besides PID gain tuning, to improve the recovery time following a collision. Assuming that reorientation is being achieved as fast as possible, then further improvements have to be made at the hardware level.

A major limitation for experiments was the continual breaking of the propeller protection. Several versions of propeller protection were built. The design trade-off was between too much flexibility and too much fragility. In effect, the propeller protection either contacted

and stalled the propellers, or they shattered. PLA, ABS, and nylon alloys were attempted as the main structural component, with carbon fiber tubing connecting each bumper to the next. However, this design has a great deal of room for improvement. For instance, the propeller protected racing drone shown in Figure 1–1 uses carbon fiber plates above and below the entire quadrotor, connected by screwed aluminum struts at many locations. While this design is significantly heavier, it is also much more rigid, less fragile, and can handle impacts from any direction.

Ideally, propeller protection is soft on the outside and hard on the interior, so that impacts have a slight rebound or elasticity but the propellers are still not touched by the protection. Redesign of propeller protection could consider some lightweight rubbery or high density foam material on the outside, with carbon fiber or aluminum on the interior.

Adding structural material to the top of the quadrotor's propeller protection was suggested earlier in the thesis as a possible way to avoid the stuck condition. It should be possible to build a quadrotor that can recover from every collision with a wall if a correct geometry of structural material is added.

One interesting direction that was not investigated is the use of bi-directional propellers. Uni-directional propellers make the stuck condition irrecoverable, but with bi-directional propellers it would still be possible to recover because a moment could be generated away from the wall. This would require fundamental modifications in the recovery control law, allowing for nearly double the range of moments. Investigating bi-directional propellers for quadrotors in general is an excellent direction for future work. They could also be used to flip the quadrotor upright if it lands upside down on the ground.

In the long term, collisions with other objects such as poles and cables should also be investigated. If the quadrotor were aware of all of the objects in its vicinity, a full collision

recovery solution would be able to identify the type of collision and decide on the best way to remove itself from any and all stuck conditions. For instance, it would be able to identify the difference between getting caught on the corner of a building and clotheslining over a cable. It would react intelligently from the wide variety of collisions that can occur. For this, reinforcement learning seems like a promising direction for collision recovery. If the quadrotor has access to a large data set of collision types and all the best thrusting profiles which have worked for other quadrotors in the past, it could draw on this information for its own reactions.

Beyond this, collisions with mobile objects should be considered but these will likely require prediction and estimation of the other objects' motion, which is indeed a challenging problem using only onboard sensing. In the future, it may be possible to consider such complex tasks; for the time being, there is still plenty of research to be done on the above-mentioned collision scenarios.

REFERENCES

[1] Donald M Atwater. The commercial global drone market. gbr.pepperdine.edu/2015/10/emerging-opportunities-for-social-and-environmental-uses-of-uavs/, 2015. Accessed October, 2016.

[2] Yash Mulgaonkar, Gareth Cross, and Vijay Kumar. Design of small, safe and robust quadrotor swarms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2208–2215. IEEE, May 2015.

[3] Gretchen West. Drone on. https://www.foreignaffairs.com/articles/2015-05-01/drone, May 2015. Accessed October, 2016.

[4] Wayne E Woldt, Eric W Frew, Maciej Stachura, Jacob Smith, and James Mack. Conducting unmanned aircraft flight operations under federal aviation administration regulations. In *2015 ASABE Annual International Meeting*, page 1. American Society of Agricultural and Biological Engineers, July 2015.

[5] H Alvarez, LM Paz, J Sturm, and D Cremers. Collision avoidance for quadrotors with a monocular camera. In *Experimental Robotics*, pages 195–209, October 2016.

[6] Yingc ai Bi, Jiaxin Li, Hailong Qin, Menglu Lan, Mo Shan, Feng Lin, and Ben M. Chen. An mav localization and mapping system based on dual realsense cameras. In Prof. Zhihong PENG and Dr. Feng LIN, editors, *International Micro Air Vechicle Competition and Conference 2016*, pages 50–55, Beijing, PR of China, October 2016.

[7] Mark W Mueller and Raffaello D'Andrea. Relaxed hover solutions for multicopters: Application to algorithmic redundancy and novel vehicles. *The International Journal of Robotics Research*, pages 873–889, October 2015.

[8] Raffaello D'Andrea, Sergei Lupashin, Mark W Mueller, and Markus Waibel. Controlled flight of a multicopter experiencing a failure affecting an effector, April 2016. US Patent 20,160,107,751.

[9] Gabriel M Hoffmann, Haomiao Huang, Steven L Waslander, and Claire J Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Proceedings of the AIAA Guidance, Navigation, & Control Conference*, volume 2, August 2007.

[10] Brendan Galea, Ehsan Kia, Nicholas Aird, and Paul G Kry. Stippling with aerial robots. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, pages 125–134. Eurographics Association, 2016.

[11] Bora Erginer and Erdinc Altug. Modeling and PD control of a quadrotor vtol vehicle. In *2007 IEEE Intelligent Vehicles Symposium*, pages 894–899. IEEE, June 2007.

[12] Samir Bouabdallah, Andre Noth, and Roland Siegwart. PID vs LQ control techniques applied to an indoor micro quadrotor. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2451–2456. IEEE, September 2004.

