UNIVERSITÉ DE MONTRÉAL

TRAJECTORY GENERATION FOR A QUADROTOR UNMANNED AERIAL VEHICLE

DOUGLAS CONOVER
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
SEPTEMBRE 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

TRAJECTORY GENERATION FOR A QUADROTOR UNMANNED AERIAL VEHICLE

présenté par : CONOVER Douglas
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

DEDICATION

To Granddad,
With every passing year I see more of you in me
but never enough to hand-cut a sunroof into my new car.

ACKNOWLEDGMENTS

I would like to thank my supervisor David Saussié, who two years ago gave me the chance to pursue my budding interest in aerospace and control. I would also like to acknowledge André Phu-Van Nguyen and Olivier Gougeon, who aided me in the experimental implementation of my work. Your knowledge of the lab and willingness to help when I hit a dead-end unquestionably saved me from hours of frustration. Thank you to Tien Nguyen, who was always there to entertain my questions and help with any theoretical issues I had. Thanks also to Justin Cano, your energy and enthusiasm was welcome every day I spent in the lab. Thank you to Maeve Reimer and Jorge Zavagno who helped me produce the composite photos found in this work.

Thank you to Pierre, Hugo, Vincent, Tangui, Clément, Adriel and the rest of the 4th floor lab for your lightheartedness and help throughout my Master's. When courses seemed overwhelming, I could always count on one of you for explanation and guidance.

Finally, I would like to thank my family for their love and support over the course of this degree. To my parents Cecilia and Kent, you kept me going with Sunday dinners and tupperwares full of pancakes. Thank you to my grandparents Paul and Betty, who supported me and never stopped encouraging me to follow my interests. Thank you to Cort, Sue, Eric, Richard and Ann-Margrit, who were always curious to hear where my studies were taking me. To my sister Rachel, who is still the only one who will watch scary movies with me. To my brother Jared, who won't watch scary movies, but is always willing to toss a frisbee around. I love you all.

RÉSUMÉ

Le domaine des véhicules aériens sans pilote de type multicoptères a connu une progression substantielle au cours de la dernière décennie. La génération et le contrôle des trajectoires ont été au centre des préoccupations de ce nouveau domaine, avec des méthodes qui permettent d'exécuter des manœuvres complexes dans l'espace. Plusieurs efforts ont été faits pour exécuter ces manœuvres en utilisant la commande non linéaire, notamment la commande par platitude différentielle. Cependant, l'absence de théorie pour l'estimation des dérivées d'ordre supérieur a empêché l'application expérimentale de plusieurs de ces techniques.

Ce travail explore tout d'abord l'approche par composition séquentielle pour l'exécution de manœuvres à travers des fenêtres étroites. Cette technique implique la combinaison de plusieurs contrôleurs théoriquement simples afin de produire un résultat complexe. Les résultats expérimentaux réalisés dans le Laboratoire de Robotique Mobile et de Systèmes Automatisés à Polytechnique Montréal démontrent la validité de cette approche, en produisant des manœuvres précises et répétables. Cependant, on atteint rapidement les limites d'une telle méthode dans les applications du monde réel, du fait de son manque de précision initiale et l'absence d'évaluation de faisabilité.

Ce mémoire se concentre ensuite sur le développement d'une architecture d'estimation d'état basée sur le filtre de Kalman linéaire afin de fournir en temps réel des estimés des 2^{e} et 3^{e} dérivées de la position d'un quadricoptère (appelées respectivement accélération, et à-coup ou *jerk*). Des filtres de complexités différentes sont développés afin d'incorporer toute l'information disponible sur le système pour améliorer l'estimé résultant. On obtient alors un estimateur d'état complet qui utilise les mesures de position et d'accélération, ainsi que les entrées de commande, et fournit des estimés pour la rétroaction. Un contrôleur du *jerk* augmenté basé sur la théorie de la commande optimale est ensuite développé afin de valider cet estimateur. Il est conçu de façon à utiliser le *jerk*, l'accélération, la vitesse et la position du drone ; sans rétroaction de chacun de ces termes, le système est alors instable. Des tests sont effectués afin d'examiner les performances de l'estimateur et du contrôleur. Tout d'abord, le quadricoptère est chargé de suivre diverses entrées de référence dans l'espace pour assurer sa stabilité. Le contrôleur permet de suivre au plus près ces références, comme réalisé en simulation. Le contrôleur doit ensuite suivre un changement de référence afin d'évaluer la précision de l'estimateur développé. Les résultats montrent que l'estimation en temps réel du *jerk* suit adéquatement les valeurs hors ligne. Pour autant que nous le sachions, c'est la première mise en œuvre dans le monde réel du retour de *jerk* pour contrôler un multicoptère.

ABSTRACT

The field of multirotor unmanned aerial vehicles (UAVs) has seen substantial progression in the past decade. Trajectory generation and control has been a main focus in this domain, with methods that enable the performance of complex three-dimensional maneuvers through space. Efforts have been made to execute these maneuvers using concepts of nonlinear control and differential flatness. However, a lack of theory for the estimation of higher-order derivatives of a multirotor UAV has prevented the experimental application of several of these techniques concentrated on trajectory control. This work firstly explores the existing control approach of sequential composition for the execution of quadrotor manoeuvres through narrow windows. This technique involves the combination of several theoretically simple controllers in sequence in order to produce a complex result. Experimental results conducted in the Mobile Robotics and Automated Systems Laboratory (MRASL) at Polytechnique demonstrate the validity of this approach, producing precise and repeatable manoeuvres through narrow windows. However, they also show the limitations of such a method in real world applications, notably its initial inaccuracy and lack of feasibility evaluation. This thesis then focuses on the development of a state-estimation architecture based on linear Kalman filter techniques in order to provide a real-time value of a quadrotor UAV's second and third derivatives (referred to as acceleration and jerk, respectively). Filters of different complexities are developed with the goal of incorporating all available system information into the resulting estimate. A full-state estimator is produced that uses a quadrotor's position and acceleration measurements as well as control inputs in order to be usable for feedback. A jerk-augmented controller based off of optimal control theory is then developed in order to validate this estimator. It is designed in such a way to use the UAV's jerk, acceleration, velocity and position as design parameters and to be unstable without feedback in each of these terms. Tests are conducted in order to examine the performance of both the estimator and controller. Firstly, the quadrotor is commanded to track various reference inputs in 3D space to ensure its stability. The controller tracks these references very closely to simulated responses. The controller is then asked to follow a changing reference in order to evaluate the precision of the developed estimator. Results show that the real-time estimation of the jerk follows offline values adequately. To the best of our knowledge, this is the first application to implement the feedback of a multirotor UAV's jerk in real-world experimentation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Core Concepts

Recent years have seen an explosion of commercial and academic applications of multirotor autonomous unmanned vehicles, or “UAVs”. These autonomous flying robots are characterized by their multiple rotating propellers fixed on a rigid frame. Their simple geometry and input force characteristics make them ideal for theoretical applications of controls engineering.

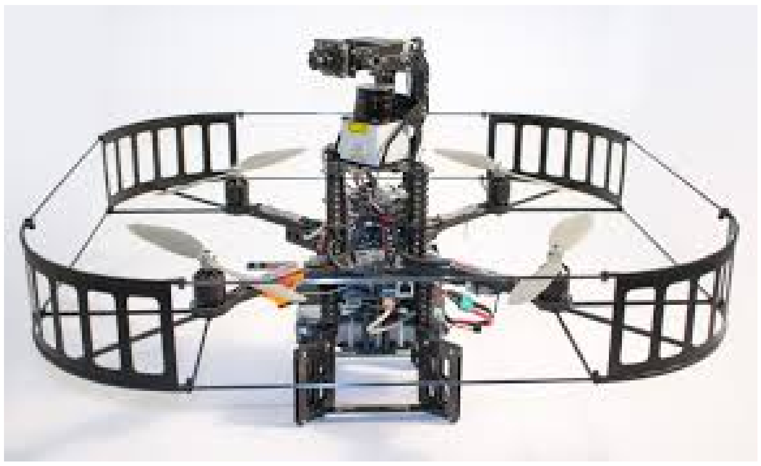

Figure 1.1 AscTec Pelican Quadrotor

Multirotor UAVs can be designed to have 4, 6 or 8 propellers and come in a variety of configurations. The AscTec Pelican, as seen in figure 1.1, flies using 4 propellers while the AscTec Firefly, pictured in figure 1.2 has a 6 rotor configuration.

Many laboratories have used multirotor UAVs as platforms for the development and testing of novel control techniques. Basic concepts of control and trajectory planning for a multirotor UAV have been outlined by Mahoney, Kumar, and Corke in [4]. They presented methods that enable the stabilization of a UAV as well as the ability to track 3D trajectories. In a more specialized application, researchers at ETH Zurich [5] developed a method to balance an inverted pendulum using a small quadrotor, as seen in figure 1.3.

Figure 1.2 AscTec Firefly Hexarotor

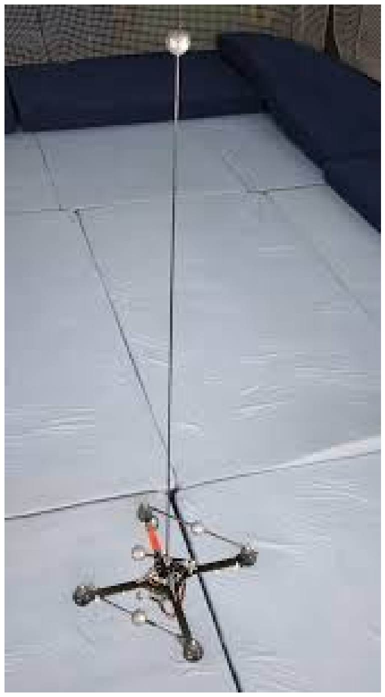

Figure 1.3 Balancing an Inverted Pendulum

There are many other possible uses for mulitrotor UAVs that have been explored by researchers in recent years. A collection of quadrotors can be used in order to safely monitor the behaviour of forest fires [6]. Borowczyk et al. developed a method to automatically land a quadrotor on moving surfaces up to 50 km/h, making retrieval of UAVs more feasible in high-speed situations [7].

A main field of study in the field is the generation and execution of trajectories in 3D space. Variations of this theme can permit a quadrotor to perch on an object, fly through narrow windows, navigate through cluttered environments and execute acrobatic manoeuvres. This work aims to treat several aspects of the problem of quadrotor trajectory generation.

1.2 Problem Definition

1.2.1 Trajectory Generation for Precise Aggressive Manoeuvres with a Quadrotor UAV

The initial focus of this Master's project was the replication of results produced in [1]. This paper had as a goal to autonomously fly a quadrotor through a narrow window at varying angles, as well as land on several differently inclined surfaces. Figures 1.4 and 1.5 show examples of the test results from this prior work.

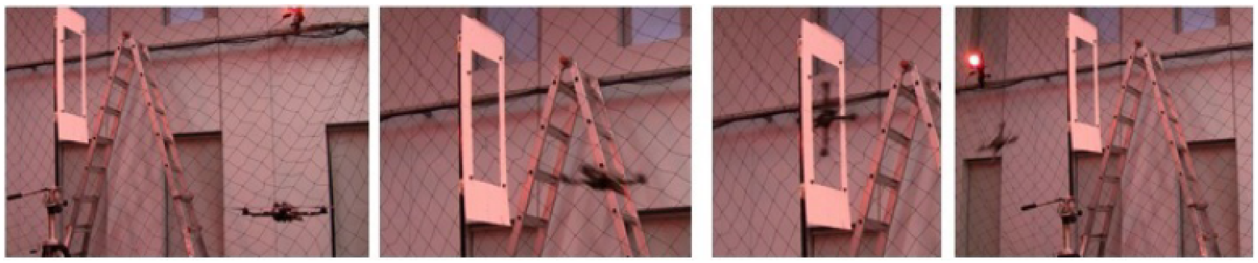

Figure 1.4 Problem Definition : Original results showing a quadrotor flying through a 90° window (taken from [1])

The main focus of the replication of this paper was on the successful passage of a quadrotor through narrow windows. Four of the cases were chosen, namely :

— Passage through a 45° vertical window

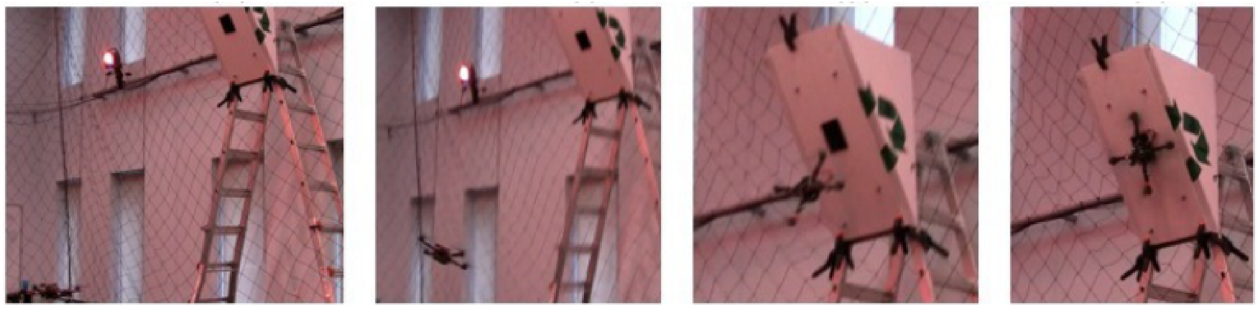

Figure 1.5 Problem Definition : Original results showing a quadrotor perching on a 120° surface (taken from [1])

— Passage through a 60° vertical window
— Passage through a 90° vertical window
— Descent through a horizontal window

To be considered a success, the quadrotor must autonomously take off, hover to a desired point in space, successfully navigate through a narrow window and recover to a stable hover at the end of the manoeuvre. This execution must be reliable and repeatable.

1.2.2 Higher-Order State Estimation

The treatment of trajectory generation and control for multirotor UAVs is sometimes approached with nonlinear control techniques. Some of these techniques involve the feedback of higher-order derivatives of the quadrotor's state. The 3rd derivative of an object's position with respect to time is often referred to as its "jerk" :

$$j = \frac{\mathrm{d}^3 x}{\mathrm{d}t^3} \tag{1.1}$$

When dealing with passenger vehicles, the jerk of an object or vehicle is related to the level of comfort experienced by the passenger. As multi-rotor vehicles grow large enough to accommodate human passengers, it could be useful to design a controller with the reduction of jerk in mind. Unfortunately, due to a lack of available sensors it is difficult to measure the jerk of a UAV. In the case of a large portion of academic applications, the available measurements when dealing with multirotor UAVs are :

— World-frame position from an external motion capture system
— Linear body-frame accelerations from on-board accelerometers
— Rotational speeds from on-board gyrometers

A naive approach to estimate the jerk of the position of a UAV would be to directly differentiate either the position signal three times or the acceleration signal once. The problem with the direct differentiation of these signals is that measurement noise quickly degrades the result to the point where the estimated jerk becomes unusable. A more realistic approach is thus needed in order to be able to apply control techniques that include feedback terms using jerk. Methods have been developed to estimate the jerk of an object using variations of a linear Kalman filter, but few publications validate these estimators using experimental data. To the best of our knowledge, none have used the estimate the jerk for the feedback control of a multirotor UAV.

1.3 Research Objectives and Organization of the Work

The first main objective of this work is to explore the results found in [1]. It would be interesting to evaluate the effectiveness of the proposed control algorithms with a small commercially available quadrotor. The aggressive trajectory generation framework should be developed in simulation and then validated with laboratory testing. The MRASL offers an ideal work space to test these methods with small UAVs in enclosed spaces.

The remainder of this work focuses on the development of a real-time estimator of the jerk of a quadrotor and a controller that makes use of it. In short, the main objectives of this ensuing section are :

— Develop an accurate state-estimator to estimate a multirotor UAV's linear jerk in real-time
— Develop a method to validate this estimator with experimental data using off-line techniques
— Apply the validated estimate to a controller that reduces the jerk of the UAV using optimal control techniques
— Validate the estimator and controller with laboratory testing

This work is organized as follows. Chapter 2 outlines the state of the art of quadrotor trajectory generation and control. It also presents possible solutions to the jerk-estimation problem as well as possible applications for the proposed estimator and controller. Chapter 3 presents a mathematical model of the quadrotor as well as the empirical constants necessary for control design. Chapter 4 describes the important elements of the test setup used for this work. Chapter 5 presents results from the reproduction of [1]. Chapter 6 outlines the development of a real-time jerk estimator and presents a method to validate such an estimator off-line. Chapter 7 shows the development of a jerk-augmented controller for a quadrotor UAV that uses optimal control techniques. Finally, Chapter 8 summarizes the results obtained in this work and suggests future projects and improvements.

CHAPTER 2 LITERATURE REVIEW

Research into quadrotor control has quickly become a large field encompassing many different specializations. This section aims to give an overview of the state of the field. Section 2.1 provides references to some basic control systems references as well as typical examples of how quadrotors can be modelled and controlled. As this work focuses on the performance of aggressive manoeuvres for UAVs, a thorough examination of quadrotor trajectory generation and control is presented in section 2.4. Some new and theoretical works on quadrotor control require feedback of the jerk in order to function. Section 2.4.1 presents an examination of the limited domain of real-time jerk estimation. Finally, examples of the potential application of jerk estimation are presented in section 2.6

2.1 General Control Theory and Quadrotor Modelling

2.2 Control System Fundamentals

Some well-established texts are available for an introduction to analog and digital control theory. Bishop and Dorf [8] give a good overview of basic analog control concepts for Single Input/Single Output (SISO) linear systems. Rugh [9] expands these techniques and applies them to multivariable linear systems. The techniques in these books rely on linear approximations of nonlinear systems. Khalil's *Nonlinear Systems* [10] provides a background in nonlinear control theory, wherein the full mathematical nature of a system is modeled and controlled. If a controller is to be used in real-world applications, it is often important to take into account the fact that control is often executed by microcontrollers or computer systems. Chen [11] offers a detailed description of control system design of digital systems, including theory on adapting controllers developed in the continuous domain to discrete applications.

2.3 Quadrotor Modelling and Control

The basis on which a control system is designed is its mathematical model. There are different types of quadrotor model that can be chosen depending on the desired area of application. The simplest of these is the result of a linearization about an equilibrium point. Examples of this type of linear model are used in [12] and [13]. Nguyen, Saussié and Saydy [12] develop a linear model for a quadrotor and use it with a fault-tolerant controller to improve performance in the event of actuator faults. Tran et al. [13] use a similar model and creates simple LQR (Linear

Quadratic Regulator) and PID (Proportional Integral Derivative) controllers to maintain a static position in the presence of wind.

Nonlinear models and methods can also be used in order to control quadrotor UAVs. Simulation results in [14] showed how to control a quadrotor directly from a nonlinear model using feedback-linearization techniques. The nonlinear model used did not take into account actuator effects and was based on Euler angles. Other nonlinear techniques such as back-stepping and sliding-mode control were tested experimentally in [15] with varying degrees of success.

Quadrotors have been used for a wide variety of complex tasks. They have been shown to effectively balance an inverted pendulum [5], tie knots and build bridges [16],[17] and use robotic manipulators [18] to name only a few.

2.4 Trajectory Generation and Control

A core problem in quadrotor control is the generation and execution of feasible trajectories. Because of their generally light-weight designs and relative mathematical simplicity, quadrotors have often been used to perform complex high-speed manoeuvres and tasks. The ability to generate and then converge onto a desired trajectory has been treated in many ways. Some of these applications are practical in nature and do not offer theoretical guarantees on the convergence onto a trajectory, while others give mathematical proof.

Linear approaches to trajectory tracking are possible and relatively easily implementable. A time-variant linear quadratic regulator (TVLQR) was developed in [19] in order to navigate through cluttered indoor environments. This thesis also proved useful because it used the same hardware that is available in the MRASL.

Other technical works have been produced using the Crazyflie 2.0 quadrotor. Luis and le Ny [2] adapt a Linear Quadratic Tracking (LQT) controller to function with the Crazyflie to track various simple trajectories. In the process, it gives detailed descriptions of the physical properties of the drone, as well as outlines the implementation of the controller for experimental tests. Hanna [20] develops a simple controller and implements it with an earlier model of the Crazyflie. Forster [21] provides a full identification of the Crazyflie 2.0 quadrotor.

Another simple way to define a quadrotor trajectory is through a technique called waypoint navigation. In this method, a trajectory is defined by a sequence of points with associated velocities. This technique is presented in [22] and validated experimentally for 2D trajectories. A version of this technique is then expanded to three dimensions and combined with sequential composition techniques to perform aggressive manoeuvres through narrow windows in [1]. This paper proved to be a very important one in the field of quadrotor trajectory generation, providing the basis for a number of other publications. However, its method depends on an iterative tuning technique that makes it impractical for real-world applications.

In order to eliminate the tuning phase of the aggressive manoeuvres, theoretical development based on the feasibility of trajectories needed to be done. This led to a focus on the property of differential flatness of a desired output. In short, an output of a system is termed differentially "flat" if it can be expressed only by its derivatives and system inputs. The consequence of an output being differentially flat is that it can be feasibly executed using the available input signals of a system.

An important example of this concept is applied in [23], where a quadrotor's position is demonstrated to be differentially flat considering propeller speeds and the fourth derivative of its position - or "snap". The authors then proceed to define feasible polynomial trajectories that numerically minimize the level of snap experienced by the UAV. This enables a quadrotor to fly quickly through static and moving hoops while respecting the desired trajectory.

Another application of differential flatness concepts is presented in [24]. The differential flatness of the quadrotor is used in order to effectively perch on inclined surfaces, as in [1], but this time without an iterative tuning phase. Its controller was based on results presented in [25], where a nonlinear controller based off of the special Euclidean group SE(3) is developed to avoid problems with Euler angles and quaternions.

The concept of differential flatness is also treated in [26]. The authors of this paper develop a controller that implements minimal snap trajectories and uses a nonlinear controller. This controller was based on dynamic inversion theory and performed feedback of acceleration measurements.

De Almeida [27] solves the aggressive flight through windows problem with an emphasis on the numerically stable nature of the trajectory solution. It addresses issues relating to the ill-conditioning issues of the quadratic programming problem that arises when optimising a trajectory for minimal snap.

Another approach to the problem of flying through narrow gaps was presented in [28]. It also made use of polynomial trajectories in order to plan a path through windows angled up to 45 degrees. It also does so without the need for external motion capture systems. Rather than use concepts of minimal snap, trajectories are generated to minimize the jerk of the overall trajectory. The selection process for these trajectories is executed using results presented in [29], which develops a computationally efficient way to produce trajectories that fall within a feasible range of input propeller forces.

Rather than use trajectories that minimize the snap of the trajectory, some works apply differential flatness to produce trajectories that minimize their jerk. Yu et al. [30] and Rakgowa [31] both use such an approach.

Other techniques such as model predictive control (MPC) have been used to execute trajectories. DeCrouzaz [32] proposes such a method with a Sequential Linear Quadratic (SLQ) controller and applied it to an AscTec Firefly hexacopter as well as a Rezero ball-balancing robot.

Another problem with trajectory tracking for quadrotors arises when it is carrying a dynamic payload. Tang [33], Taylor [34], Foehn [35] and Palunko [36] all treat the problem of generating manoeuvres with an attached swinging payload.

2.4.1 General State Estimation

The underlying assumption in all of the preceding papers is that there are elements of a quadrotor's state that are available in real-time for feedback. Most of these applications only require feedback of position, velocity, orientation and angular velocity. Some applications, such as [26] and [37] use feedback of acceleration as well. As a result, methods to acquire such measurements have been developed in detail.

The problem of combining data from multiple sensors to provide a full-state estimation is treated in [38]. This book is a good resource for robotics in general, but specifically for multi-sensor data fusion. Estimators presented in [39] provide a way to estimate a quadrotor's attitude based on gyrometer and accelerometer measurements. Extended Kalman Filters described in [40] allow full-state estimation from multiple sensors. [41] provided a way to estimate a quadrotor's position based on on-board camera data. [42] uses multiple refinements of a quadrotor's dynamic model to improve its real-time state estimate.

2.5 Estimation of Jerk

Possible solutions to the problem of real-time estimation of a vehicle's state have been proposed in [3], [43], [44] and [45]. A continuous linear Kalman and $\mathcal{H}_\infty$ filter were developed in [3] for the purpose of evaluating the jerk of an object in real time using only a noisy acceleration signal. However, its estimate either had significant delay, or substantial noise, that made it unsuitable for the purpose of control of a UAV. It contained a model for the accelerometer noise that included a coloured noise component, which could take into account high frequency vibrations from spinning propellers.

[43] developed an extended Kalman filter in order to track evasive flying targets using radar. They compared a state-estimation based on an acceleration model with an augmented estimator that includes the behaviour of the jerk. Simulations showed that including the jerk in their model improved the overall estimate of the state of the evasive target.

[45] produced an unscented Kalman filter (UKF) and an extended Kalman filter in order to generate a full state estimate for a flexible joint. The proposed nonlinear five bar linkage control scheme required a feedback of the jerk in order to be stable. Simulations showed that both the UKF and EKF had the capacity to track the jerk of the components of the five bar linkage.

Although there has been limited development in terms of jerk estimation in the field of robotics, multiple estimators have been proposed for automotive applications. An estimation of the jerk was also developed in [44] in order to evaluate the intention of a driver. They used a linear Kalman Filter (LKF) combined with a high-gain filter in order to estimate the jerk of a vehicle based on torque requests from a driver.

Another analysis of driver intention based on jerk was presented in [46]. A jerk model was used in [47] in order to improve performance of an automotive cruise control application ; a controller was then developed to effectively avoid collisions while limiting the maximum jerk experienced by the vehicle. A high-order nonlinear observer was proposed in [48] to track the state of a quadrotor up to jerk, but was only validated by simulation.

In terms of quadrotor applications, only one work was found that treated the estimation of jerk explicitly. An estimator produced in [49] used a Kalman filter with an augmented state vector that included the jerk and snap (referred to as jounce in this work). However, this was only used in order to increase the accuracy of position estimates in an outdoor environment. The article lacked a detailed analysis of the performance of the estimation of jerk and snap and did not use these estimates for feedback.

2.6 Potential Applications

A possible use for the estimation of jerk could come with the consideration of rider comfort for a large autonomous vehicle. Results presented in [50] showed a strong relation between the high levels of jerk caused by the longitudinal roughness of a road and a degradation of rider comfort.

Problems associated with high levels of jerk also occur when position estimation comes from on-board cameras. Motion-blur caused by high levels of rotation rate (which is directly related to linear jerk) caused significant enough blurring of camera data to force researchers in [41] to minimize the jerk of their calculated trajectories. Test results from a ground-based robot performing mapping operations in [51] also suffered from high levels of acceleration rate.

Applications of real-time estimates of the jerk have been produced in the field of nonlinear quadrotor control. However, because of difficulty of estimation, some works have actively tried to avoid doing so. A controller designed in [52] was produced specifically to eliminate the need for a feedback of the jerk.

An exact feedback linearization of the quadrotor results in the need for a feedback of the jerk, as seen in [53]. This work presents a nonlinear controller with an associated observer, but does not validate with experimental results.

The feedback of jerk appears again in [54]. This article develops a controller using feedback linearization techniques in order to guarantee the convergence of a quadrotor onto a 3-D trajectory. The controller, however, requires feedback of the quadrotor's jerk and acceleration in order to be stable. The paper did not attempt to estimate the jerk and only validated its contributions via simulation.

2.7 Conclusions on the State of the Art

The field of quadrotor trajectory generation and control has grown significantly in the last decade. A increase of interest in nonlinear control concepts in new works has led to more theoretically complex solutions to this problem. However, a lack of development in higher-order state estimation has prevented many of these applications from being validated experimentally. This absence of actual testing of controller configurations is a significant gap in the field. This is the main reason for the focus on testing and experimentation in this work.

CHAPTER 3 MATHEMATICAL MODEL OF A QUADROTOR UAV

In order to design a control system for a quadrotor UAV, a proper mathematical model must be defined. This chapter outlines a typical nonlinear model for a quadrotor and resulting first-order linearization. Various independent reduced models are then extracted from this linearization, which become practical during the controller design process. Empirical values for the Crazyflie 2.0 used in testing are then presented.

3.1 Nonlinear Model

This section introduces the nonlinear model of a quadrotor UAV as presented in [12]. A body-fixed frame $\{\mathsf{B}\}$ is defined at the center of mass of the quadrotor, with the z-axis pointing downwards and the x- and y-axis along the arms according to the so-called plus "+" configuration (Fig. 3.1). The "×" configuration corresponds to the case where the x- and y-axis are between the arms.