[13] Mingfeng Zhang, Adam Harmat, and Inna Sharf. Autonomous flight of a quadrotor using multi-camera visual slam. In *Proceedings of International Conference on Intelligent Unmanned Systems*, volume 10, May 2014.

[14] Teodor Tomić and Sami Haddadin. A unified framework for external wrench estimation, interaction control and collision reflexes for flying robots. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4197–4204. IEEE, September 2014.

[15] Christopher D McKinnon and Angela P Schoellig. Unscented external force and torque estimation for quadrotors. *arXiv preprint arXiv:1603.02772*, August 2016.

[16] Fiona Chui. Quadrotor collision dynamics and fuzzy logic characterization. Master's thesis, McGill University, 2016.

[17] H Bouadi, M Bouchoucha, and M Tadjine. Sliding mode control based on backstepping approach for an uav type-quadrotor. *World Academy of Science, Engineering and Technology*, 26(5):22–27, 2007.

[18] Dario Brescianini, Markus Hehn, and Raffaello D'Andrea. Nonlinear quadrocopter attitude control. Technical report, October 2013.

[19] Taeyoung Lee, Melvin Leoky, and N Harris McClamroch. Geometric tracking control of a quadrotor uav on SE(3). In *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425. IEEE, December 2010.

[20] Dario Brescianini, Markus Hehn, and Raffaello D'Andrea. Quadrocopter pole acrobatics. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3472–3479. IEEE, November 2013.

[21] Christopher G Mayhew, Ricardo G Sanfelice, and Andrew R Teel. Quaternion-based hybrid control for robust global attitude tracking. *IEEE Transactions on Automatic Control*, 56(11):2555–2566, January 2011.

[22] Paul Pounds, Robert Mahony, and Peter Corke. Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 18(7):691–699, July 2010.

[23] Dingjiang Zhou. Quadrotor dynamics and the differential flatness theory based aggressive control. Technical report, Tech. Rep., Boston University, 2013.

[24] Samir Bouabdallah and Roland Siegwart. Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 2247–2252. IEEE, April 2005.

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

[26] Sergei Lupashin, Angela Schöllig, Michael Sherback, and Raffaello D'Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1642–1648. IEEE, May 2010.

[27] Jeremy H Gillula, Haomiao Huang, Michael P Vitus, and Claire J Tomlin. Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1649–1654. IEEE, May 2010.

[28] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Control of complex maneuvers for a quadrotor uav using geometric methods on SE(3). *arXiv preprint arXiv:1003.2005*, May 2010.

[29] Davide Scaramuzza, Michael C Achtelik, Lefteris Doitsidis, Fraundorfer Friedrich, Elias Kosmatopoulos, Agostino Martinelli, Markus W Achtelik, Margarita Chli, Savvas Chatzichristofis, Laurent Kneip, et al. Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in gps-denied environments. *IEEE Robotics & Automation Magazine*, 21(3):26–40, August 2014.

[30] Volker Grabe, Heinrich H Bülthoff, and Paolo Robuffo Giordano. On-board velocity estimation and closed-loop control of a quadrotor uav based on optical flow. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 491–497. IEEE, June 2012.

[31] Raul Mur-Artal, JMM Montiel, and Juan D Tardós. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, August 2015.

[32] Adam Harmat, Michael Trentini, and Inna Sharf. Multi-camera tracking and mapping for unmanned aerial vehicles in unstructured environments. *Journal of Intelligent & Robotic Systems*, 78(2):291–317, May 2015.

[33] José Martínez-Carranza, Nils Loewen, Francisco Márquez, Esteban O García, and Walterio Mayol-Cuevas. Towards autonomous flight of micro aerial vehicles using orb-slam. In *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pages 241–248. IEEE, November 2015.

[34] Philippe Martin and Ioannis Sarras. A simple model-based state estimator for the quadrotor using only inertial measurements. *arXiv preprint arXiv:1510.03249*, April 2015.

[35] Abdelhamid Tayebi and Stephen McGilvray. Attitude stabilization of a vtol quadrotor aircraft. *IEEE Transactions on Control Systems Technology*, 14(3):562–571, April 2006.

[36] Ern J Lefferts, F Landis Markley, and Malcolm D Shuster. Kalman filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, 5(5):417–429, October 1982.

[37] Lorenz Meier, Petri Tanskanen, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots*, 33(1-2):21–39, August 2012.

[38] Farhad A Goodarzi and Taeyoung Lee. Extended kalman filter on SE(3) for geometric control of a quadrotor uav. *arXiv preprint arXiv:1605.02031*, May 2016.

[39] Matthias Faessler, Flavio Fontana, Christian Forster, and Davide Scaramuzza. Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor. In *IEEE International Conference on Robotics & Automation (ICRA)*, May 2015.

[40] Fiona Chui, Gareth Dicker, and Inna Sharf. Dynamics of a quadrotor undergoing impact with a wall. In *Unmanned Aircraft Systems (ICUAS), 2016 International Conference*, pages 717–726. IEEE, June 2016.

[41] Randal Beard. Quadrotor dynamics and control rev 0.1. `http://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=2324&context=facpub`, 2008. Accessed October, 2016.

[42] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35, January 2006.

[43] Emil Fresk and George Nikolakopoulos. Full quaternion based attitude control for a quadrotor. In *2013 European Control Conference (ECC)*, pages 17–19, July 2013.