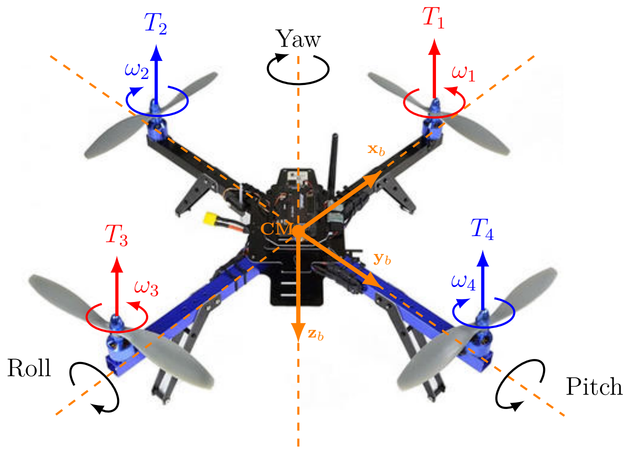

Figure 3.1 Quadrotor configuration

This frame is related to the inertial frame $\{\mathsf{N}\}$ by a position vector $\mathbf{p} = [x\ y\ z]^\top$ and three Euler angles $\mathbf{\Phi} = [\phi\ \theta\ \psi]^\top$, representing respectively roll, pitch and yaw. The rotation matrix resulting from a yaw-pitch-roll sequence is as follows :

$$\mathbf{R}_{\mathsf{B/N}} = \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{bmatrix} \tag{3.1}$$

where $c_x = \cos x$ and $s_x = \sin x$. The rotation matrix is orthogonal, thus $\mathbf{R}_{\mathsf{N/B}} = \mathbf{R}_{\mathsf{B/N}}^\top$. The angular velocity of frame $\{\mathsf{B}\}$ with respect to frame $\{\mathsf{N}\}$ expressed in frame $\{\mathsf{B}\}$ is denoted $\boldsymbol{\omega} = [p\ q\ r]^\top$ and the transformation matrix for angular velocities is $\mathbf{H}(\boldsymbol{\Phi})$. Therefore, one has :

$$\dot{\boldsymbol{\Phi}} = \mathbf{H}(\boldsymbol{\Phi})\boldsymbol{\omega},\ \mathbf{H}(\boldsymbol{\Phi}) = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi/c_\theta & c_\phi/c_\theta \end{bmatrix} \tag{3.2}$$

in which $t_x = \tan x$. Combined with Eq. 3.2, the equations of motion are written as :

$$m\dot{\mathbf{v}} = m\mathbf{g} + \mathbf{R}_{\mathsf{N/B}}\mathbf{F} \tag{3.3}$$

$$\mathbf{I}_{\mathsf{B}}\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times \mathbf{I}_{\mathsf{B}}\boldsymbol{\omega} + \mathbf{M} \tag{3.4}$$

where $\mathbf{v} = [v_x\ v_y\ v_z]^\top = \dot{\mathbf{p}}$, $\mathbf{F}$ and $\mathbf{M}$ denote the forces and torques created by the rotors in frame $\{\mathsf{B}\}$, $\mathbf{g} = [0\ 0\ g]^\top$ the gravity vector in frame $\{\mathsf{N}\}$, m the mass of the quadrotor and $\mathbf{I}_{\mathsf{B}}$ the inertia matrix about the center of mass. Because of the symmetric structure, the inertia matrix is assumed to be diagonal, i.e., $\mathbf{I}_{\mathsf{B}} = \mathrm{diag}\,(I_{xx}, I_{yy}, I_{zz})$ with $I_{xx} = I_{yy}$. Each rotor i creates thrust force T_i in the direction of $-\mathbf{z}_b$, producing forces and moments. The force and moment expressions are given by

$$\mathbf{F} = \begin{bmatrix} 0 \\ 0 \\ -f \end{bmatrix},\ \mathbf{M} = \begin{bmatrix} u_\phi \\ u_\theta \\ u_\psi \end{bmatrix} \tag{3.5}$$

where f represents the total thrust, u_ϕ and u_θ the rolling and pitching moments and u_ψ the yawing moment due to the reaction torques of the rotors.

Note that the equations 3.2, 3.3 and 3.4 are valid for both “+” and “×” configurations with the variables being defined accordingly to the axis systems in use.

3.2 Model linearization

The nonlinear model in Eqs. 3.2, 3.3, and 3.4 is trimmed and linearized by assuming hovering flight ($f = mg$, $u_\phi = u_\theta = u_\psi = 0$) with null yaw ($\psi = 0$). This yields the classical linearized equations :

$$\begin{aligned} \Delta\ddot{x} &= -g\Delta\theta & I_{xx}\Delta\ddot{\phi} &= \Delta u_\phi \\ \Delta\ddot{y} &= g\Delta\phi & I_{yy}\Delta\ddot{\theta} &= \Delta u_\theta \\ m\Delta\ddot{z} &= \Delta f & I_{zz}\Delta\ddot{\psi} &= \Delta u_\psi \end{aligned} \tag{3.6}$$

where Δ denotes the deviation of a variable from its equilibrium value.[1] A corresponding state-space model is then

$$\begin{cases} \Delta\dot{\mathbf{x}} = \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\Delta\mathbf{u} \\ \Delta\mathbf{y} = \mathbf{C}\Delta\mathbf{x} + \mathbf{D}\Delta\mathbf{u} \end{cases} \tag{3.7}$$

where the state vector $\Delta\mathbf{x}$, the input vector $\Delta\mathbf{u}$ and the output vector $\Delta\mathbf{y}$ are chosen as follows :

$$\Delta\mathbf{x} = \left[\Delta x\ \Delta y\ \Delta z\ \Delta v_x\ \Delta v_y\ \Delta v_z\ \Delta\phi\ \Delta\theta\ \Delta\psi\ \Delta p\ \Delta q\ \Delta r\right]^\top \tag{3.8}$$

$$\Delta\mathbf{u} = [\Delta f\ \Delta u_\phi\ \Delta u_\theta\ \Delta u_\psi]^\top \tag{3.9}$$

$$\Delta\mathbf{y} = [\Delta x\ \Delta y\ \Delta z\ \Delta\phi\ \Delta\theta\ \Delta\psi]^\top \tag{3.10}$$

The state-space matrices are consequently given by :

$$\mathbf{A} = \left[\begin{array}{cccc} \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \hdashline \mathbf{0}_3 & \mathbf{0}_3 & \begin{matrix} 0 & -g & 0 \\ g & 0 & 0 \\ 0 & 0 & 0 \end{matrix} & \mathbf{0}_3 \\ \hdashline \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \\ \hdashline \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{array}\right], \mathbf{B} = \left[\begin{array}{cccc} \multicolumn{4}{c}{\mathbf{0}_{3\times 4}} \\ \hdashline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1/m & 0 & 0 & 0 \\ \hdashline \multicolumn{4}{c}{\mathbf{0}_{3\times 4}} \\ \hdashline 0 & 1/I_{xx} & 0 & 0 \\ 0 & 0 & 1/I_{yy} & 0 \\ 0 & 0 & 0 & 1/I_{zz} \end{array}\right] \tag{3.11}$$

$$\mathbf{C} = \left[\begin{array}{cccc} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \hdashline \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \end{array}\right], \mathbf{D} = \mathbf{0}_{6\times 4} \tag{3.12}$$

Many of the system's input/output characteristics can be decoupled. For instance, referring

1. For most of the state variables, the equilibrium value is simply 0.

to equation 3.6, the state-space equivalent of the relation between the input thrust and the quadrotor's height can be expressed as :

$$\begin{bmatrix}\Delta\dot{z}\\ \Delta\dot{v}_z\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\Delta z\\ \Delta v_z\end{bmatrix} + \begin{bmatrix}0\\ -1/m\end{bmatrix}\Delta f \tag{3.13}$$

Which shows that, in terms of the linearized model, the height of the UAV is only dependent on the total thrust generated by the propellers. Similarly, lateral and longitudinal models can be extracted.

$$\begin{bmatrix}\Delta\dot{x}\\ \Delta\dot{v}_x\\ \Delta\dot{\theta}\\ \Delta\dot{q}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & -g & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta v_x\\ \Delta\theta\\ \Delta q\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ 1/I_{yy}\end{bmatrix}\Delta u_\theta \tag{3.14}$$

The lateral reduced model can also be expressed :

$$\begin{bmatrix}\Delta\dot{y}\\ \Delta\dot{v}_y\\ \Delta\dot{\phi}\\ \Delta\dot{p}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & g & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\Delta y\\ \Delta v_y\\ \Delta\phi\\ \Delta p\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ 1/I_{xx}\end{bmatrix}\Delta u_\phi \tag{3.15}$$

Finally the reduced model for yaw control is :

$$\begin{bmatrix}\Delta\dot{\psi}\\ \Delta\dot{r}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\Delta\psi\\ \Delta r\end{bmatrix} + \begin{bmatrix}0\\ 1/I_{zz}\end{bmatrix}\Delta u_\psi \tag{3.16}$$

3.3 Properties of the the Crazyflie 2.0

In this work, all flight tests were performed with the Crazyflie 2.0 quadrotor. For the purpose of these tests, mathematical constants inherent to the quadrotor were necessary. Most of these model parameters have already been described in [21, 19, 2, 20] as well as on internet sources. Other elements, such as the characteristics of the on-board accelerometers needed to be identified by testing. This section outlines a compilation of the information available on the Crazyflie 2.0.

3.3.1 Mass, Geometry and Inertia

The mass of the Crazyflie 2.0 with and without Vicon marker was measured with a scale in the MRASL. As for the drone's inertial properties, a rigorous identification of the Crazyflie's inertia matrix was conducted in [21]. These values can be found in table 3.1 :

Table 3.1 Crazyflie 2.0 Mass, Moment of Inertia and Arm Length

m (without Vicon Marker)	$0.029\,\mathrm{kg}$
m (with Vicon Marker)	$0.032\,\mathrm{kg}$
$I_{xx} = I_{yy}$	$6.410179 \cdot 10^{-6}\,\mathrm{kg.m^2}$
I_{zz}	$9.80228 \cdot 10^{-6}\,\mathrm{kg.m^2}$
d	$39.73 \cdot 10^{-3}\,\mathrm{m}$

The layout of the Crazyflie's propellers is in what is called the "×" configuration, meaning they are at 45 degree angle with respect to the body-frame axes. This can be seen in figure 3.2 :

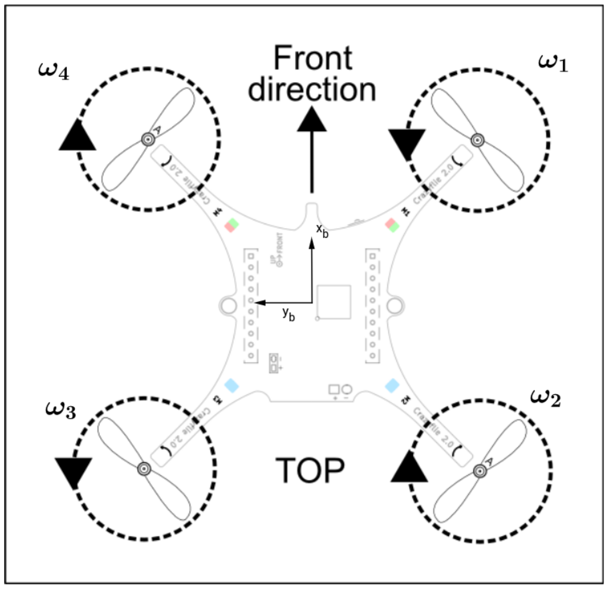

Figure 3.2 Crazyflie 2.0 "×" Mode

3.3.2 Motor Characteristics

The characteristics of the thrust and moments produced have been described in several ways. This subsection presents an overview of the thrust properties that have been proposed. One interesting set of behaviours is the complete relation between input signal and output thrust.

In terms of the Crazyflie 2.0, the overall desired thrust is commanded by setting the value of a PWM[2] register. This value can be between $0 - 65535$, corresponding to the range between 0 and maximum thrust. Because of this high resolution, it makes numerical sense to express this value as a fraction of the maximum PWM value. Test results presented on the Crazyflie's website[3] were adapted to express this relation, shown in figure 3.3 :

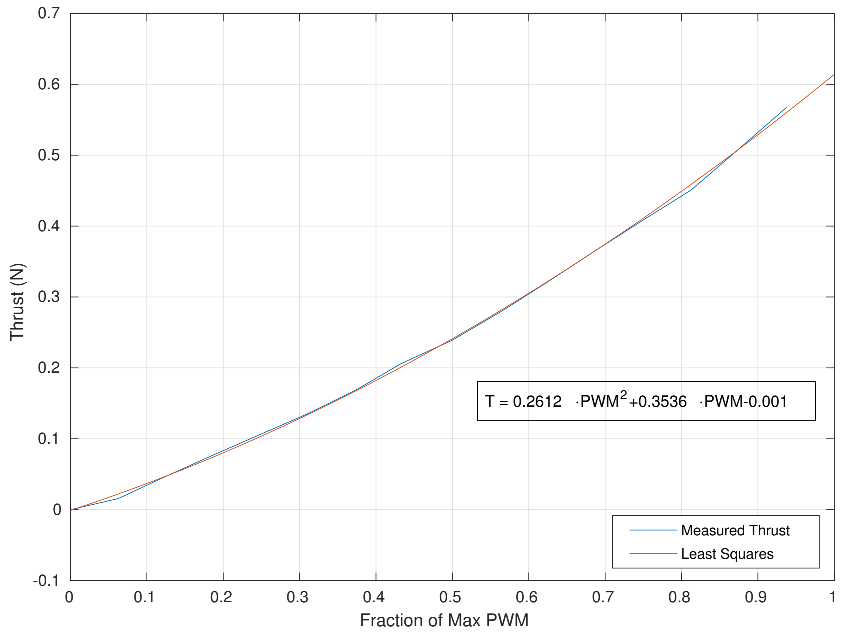

Figure 3.3 Total Thrust Produced as a Function of PWM Register Fraction

A second-order polynomial provided a good fit for this relation. Inverting it, the equivalent PWM value for a desired total force could then be calculated as in equation 3.17.

$$\text{PWM} = 65535 * \frac{-0.3536 + \sqrt{0.3536^2 + 4 \cdot 0.2612 \cdot (T + 0.001)}}{2 \cdot 0.2612} \tag{3.17}$$

The relation between PWM values and the resulting force coming from individual motors was also determined in [21].

In control theory, the forces and torques created by each propeller i is often modelled as being a function of the square of the angular velocity of the propeller :

2. Pulse Width Modulation
3. https ://wiki.bitcraze.io/misc :investigations :thrust

$$T_i = k_f \omega_i^2 \tag{3.18}$$

$$\tau_i = k_\tau \omega_i^2 \tag{3.19}$$

where k_f and k_τ are the force and moment coefficients, respectively. Theoretical approximations of the values of k_f and k_τ were shown in [2] and are compiled in table 3.2 :

Table 3.2 Crazyflie 2.0 Theoretical Motor Constants

k_f	$3.1582 \cdot 10^{-10}\,\mathrm{N/rpm}^2$
k_τ	$7.9379 \cdot 10^{-12}\,\mathrm{N/rpm}^2$

With these values and "×" configuration of figure 3.3, the total thrust and motor allocation can be expressed in terms of rotor velocities :

$$\begin{bmatrix} f \\ u_\phi \\ u_\theta \\ u_\psi \end{bmatrix} = \begin{bmatrix} k_f(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ \frac{dk_f}{\sqrt{2}}(-\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2) \\ \frac{dk_f}{\sqrt{2}}(-\omega_1^2 + \omega_2^2 + \omega_3^2 - \omega_4^2) \\ k_\tau(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{bmatrix} \tag{3.20}$$

A relation between PWM register value and propeller angular velocity was found in [21] :

$$\omega_i(\mathrm{rad/s}) = 0.04076521 \cdot \mathrm{PWM} + 380.8359 \tag{3.21}$$

It is important to note, however, that in real-time flight, significant firmware modifications to the Crazyflie 2.0 are needed to directly control the motors. Efforts to run a completely off-board controller that commanded rotor speeds directly were made in [19], but produced dissatisfactory results.

Finally, an empirical analysis of the z-axis torque produced by each motor was made. It was found to follow the following relation :

$$\tau_i(\mathrm{Nm}) = 5.96 \cdot 10^{-3} T_i + 1.563383 \cdot 10^{-5} \tag{3.22}$$

3.3.3 Motor Time Constant

The motor actuators were modeled to respect a first-order transfer function according to :

$$\frac{T(s)}{T_c(s)} = \frac{1}{\tau s + 1} \tag{3.23}$$

where T_c is the commanded force and τ was found to have a value of 45 ms after identification. Because the implemented Kalman filter uses commanded virtual snap commands, this relation improves the accuracy of real-time estimation of the Crazyflie's state.

3.3.4 Sensor Identification

For this work, the quadrotor's acceleration and position measurements were needed in order to complete the augmented-stated estimator presented in chapter 6. However these signals are not perfect and contain measurement noise. In order to effectively use these signals, an identification of their properties was necessary. Due to the high rate of rotation of the Crazyflie's propellers, measurements taken by the accelerometer were prone to high levels of measurement noise. In order to quantify this noise, a simple test was performed. The drone was fastened to a stable flat surface as seen in figure 3.4 and sent a constant PWM value of 15000 for 60 seconds. A gaussian distribution was then fit to the resulting data in order to obtain a reasonable value for the white-noise variances. The results can be seen in figure 3.5.

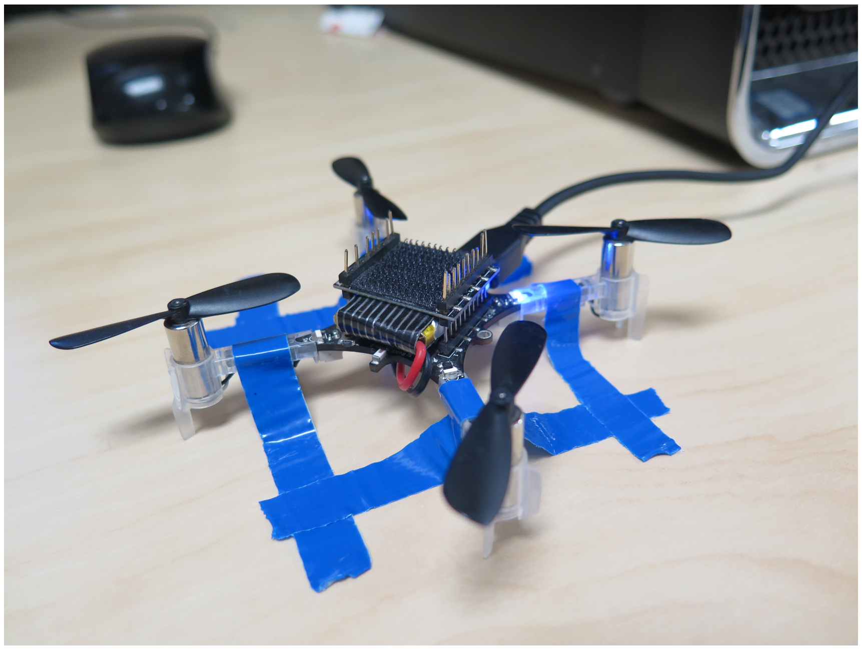

Figure 3.4 Test Bench for Accelerometer Variance Data

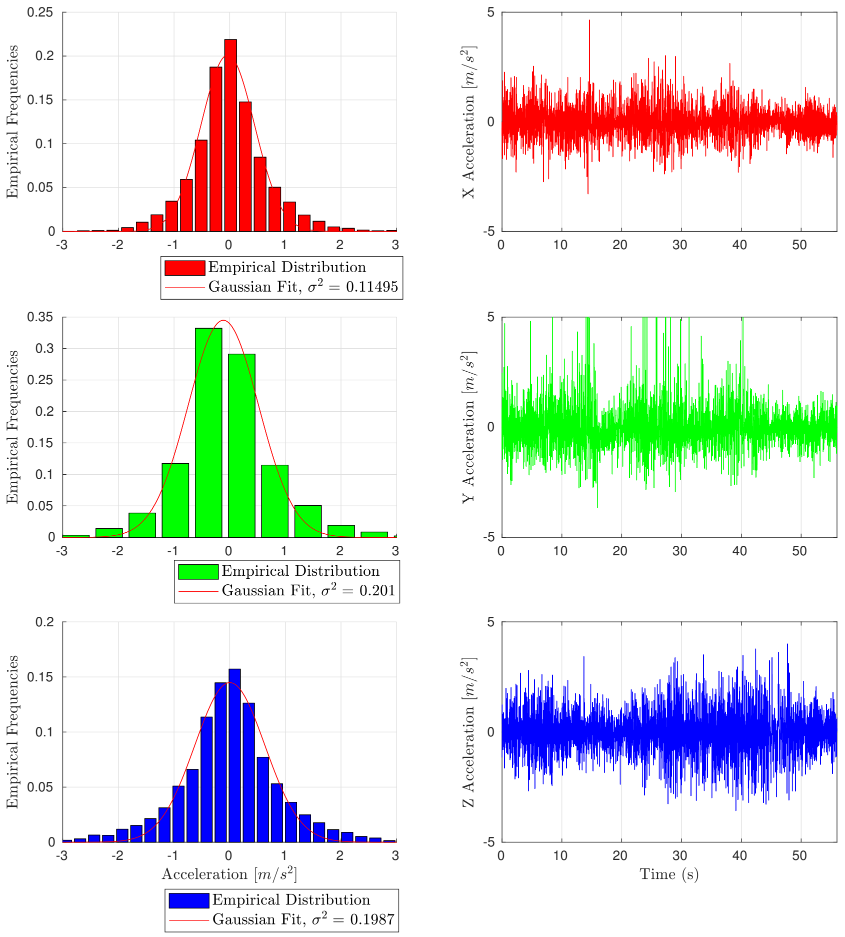

Figure 3.5 Signal Identification for Acceleration Measurements

A similar procedure was performed to determine the precision of the Vicon position measurements. A Vicon marker was placed at the volume origin of the flight arena and its position was recorded for 60 seconds. the resulting data can be seen in figure 3.6.

These tests revealed important information about the quality of the on-board and off-board measurements. Both displayed close to gaussian behaviour, validating the assumption that the involved noise is a uniform white noise. For the accelerometer, noise variance had a range between 0.11 in the x-axis direction to 0.21 in the y-axis direction. As for the position variances, the collected data demonstrated how precise the Vicon setup was. Variances ranged from $2.43 \cdot -9$ to $1.13 \cdot 10^{-8}$. Using these results, resulting conservative estimates of signal variances were created and compiled in table 3.3.

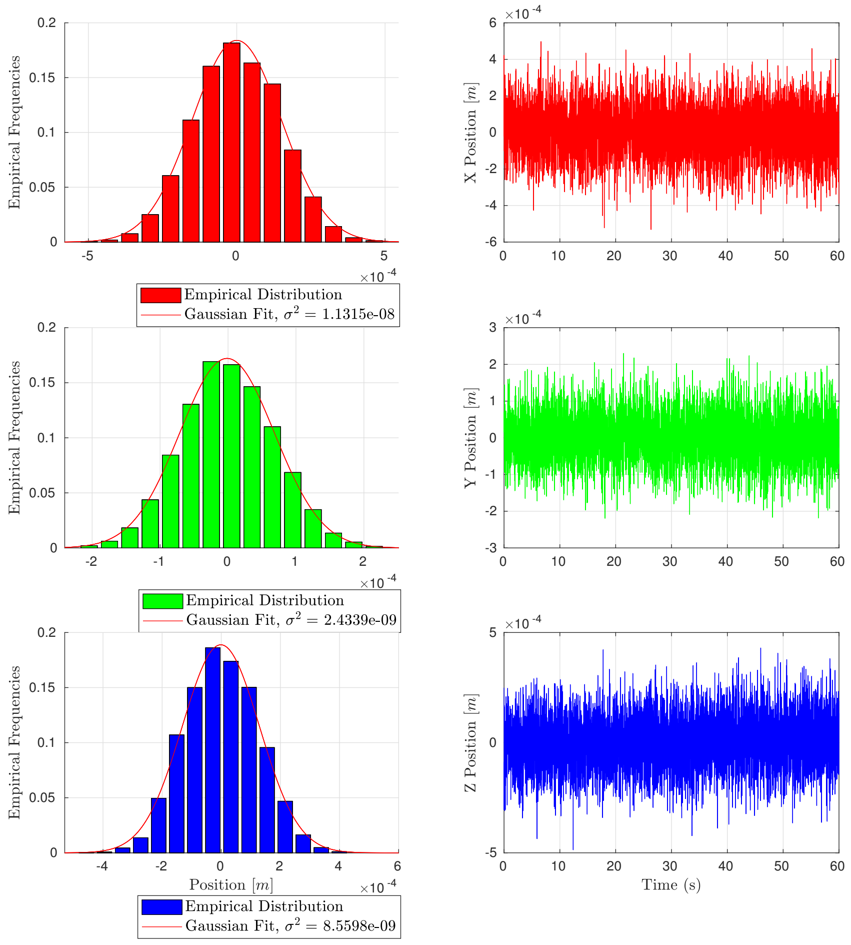

Figure 3.6 Vicon Signal Properties

Table 3.3 Crazyflie 2.0 White Noise Signal Variances

Measurement		Variance
Position	x	$1.5 \cdot 10^{-8}$
	y	$1.5 \cdot 10^{-8}$
	z	$1.5 \cdot 10^{-8}$
Acceleration	x	$3 \cdot 10^{-1}$
	y	$3 \cdot 10^{-1}$
	z	$3 \cdot 10^{-1}$

CHAPTER 4 EXPERIMENTAL SETUP FOR TESTING

Passing from simulation to experimental validation is a non-trivial step in the development of new quadrotor applications. This chapter aims to outline the necessary equipment and framework for the application of the control elements presented in the rest of this work. Section 4.1 outlines the specifics of ROS (Robotics Operating System), a necessary piece of software for communication between the related systems during testing. Section 4.2 presents the details of the Vicon motion capture system used for quadrotor positioning. Section 4.3 discusses the use of Matlab for off-board control and data collection. Technical details of the Crazyflie 2.0 are presented in section 4.4.

4.1 ROS

ROS is an open-source, meta-operating system commonly used in the field of robotics. It serves as the link between all of the necessary hardware used during testing. It is a complex tool, and a detailed description of ROS concepts can be found on their website [1]. However, there are a few fundamental elements of the ROS framework that are useful to know in order to understand how signals are sent between machines. Elements of note include :

— Nodes
— Topics
— Messages
— Master
— Packages

These elements are described in the following subsections.

4.1.1 Nodes

In the ROS network, a node is simply a process that performs computation. They can be in the form of MATLAB script, C++ code, Python code, etc. Nodes are connected to each other on the ROS network to form a graph. A typical ROS node can read (referred to as "subscribing") information from the network, execute a calculation and then publish an output to the network. The code related to the off-board control of the drone is an example of this. First, the control node must subscribe to position, velocity, orientation and angular rate data from the network. It then computes desired forces and orientations, which are then

1. http ://wiki.ros.org/ROS/Concepts

published to the ROS network. This information is accessed via ROS topics and is structured according to a desired message type.

4.1.2 Topics

A topic is is a named bus that allows nodes to transfer data to one another. Topics are defined on the ROS network, and have a specific structure that can contain multiple data streams. For instance, data from the on-board IMU of the Crazyflie 2.0 is received by an associated radio antenna connected to a computer on the ROS network. This data can be accessed in a terminal using the rostopic echo command :

In this example, a user is manually accessing data from the Crazyflie ROS node by subscribing to the Crazyflie topic and reading from IMU data organized in the form of a ROS message. Messages are discussed in the following subsection.

4.1.3 Messages

In ROS, a message is a data structure designed to enable the transfer of data between nodes. They are accessed by nodes via topics on the ROS network. Some examples of message types are :

- integer
- floating point
- boolean
- pose

Messages can store simple data, such as integer or boolean values, but can also be built to store sensor data from an IMU, or position and orientation values from a positioning system. Custom messages can also be built in order to store data structures that are not included in the default message types.

4.1.4 Master

The ROS master provides naming and registration services to the rest of the nodes in the ROS system. In other words, it allows the nodes in the ROS network to find and communicate with one another. Once nodes have established contact, they communicate directly with one another. In order to initiate the Master, the roscore command must be used :

4.1.5 Packages

In ROS, a package is a collection of files and folders that can contain nodes, libraries, datasets or any code that can be a useful independent collection of code. They are made to be easily reusable in different projects. For instance, in the case of this work, individual ROS packages were used to interface with Vicon, MATLAB and the Crazyflie 2.0 respectively. Packages can be created by manufacturers in order to enable interfacing with their products or users who want to build their own framework for robotics testing. Introductory tutorials on how to build packages and start simple ROS projects can be found on the dedicated ROS website [2].

4.1.6 Overview of ROS Network During Testing

The overall architecture of the ROS network for the tests can be seen as in figure 4.1. The controller node receives data from Vicon, Crazyflie, a joystick and a user-written "goal" node. This node interfaces with MATLAB and makes communication with the off-board controller simpler. The joystick node is used simply as a safety feature, and only sends whether a button is being pushed in order to allow the continuation of testing.

4.2 Vicon

In order to quickly and precisely measure the position of the quadrotor during testing, the MRASL relies on a Vicon motion-capture system. The system consists of a collection of 12 specialized cameras set up around the perimeter of the MRASL's flight arena. Each of these cameras tracks a desired number of Vicon markers within its field of view (Figs 4.2 and 4.3).

Combining data from all 12 cameras, high-frequency (100 Hz) estimates of position can be made to a precision of less than 1 mm. This makes Vicon a useful tool when trying to validate new types of control systems that require feedback of position and velocity.

Vicon also gives the option to track the orientation of objects in real-time. If multiple Vicon markers are attached to a rigid object, Vicon can use their displacements to estimate its rotation in 3D space. However, because of the small size of the quadrotor used in testing and the configuration of its firmware, only one marker was used and only its position was read explicitly by Vicon. Orientation estimates were read directly from inertial measurements on-board the Crazyflie.

2. http ://wiki.ros.org/ROS/Tutorials

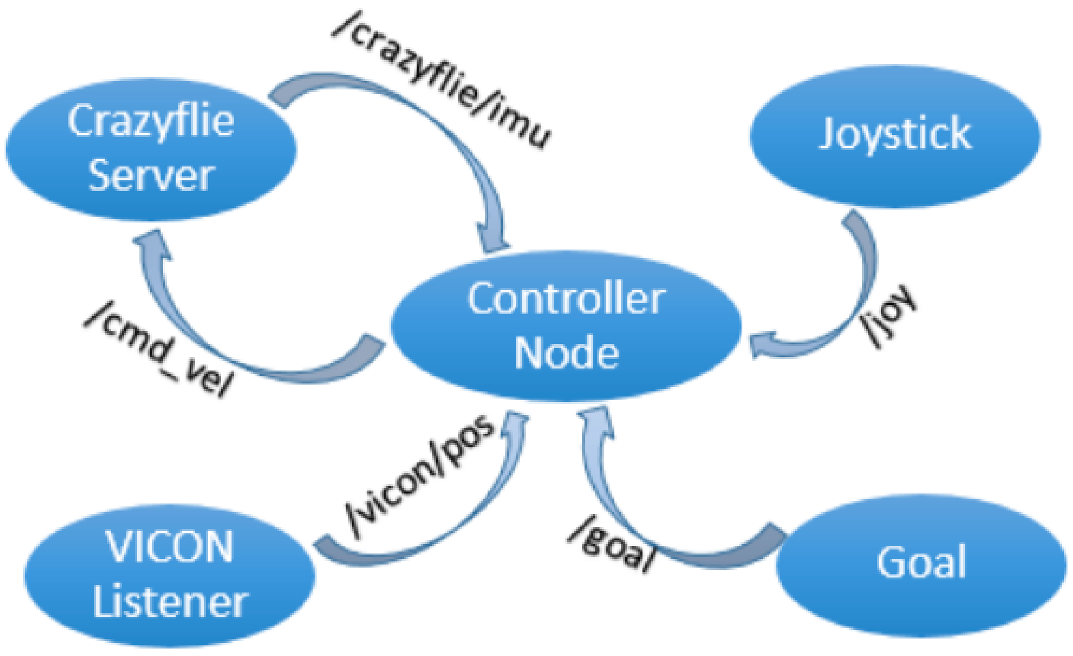

Figure 4.1 ROS Node Architecture, from [2]

Figure 4.2 MRASL Flight Arena

Figure 4.3 Vicon Camera Used during Testing

4.3 MATLAB

For engineering researchers, MATLAB is an indispensable tool for the simulation of new concepts. In the scope of this work, it was also useful for the implementation of controllers during testing. In order to facilitate the experiments performed, MATLAB/Simulink was used with the following add-on packages :

- ROS Package
- Real-Time Pacer
- Stateflow

4.3.1 ROS Package

Rather than create ROS nodes in compiled C++ or Python code, it can be practical to develop control nodes directly in Simulink. This makes changes to control structure as well as the visualization of data considerably easier for someone who is well-versed in the environment. In order to have MATLAB communicate as a node in the ROS network, the ROS add-on package must be used. After setup, this package offers blocks that can subscribe to and publish from topics on the ROS network. In the case of this work, blocks from the ROS package were used to subscribe to topics providing Vicon position data and Crazyflie sensor data. Based off of this data, control inputs could then be calculated and published to the

ROS network.

4.3.2 Real-Time Pacer

For the estimation modules used in this project, it was important for each time-step to run at a fixed frequency. By default, Simulink is intended for computer simulation and is thus designed to run as quickly as possible. This means that without management, while the state estimators used were expecting data at a frequency of 100 Hz, the actual control loop could be running much more quickly. Using the Simulink real-time package helped to solve this issue. The precision requirements for each time-step were relatively coarse (0.001 s), so the solution did not require the use of an actual real-time operating system (RTOS). Instead, the real-time package compared the simulation time with the clock of the PC being used and delayed the next time-step until the total time elapsed was 0.01 seconds. After this delay, the simulation clock was allowed to advance and new measurements, state estimates and control inputs could be calculated.

The package used was from a third party and can be found online[3].

4.3.3 Stateflow

In the case of testing, there can be multiple modes of operation and controllers. For example, a single test run for an aggressive trajectory as presented in chapter 5 begins with the motors not spinning for several seconds. The drone is then commanded to track a desired setpoint in 3D space and hover. After the drone has attained this point and reached null velocity, the drone then switches from a hover controller to a trajectory tracking controller with different architecture, followed by attitude control and finally hover control to finish the manoeuvre. Each of these controller switches has a different set of conditions in order to occur. Using conventional logic gates or user-defined functions in Simulink is possible, but inevitably creates cumbersome block diagrams that are not intuitive to read. The Simulink stateflow package enables much more elegant development of mode-switching algorithms and logic operations. More information can be found on MATLAB's website[4].

4.4 Crazyflie 2.0 Quadrotor

The drone used for testing in this work was the Crazyflie 2.0 nanoquadrotor. It is an open-source, programmable UAV that can be safely used indoors. For academic applications, it

3. http ://freesourcecode.net/matlabprojects/66903/real-time-pacer-for-simulink
4. http ://www.mathworks.com/help/stateflow/getting-started.html

has a radio antenna called the Crazyradio PA that can plug directly into a USB port. This enables an off-board PC to send command signals to the drone at a rate of up to 100 Hz. The Crazyflie comes with a pre-programmed on-board attitude controller with the structure shown in figure 4.4 :

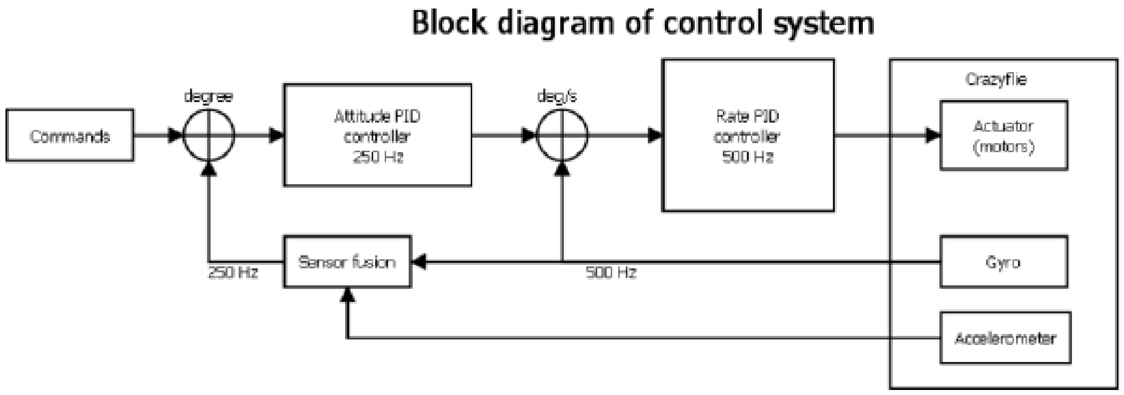
Block diagram of control system
Commands
degree
Attitude PID controller 250 Hz
deg/s
Rate PID controller 500 Hz
Crazyflie
Actuator (motors)
Sensor fusion
250 Hz
500 Hz
Gyro
Accelerometer

Figure 4.4 Crazyflie 2.0 Attitude Controller [5]

With the commands sent to the Crazyflie 2.0 being :

— Desired roll angle ϕ (deg)
— Desired pitch angle θ (deg)
— Desired yaw rate r (deg/sec)
— Desired overall thrust, expressed as a value between 0-65535

The on-board attitude controller receives these signals and executes 2 internal control loops. The outermost loop receives the desired roll and pitch angles and processes them at 250 Hz in a PID controller. It generates desired angular rates and sends them to an inner rate loop. This loop executes another PID controller at 500 Hz and controls for desired angular rates p, q, r. An off-board component of the attitude controller was added to the stock controller architecture in order to maintain constant yaw angle ψ throughout testing. This consisted of a PI controller that executed at 100 Hz.

The output of the on-board attitude controller is then combined with the desired overall thrust in order to calculate the desired rotor speeds :

$$\begin{bmatrix} \omega_{1,des} \\ \omega_{2,des} \\ \omega_{3,des} \\ \omega_{4,des} \end{bmatrix} = \begin{bmatrix} 1 & -1/2 & -1/2 & -1 \\ 1 & -1/2 & 1/2 & 1 \\ 1 & 1/2 & 1/2 & -1 \\ 1 & 1/2 & -1/2 & 1 \end{bmatrix} \begin{bmatrix} \omega_e + \Delta_T \\ \Delta_\phi \\ \Delta_\theta \\ \Delta_\psi \end{bmatrix} \tag{4.1}$$

where ω_e is the PWM value necessary for a stable hover, Δ_T is the desired deviation from the ω_e and $\Delta_\phi, \Delta_\theta$ and Δ_ψ are the outputs of the PID rate-loop.

The overall organization of communication between devices in testing can be seen in figure 4.5. On-board sensor data along with Vicon position data are simultaneously sent to an off-board PC with MATLAB running. Using this information, desired angles and thrusts are calculated and sent back to the CF2 via the Crazyradio PA radio antenna.

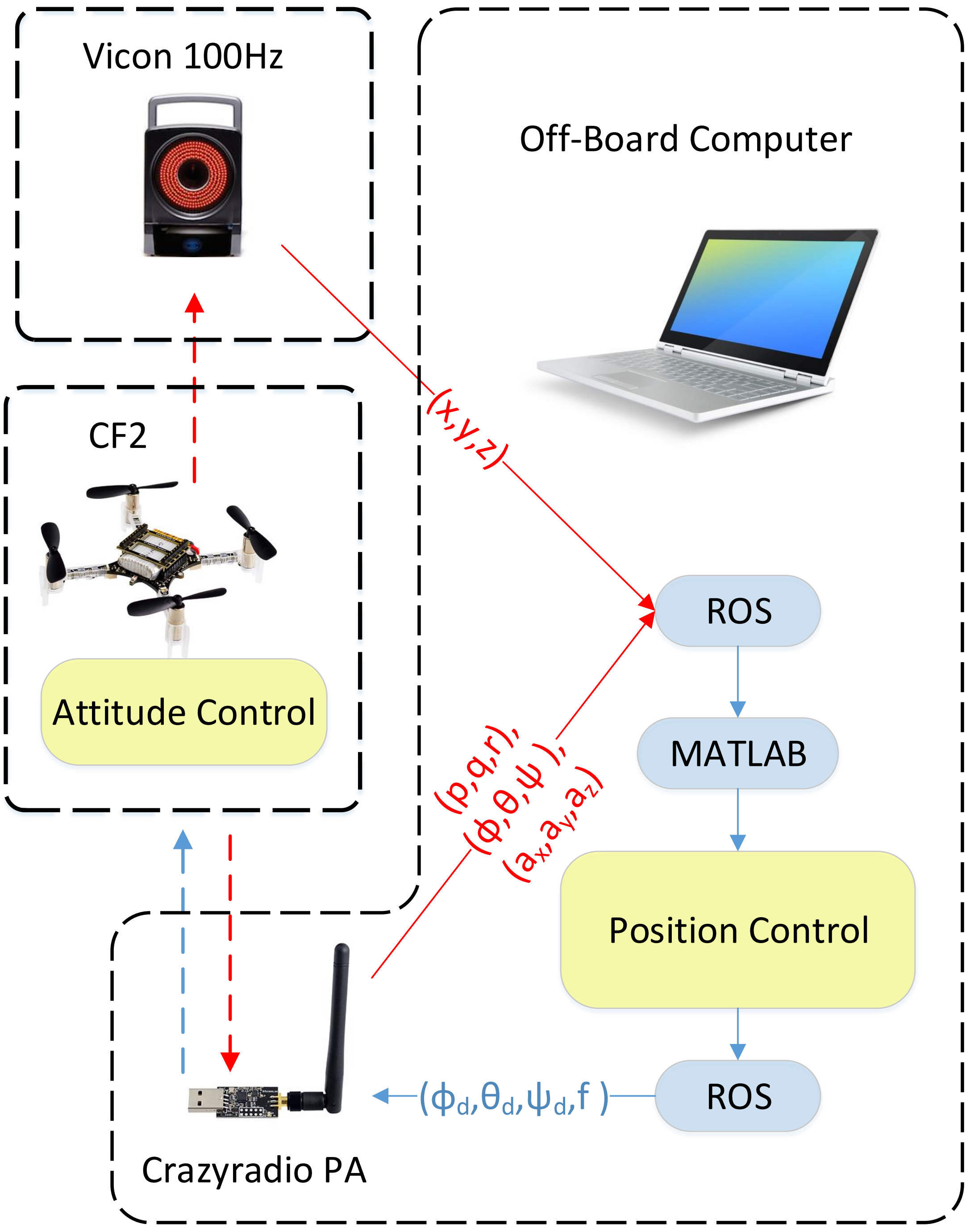

Figure 4.5 Experimental Setup

CHAPTER 5 AGGRESSIVE TRAJECTORY GENERATION

The development and execution of complex maneuvers for quadrotors has been the topic of many research papers in recent years. The ability to pass through narrow passageways at high velocities has been demonstrated with medium-sized quadrotors optimized for research purposes, notably by Mellinger, Michael and Kumar in [1]. This paper used a relatively simple approach to generate such trajectories involving the sequencing of several simple controllers to obtain a desired result. Their results showed that with a relatively simple control architecture, many possible types of complex maneuvers are possible. The technique of sequential composition was used in order to perform the passage of a quadrotor through narrow windows at varying angles as well as perching on inclined surfaces.

This chapter aims to validate such a control architecture by replicating their results in the MRASL's flight arena at Polytechnique Montreal. Section 5.1 outlines the various controllers that were proposed in [1] and briefly discusses how they are implemented in testing. Section 5.2 describes how these controllers are sequenced in order to generate desired trajectories through windows. Finally, section 5.3 outlines the results of testing.

5.1 Control Design

The control architecture of the project consists of 3 main types of controller :

— an on-board attitude controller ;

— an off-board hover controller ;

— an off-board 3D trajectory controller.

The "on-board" controllers are implemented in the firmware of the Crazyflie 2.0 and ran at either 250 Hz or 500 Hz on the device. The "off-board" controllers are managed in MATLAB/Simulink and executed at 100 Hz over the Crazyflie's Crazyradio FM transmitter antenna. In order to achieve the desired maneuvers, these controllers are sequenced in a specific order. A detailed description of each controller follows.

5.1.1 Attitude Controller

The goal of the attitude controller controller used in [1] was to achieve a desired angle in a given settling time. It was a proportional-derivative controller with the following form :

$$\begin{bmatrix} \omega_{1,des} \\ \omega_{2,des} \\ \omega_{3,des} \\ \omega_{4,des} \end{bmatrix} = \begin{bmatrix} 1 & -1/2 & -1/2 & -1 \\ 1 & -1/2 & 1/2 & 1 \\ 1 & 1/2 & 1/2 & -1 \\ 1 & 1/2 & -1/2 & 1 \end{bmatrix} \begin{bmatrix} \omega_e + \Delta_T \\ \Delta_\phi \\ \Delta_\theta \\ \Delta_\psi \end{bmatrix} \tag{5.1}$$

$$\Delta_\phi = k_{p,\phi}(\phi_{des} - \phi) + k_{d,\phi}(p_{des} - p) \tag{5.2}$$

$$\Delta_\theta = k_{p,\theta}(\theta_{des} - \theta) + k_{d,\theta}(q_{des} - q) \tag{5.3}$$

$$\Delta_\psi = k_{p,\psi}(\psi_{des} - \psi) + k_{d,\psi}(r_{des} - r) \tag{5.4}$$

where the gains $k_{\{p,d\},\{\phi,\theta,\psi\}}$ of the controller were tuned to produce a desired time-response. As discussed in section 4.4, the on-board attitude controller of the Crazyflie 2.0 has a similar structure. It is also augmented with integral components. This controller was tuned to have a 2% step response of time of 0.4 seconds. Validation of this controller is shown in section 5.3.

5.1.2 Hover Controller

The off-board hover controller's goal is to attain a specified position in 3-D space within a given settling time. In order to be able to methodically find desired gains, the structure of the hover controller is slightly different from the one presented in [1] :

$$\ddot{\mathbf{r}}_{i,des} = -k_{p,i}\mathbf{r}_i + k_{i,i}\int(\mathbf{r}_{i,des} - \mathbf{r}_i)\mathrm{d}t - k_{d,i}\dot{\mathbf{r}}_i \tag{5.5}$$

With $\mathbf{r}_{i,des}$ denoting the quadrotor's x, y, and z desirced position in the world-frame and $\mathbf{r}_i$ the actual position. Note that this controller configuration is essentially a state-feedback controller with integral action. Considering the non-linear system described in equation 3.3, and adding the attitude controller presented in section 5.1.1, we can produce a linearized system with the desired angles and total thrust as inputs and $\mathbf{r}$, $\dot{\mathbf{r}}$ as outputs. Using eigenstructure assignment techniques with output feedback on the measured positions and velocities, gains $k_{p,i}$, $k_{i,i}$ and $k_{d,i}$ can be found numerically. For certain maneuvers, it would be desirable to operate with an arbitrary yaw angle ψ. Linearizing (3.3), we can express the desired accelerations as a function of the desired roll, pitch and yaw angles :

$$\ddot{\mathbf{r}}_{1,des} = g(\theta_{des}\cos(\psi_{des}) + \phi_{des}\sin(\psi_{des})) \tag{5.6}$$

$$\ddot{\mathbf{r}}_{2,des} = g(\theta_{des}\sin(\psi_{des}) - \phi_{des}\cos(\psi_{des})) \tag{5.7}$$

$$\ddot{\mathbf{r}}_{3,des} = \frac{8k_F\omega_e}{m}\Delta_T \tag{5.8}$$

inverting these functions gives :

$$\phi_{des} = \frac{1}{g}(\ddot{\mathbf{r}}_{1,des}\sin(\psi_{des}) - \ddot{\mathbf{r}}_{2,des}\cos(\psi_{des})) \tag{5.9}$$

$$\theta_{des} = \frac{1}{g}(\ddot{\mathbf{r}}_{1,des}\cos(\psi_{des}) + \ddot{\mathbf{r}}_{2,des}\sin(\psi_{des})) \tag{5.10}$$

$$\Delta_T = \frac{m}{8k_F\omega_e}\ddot{\mathbf{r}}_{3,des} \tag{5.11}$$

Depending on the phase of the maneuver, the gains of this controller are modified to fulfill specific requirements. In the first phase of each maneuver, it is important to be able to maintain a precise position in space. The first set of gains is thus designed to be the "stiffer" of the two sets, and is tuned to have a settling time of 5 seconds with an overdamped response. The hover controller is also used in the recovery stage. In this phase, the goal of the controller is just to ensure the stability of the quadrotor given potentially significant initial conditions. The second set of gains is designed to be stiff enough to prevent the quadrotor from touching the ground during recovery but not so stiff as to saturate the motors and destabilise the system.

5.1.3 3D path following

The goal of the 3D path following controller is to follow defined trajectories in 3D space. The trajectories are defined by a set of points with associated velocities. In the case of this paper, these trajectories are limited to simple line segments between points $\mathbf{r}_T(i)$ with assigned yaw angles $\psi_T(i)$ (Fig. 5.1). At each moment, the controller evaluates which point $\mathbf{r}_T(i)$ on the trajectory is closest to the quadrotor and executes a specific control algorithm. In order to do this, we define $\hat{\mathbf{t}}$, the unit tangent vector of the trajectory associated with $\mathbf{r}_T(i)$ and desired velocity vector $\dot{\mathbf{r}}_T(i)$.

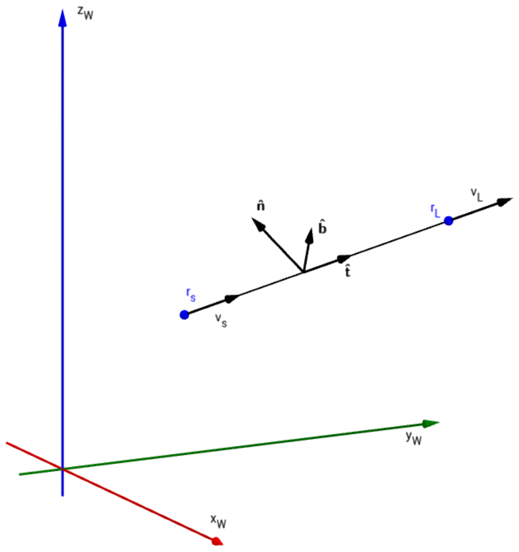

Figure 5.1 Line Segment Trajectory

The position error is defined as :

$$\mathbf{e}_p = ((\mathbf{r}_T(i) - \mathbf{r}) \cdot \hat{\mathbf{n}})\hat{\mathbf{n}} + ((\mathbf{r}_T(i) - \mathbf{r}) \cdot \hat{\mathbf{b}})\hat{\mathbf{b}} \tag{5.12}$$

where $\hat{\mathbf{n}}$ is chosen to be a unit vector orthogonal to $\hat{\mathbf{t}}$, and $\hat{\mathbf{b}}$ is chosen orthogonal to $\hat{\mathbf{n}}$ and $\hat{\mathbf{t}}$. Here, only the normal and binormal error is considered. This ensures that the position controller will only force the quadrotor onto the line segment trajectory and not cause it to “catch up” to the next point. The velocity error is defined as :

$$\mathbf{e}_v = \dot{\mathbf{r}}_T(i) - \dot{\mathbf{r}} \tag{5.13}$$

The desired accelerations of the Crazyflie 2.0 can then be calculated according to the following control law :

$$\ddot{\mathbf{r}}_{i,des} = k_{p,i}\mathbf{e}_{i,p} + k_{d,i}\mathbf{e}_{i,v} + \ddot{\mathbf{r}}_{i,T}(i) \tag{5.14}$$

where $\ddot{\mathbf{r}}_{i,T}(i)$ corresponds to optional feed-forward acceleration elements of the trajectory. While potentially useful in situations where the trajectory involves high accelerations, this component is set to 0 for the tests that were performed. The resulting desired overall thrusts and Euler angles can be calculated as in equations 5.9 through 5.11

5.2 Trajectory Generation for Window Maneuvers

Using a combination of the controllers presented in section 5.1, various complex maneuvers can be generated. This paper focuses on three of the maneuvers developed in [1] :

— flight through a 60 degree inclined vertical window
— flight through a 90 degree inclined vertical window
— descent through a horizontal window

Each of these maneuvers follows a specific sequence of controllers (Fig. 5.2).

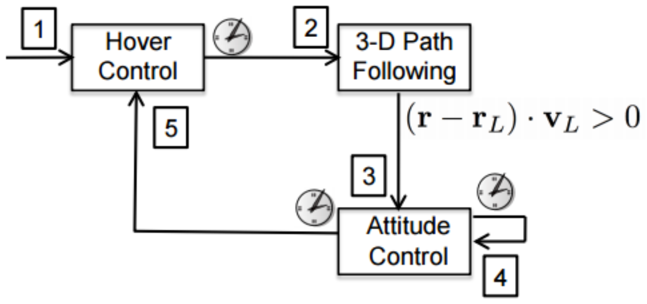

Figure 5.2 Maneuver Sequence (taken from [1])

Each sequence's goal is to reach a goal state G with specified position $\mathbf{r}_G$ velocity $\mathbf{v}_G$, yaw angle ψ_G and roll angle ϕ_G with zero angular velocity and pitch angle. We use the same sequence as in [1] :

1. Hover control with stiff gains to a start position $\mathbf{r}_s$
2. 3D path following to a desired position $\mathbf{r}_L$ with desired velocity $\mathbf{v}_L$ and yaw angle ψ_{des}
3. Attitude control to desired roll angle ϕ_{des} and yaw angle ψ_{des} and zero pitch angle θ
4. Attitude control to zero roll and pitch angles ϕ, θ and desired yaw angle ψ_{des}
5. Hover control with soft gains to a final position

5.2.1 Initial Parameter Selection

A first choice for the values of $\mathbf{r}_L$, $\mathbf{v}_L$ and $\mathbf{r}_s$ can be made by working backwards from the goal state G. Assuming that the roll angle behaves as a critically damped second-order system with settling time T_s while maintaining a constant overall thrust $\alpha \cdot mg$, we can estimate the sum of the forces on the system and integrate backwards to find $\mathbf{v}_L$ and $\mathbf{r}_L$. $\mathbf{r}_s$ can then be found using :

$$\mathbf{r}_s = \mathbf{r}_L - l\frac{\mathbf{v}_L}{\|\mathbf{v}_L\|} \tag{5.15}$$

where l is the overall length of the line segment trajectory chosen for the 3D path following controller.

5.2.2 Parameter Adaptation

In order to render the maneuver more accurate, the parameters chosen in the previous section are iteratively modified during testing. The maneuver is first refined for k trials in order to ensure that at the end of the first attitude phase (phase 3) the roll angle is within 2% of the desired angle. The desired roll angle ϕ is tuned accoring to the following algorithm :

$$\phi_C^{k+1} = \phi_C^k + \gamma_\phi(\phi_G - \phi_{act}^k) \tag{5.16}$$

where γ_ϕ is a constant chosen to be between 0 and 1. After the desired attitude behaviour is achieved, we turn to the velocity at the end of phase 3. For each type of maneuver, we control for a desired velocity as the Crazyflie passes through the window. The velocity is tuned in a similar fashion to the desired roll :

$$\mathbf{v}_C^{k+1} = \mathbf{v}_C^k + \gamma_v(\mathbf{v}_G - \mathbf{v}_{act}^k) \tag{5.17}$$

where γ_v is an adaptation constant chosen between 0 and 1. We then run the maneuver 15 times in order to observe the mean position $\bar{\mathbf{r}}_{act}$ at the end of phase 3. Finally, the entire trajectory is shifted in order to force it through the desired position of the window according to the following rule :

$$\mathbf{r}_L = \mathbf{r}_L + (\mathbf{r}_G - \bar{\mathbf{r}}_{act}) \tag{5.18}$$

$$\mathbf{r}_s = \mathbf{r}_s + (\mathbf{r}_G - \bar{\mathbf{r}}_{act}) \tag{5.19}$$

Note that because of the simplified nature of the model used as well as the “open-loop” behaviour of the quadrotor between phases 3 and 4, this adaptation procedure is an unfortunate yet necessary step in order to ensure the accuracy of the aggressive maneuvers.

5.3 Test Results

5.3.1 Attitude Controller

We first analyse the response of the roll angle during phase 3 to verify that it satisfies the behaviour hypothesized in section 5.2.1. We generate a vertical velocity and command a desired angle of 90 degrees for 0.6 seconds :

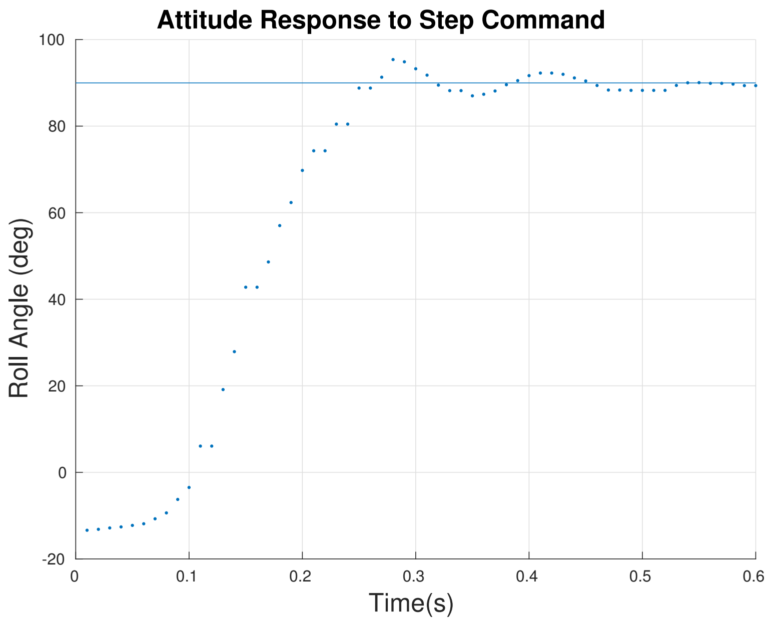

Figure 5.3 Attitude Response

We see that the on-board controller provided with the Crazyflie 2.0 provides a somewhat underdamped response to a large step command in ϕ_{des}. Its 2% settling time is very close to 4 seconds however, and is acceptable for the purposes of the considered maneuvers.

5.3.2 Vertical Window

5.3.2.1 60 Degrees

The goal state G for the case of the 60 degree vertical window was :

— $\mathbf{r}_G = \begin{bmatrix}0 & 0 & 1.75\end{bmatrix}^\top$ m
— $\mathbf{v}_G = \begin{bmatrix}0 & 2 & 0\end{bmatrix}^\top$ m/s
— $\phi_G = \pi/3\,\text{rad}$
— $\psi_G = \pi/2\,\text{rad}$

We maintain a constant overall thrust of $1 \cdot mg$ throughout the attitude control phases (3 and 4). After 1 iteration, the desired roll angle of the quadrotor remained within 2 degrees of the goal value ϕ_G. The next step was to iterate for the desired velocity at the end of phase 3. The maneuver converged to a satisfactory value after 4 iterations of the algorithm presented in equation 5.17 (Fig. 5.4).

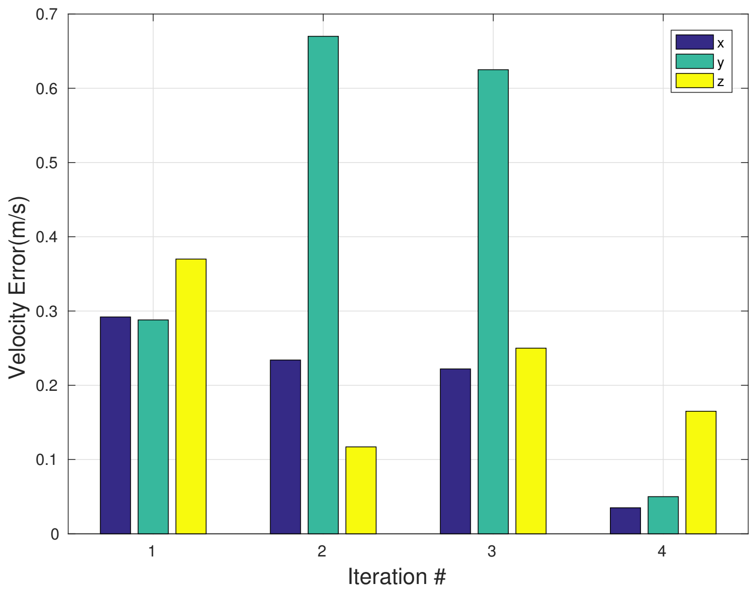

Figure 5.4 Velocity Improvement for 60 degree window

Figure 5.5 shows deviations from the mean position $\bar{\mathbf{r}}_{act}$ obtained after 15 trials of the sixty degree maneuver. We see that the maneuver is precise to within a range of less than 10 cm in the x and z directions. The standard deviations of the final positions through the 60 degree vertical window were 0.0423 and 0.0308 in the x and z directions respectively. The mean x-z position through the window's plane was found to be $\begin{bmatrix}-0.288 & 1.602\end{bmatrix}$ m. The final goal state was shifted by the deviation of the mean position from the desired position.

The maneuver was then tested with the window in place. A visualization of a representative test-run can be seen in figure 5.6. Each snapshot of the Crazyflie in the above graphic corresponds to a 0.1 second increment in experiment time. Here we see the quadrotor initially at rest. It then accumulates speed and follows an automatically generated trajectory until it crosses the plane defined by $\mathbf{r}_L$ and $\mathbf{v}_L$. The quadrotor then controls for a 60 degree roll angle for 0.4 seconds until it reaches the plane of the window. The desired attitude is then reset to 0 for roll and pitch angles for 0.4 seconds. It finally implements the hover controller in order to achieve its final stationary position.

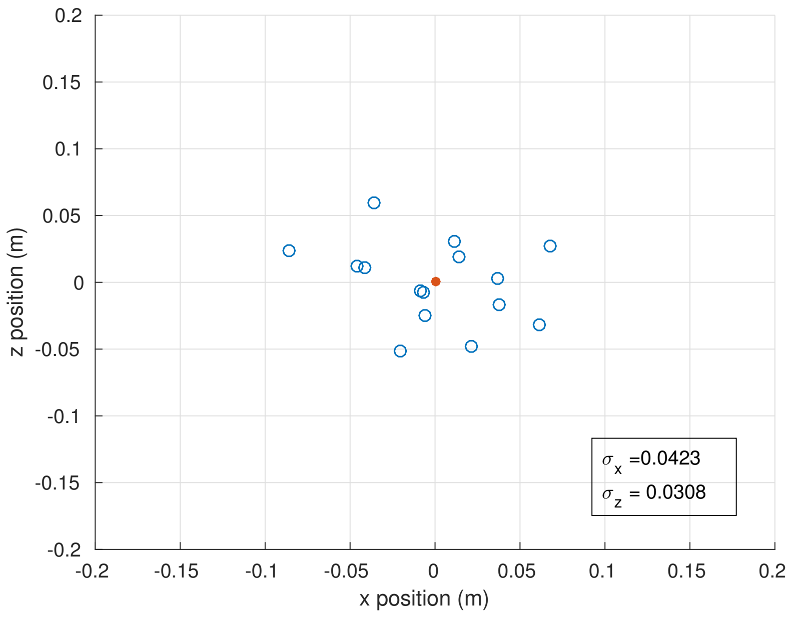

Figure 5.5 Precision of 60 Degree Vertical Window Maneuver, $\alpha = 1.0$

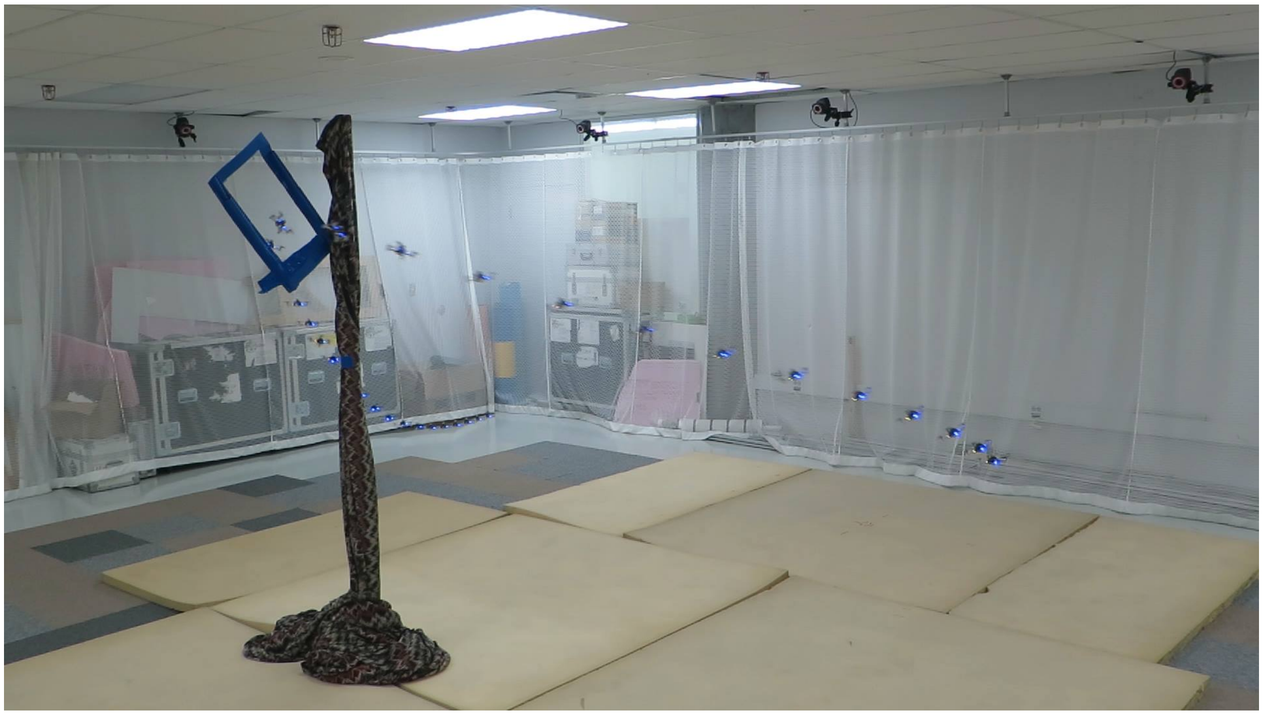

Figure 5.6 60 Degree Vertical Window Maneuver

5.3.2.2 90 Degrees

We perform the same procedure for a vertical window at 90 degrees with the following goal state :

- $\mathbf{r}_G = \begin{bmatrix}0 & 0 & 1.75\end{bmatrix}^\top$ m
- $\mathbf{v}_G = \begin{bmatrix}0 & 2 & 0\end{bmatrix}^\top$ m/s
- $\phi_G = \pi/2\,\mathrm{rad}$
- $\psi_G = \pi/2\,\mathrm{rad}$

The velocity adaptation phase showed a slower convergence to the desired velocities (Fig. 5.7).

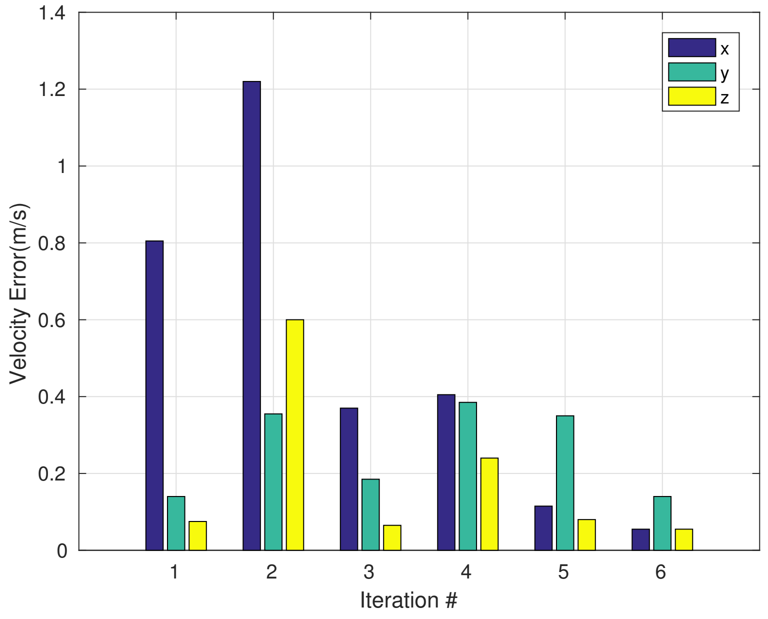

Figure 5.7 Velocity Improvement for 90 Degree window

Because this maneuver operates further outside of the linearized space considered in the controller design, it seems that the approximations used were less precise than for the previous maneuver. This also became clear when observing the position precision (Fig. 5.8). Here, the possible x position values of the Crazyflie 2.0 varied within a range of over 30 cm and had a significantly higher standard deviation with respect to both axes than in the 60 degree case. The parameter α was decreased in order to decrease the lateral acceleration during the attitude phase and thus hopefully improve the precision of the maneuver (Fig. 5.9).

Just by decreasing the overall thrust through phases 3 and 4, the standard deviation of the quadrotor position in the plane of the window decreased by almost 50%. The z direction

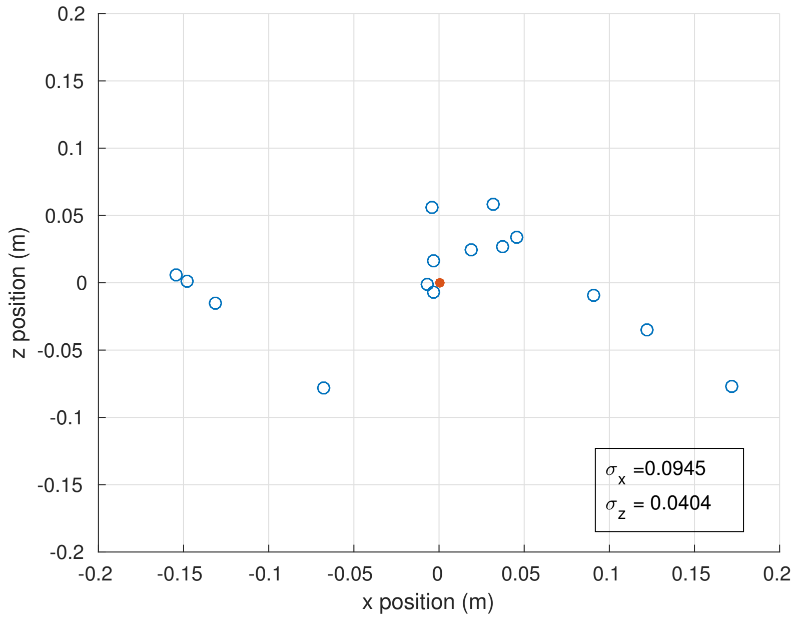

Figure 5.8 Precision of 90 Degree Vertical Window Maneuver, $\alpha = 1.0$

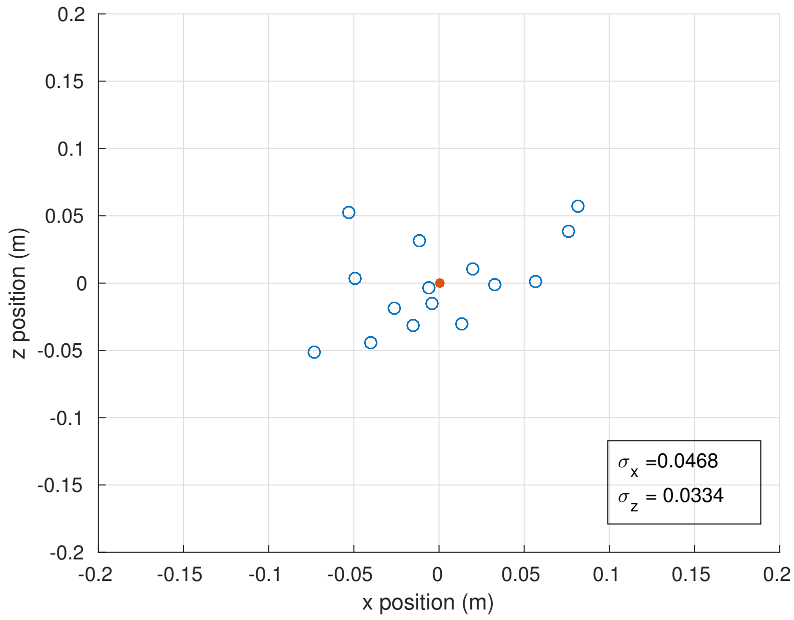

Figure 5.9 Precision of 90 Degree Vertical Window Maneuver, $\alpha = 0.8$

standard deviation decreased as well. The trade-off with lowering the value of α is it effectively decreases the thrust being delivered at the beginning of the final hover phase. Because of the motor dynamics, the desired propeller speeds require more time to be reached, and thus more vertical space to recover. This phenomenon becomes more important in an area with limited vertical space (i.e low ceilings). A visualization of a representative test-run can be seen in figure 5.10.

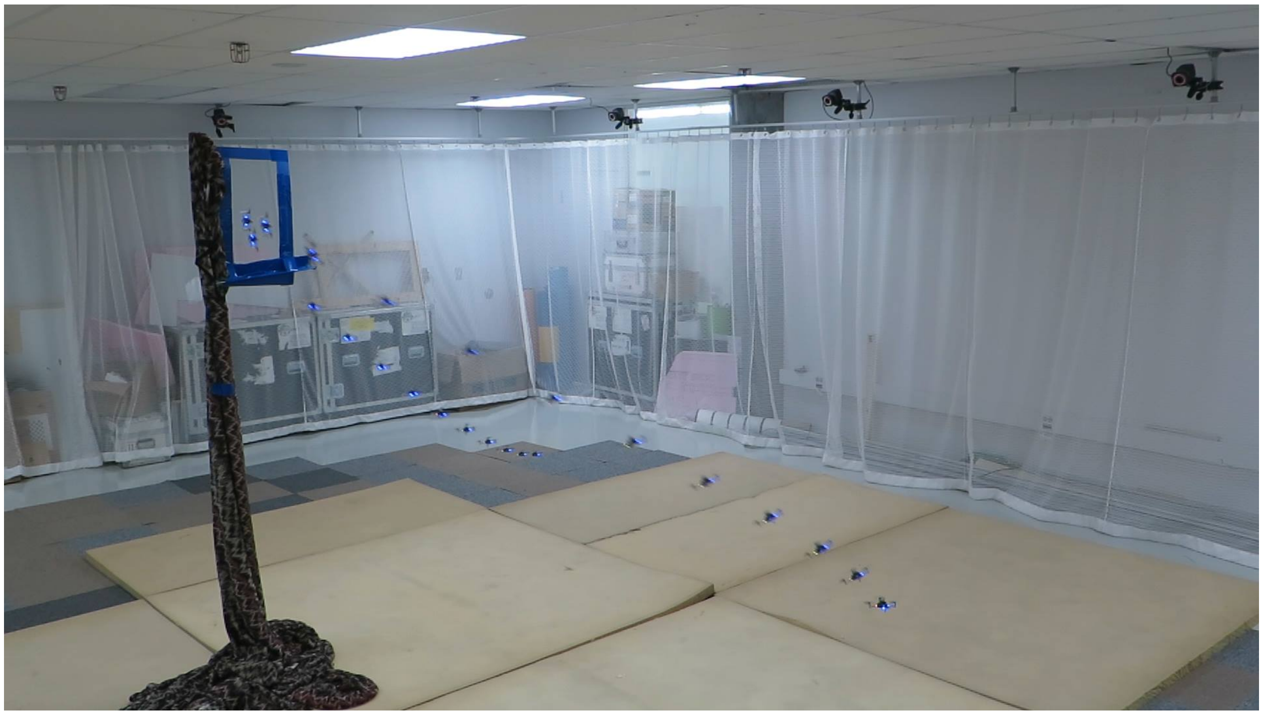

Figure 5.10 90 Degree Vertical Test

5.3.3 Descent Through Horizontal Window

The goal state for vertical descent maneuver was as follows :

— $\mathbf{r}_G = \begin{bmatrix} 0 & 0 & 2 \end{bmatrix}^\top$ m
— $\mathbf{v}_G = \begin{bmatrix} 0 & 0 & -0.6 \end{bmatrix}^\top$ m/s
— $\phi_G = \pi/2\,\text{rad}$
— $\psi_G = 0\,\text{rad}$

Here, it was found that for the limited space of the flight arena available, only relatively small values for the z velocity were realistic for producing a repeatable maneuver with stable recovery. We again refined the launch velocity in order to achieve the goal state through the window (Fig. 5.11). We see a similar rate of convergence to that of the 90 degree vertical window. Regardless of the soft hover controller gains chosen, the recovery stage of the maneuver only generated reliable stable results with higher values of α. This is once again due to the limitations of the motor actuators. As in the vertical window case, the high value

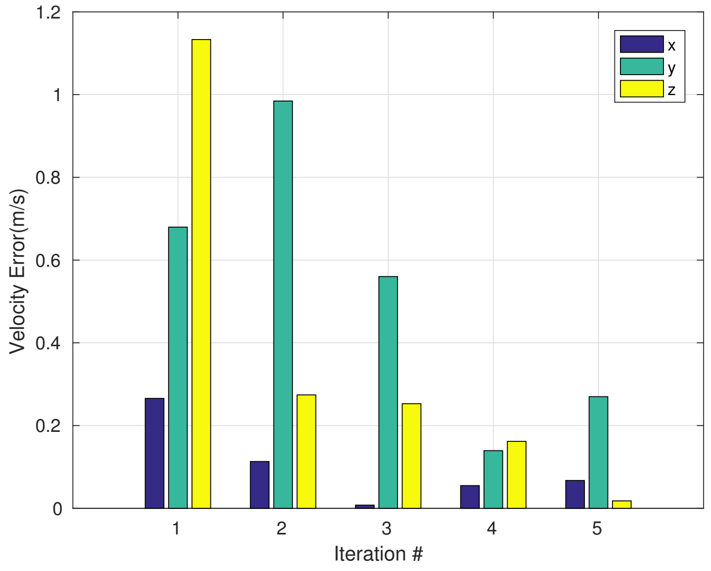

Figure 5.11 Velocity Improvement for Descent through a Horizontal Window

of α reduced the precision of the maneuver (Fig. 5.12). We see once again, as for the 90 degree window case with $\alpha = 1$, the user must accept a relatively large margin of error in the precision of the maneuver. It is clear that for a confined space such as the flight arena used, the controller architecture proposed must be modified to increase the stability of phase 5. A typical test run can be seen in figure 5.13.

5.4 Conclusion

Results presented in [1] of a control architecture enabling aggressive flight through windows were reproduced and validated with limited success. The process of sequential composition of trajectories enabled precise, repeatable maneuvers at the cost of time-consuming adaptation phases that would not be feasible in real world applications. Much of the error produced in testing can be attributed to multiple simplifications that could be considered in future testing. In terms of the 3D path-following controller, only line segments with unrealistic instantaneous accelerations to goal velocities were considered. More realistically achievable minimal snap trajectories have been proposed and developed in [23]. In terms of the nonlinear

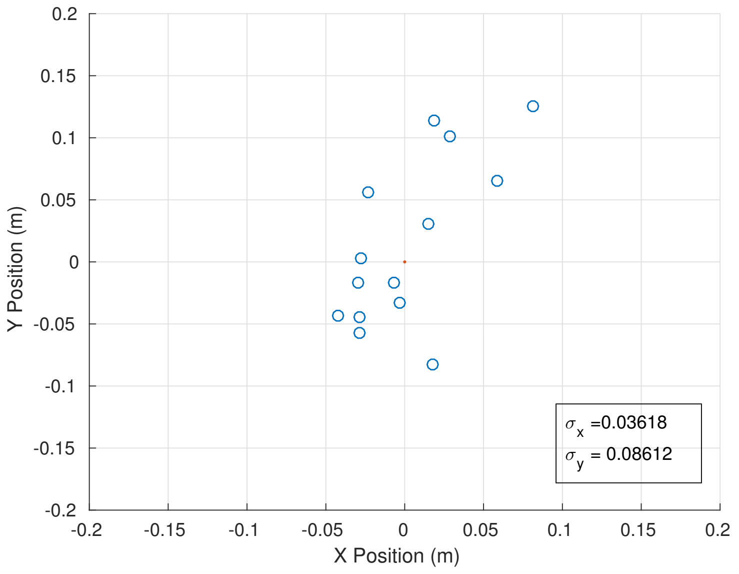

Figure 5.12 Results for 15 tests of a Vertical Descent. X-Y position deviation of the quadrotor as it passes through the horizontal plane of the window

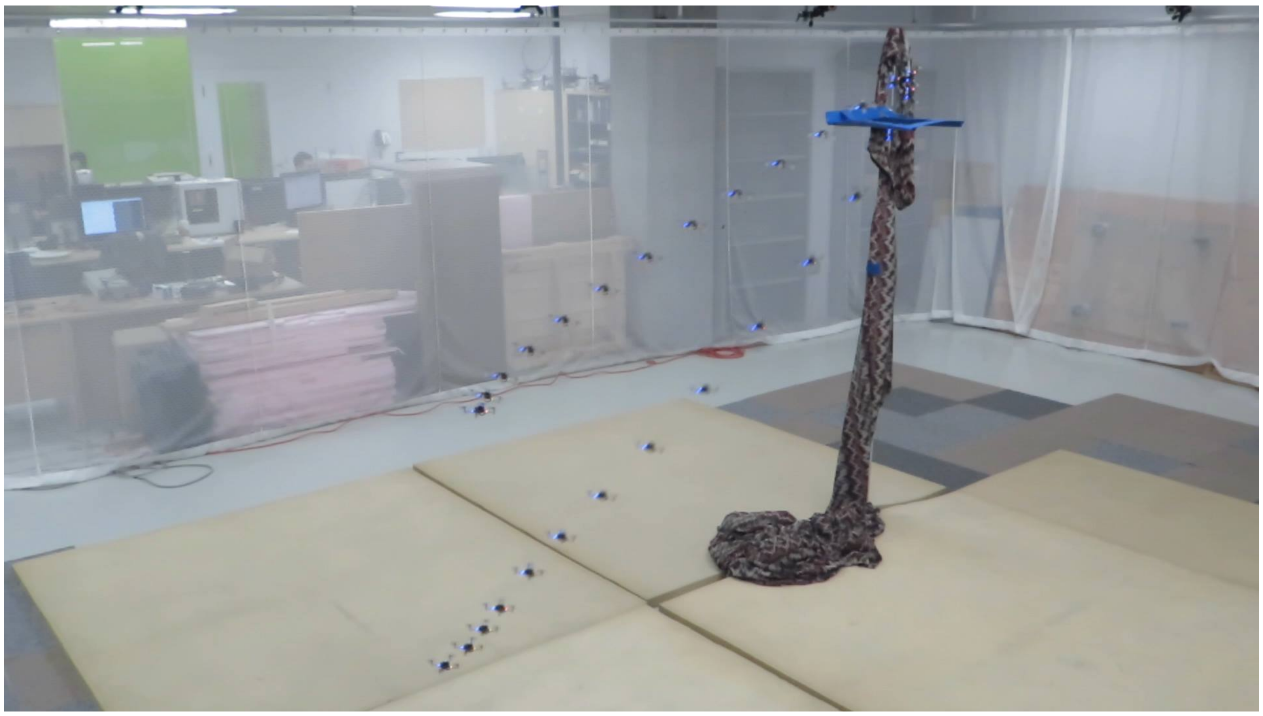

Figure 5.13 Horizontal Window Test Run

model, several simplifications were made that greatly reduced the accuracy of the maneuvers. Using a nonlinear trajectory controller such as the one presented in [54] could provide a more mathematically rigorous solution to this problem. Furthermore, increasing the complexity of the problem by applying an aerodynamic model similar to the one presented in [13] could improve upon the hypotheses used in section 5.2.1.

The advantage of the techniques used in this chapter is the simplicity of each individual component. However, in order to make these manoeuvres more applicable in the real world, a more complex and elegant solution is necessary. Results in [54] enable the convergence of a quadrotor onto a sufficiently smooth 3D trajectory. This solution, however, requires a feedback of higher-order elements of its state, including acceleration and jerk. These elements are rarely used and seldom appear in control literature, mainly due to difficulty in their estimation. The following chapters describe an effort to fill this gap in the field.

CHAPTER 6 REAL-TIME JERK ESTIMATION OF A QUADROTOR UAV

A critical component of any control system application is the ability to accurately measure the state of the system being controlled. For a UAV, the necessary elements for stable flight are usually its position, velocity, orientation and angular velocity. However, as mentioned in section 5.4, there exist control applications that rely on the feedback of higher-order elements - namely the acceleration and jerk of a vehicle. Unfortunately, at this time there exists no easily available jerk sensor for these purposes. Furthermore, with applications using UAV's with rotating propellers, data extracted from on-board accelerometers is often affected by high levels of noise. Any resulting control application using the feedback of the jerk and acceleration therefore must rely on estimation techniques, for which there is only a minimal amount of existing theory. This chapter aims to develop a method for the estimation of the jerk that can be immediately applied to control applications. It outlines several types of linear Kalman filter (LKF) with different levels of complexity and attempts to demonstrate the most reliable method for testing. It also describes a method for validating the quality of the estimate offline using a bounded least-squares technique.

6.1 Linear Jerk Estimator

6.1.1 Estimation Based only on Accelerometer Data

Depending on the physical system being studied, there are several options for how to design an estimator for the jerk. The first attempt made in this project was to estimate the jerk using only measurements from the accelerometers. Details of this estimator are outlined in subsection 6.1.1.1. The estimate was then augmented using a precise measurement of position in subsection 6.1.3.

6.1.1.1 Kalman Filter with Coloured Noise Input

A Kalman filter similar to the one presented in [3] was developed. This approach models the jerk as being the result of a random walk process and models the noise as being the sum of white noise and coloured noise components. This could be useful in terms of taking into account the component of noise produced by the rotating propellers of a UAV. The model of the system is depicted in figure 6.1.

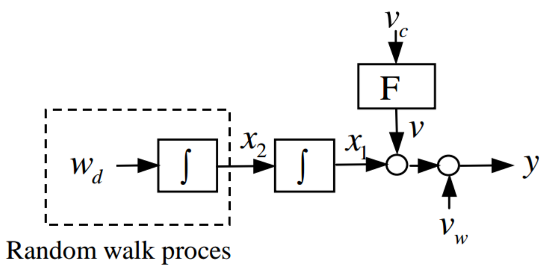

Figure 6.1 Random Walk Process (taken from [3])

Figure 6.1 shows how the estimator output y is modeled as a function of system disturbance ω_d and measurement noise v_c and v_w. The signal v_c is passed through a first order low-pass filter F in order to approximate the behaviour of coloured noise. F is defined to have the following dynamic :

$$\dot{\eta} = \Phi_F \boldsymbol{\eta} + \Gamma_F v_c \tag{6.1}$$

$$\boldsymbol{v} = H_F \boldsymbol{\eta} + d v_c \tag{6.2}$$

where Φ_F, Γ_F, H_F and d can be chosen to produce the desired frequency response. The variances of ω_d, v_w and v_c are defined as as σ_d^2 , σ_w^2 and σ_c^2 respectively. The signals ω_d, v_w and v_c can then be expressed as $\omega_d = \sigma_d \omega_1$, $v_w = \sigma_w \omega_2$ and $v_c = \sigma_c \omega_3$, respectively. $\omega_i (i = 1, 2, 3)$ are considered to be mutually independent zero-mean white noise processes with unit variances. The resulting system has the following state-space representation :

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\boldsymbol{\omega} \tag{6.3}$$

$$y = \mathbf{C}\mathbf{x} + \mathbf{D}\boldsymbol{\omega} \tag{6.4}$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \boldsymbol{\eta}^\top \end{bmatrix}^\top \tag{6.5}$$

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \end{bmatrix}^\top \tag{6.6}$$

and

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \Phi_F \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ \sigma_d & 0 & 0 \\ 0 & 0 & \sigma_c \Gamma_F \end{bmatrix} \tag{6.7}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & H_F \end{bmatrix}, \mathbf{D} = \begin{bmatrix} 0 & \sigma_w & d\sigma_c \end{bmatrix} \tag{6.8}$$

Using the same procedure developed in [3], a continuous Kalman filter estimate can be developed that takes into account the correlation between the dynamic noise term and the observation noise term in eq. 6.3. The time derivative of the state estimate $\hat{\mathbf{x}}$ can be expressed as :

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{K}(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}) \tag{6.9}$$

where $\mathbf{K}$ is the Kalman filter gain matrix given by :

$$\mathbf{K} = \mathbf{P}\mathbf{C}^\top(\mathbf{D}\mathbf{D}^\top)^{-1} + \mathbf{B}\mathbf{D}(\mathbf{D}\mathbf{D}^\top)^{-1} \tag{6.10}$$

where $\mathbf{P}$ is the symmetric positive definite matrix, solution of the following Riccati equation

$$\begin{aligned} \mathbf{P}(\mathbf{A} - \mathbf{B}\mathbf{D}(\mathbf{D}\mathbf{D}^\top)^{-1}\mathbf{C})^\top + (\mathbf{A} - \mathbf{B}\mathbf{D}(\mathbf{D}\mathbf{D}^\top)^{-1}\mathbf{C})\mathbf{P} - \mathbf{P}\mathbf{C}^\top(\mathbf{D}\mathbf{D}^\top)^{-1}\mathbf{C}\mathbf{P} \\ + \mathbf{B}\left(\mathbf{I} - \mathbf{D}^\top(\mathbf{D}\mathbf{D}^\top)^{-1}\mathbf{D}\right)\mathbf{B}^\top = \mathbf{0} \end{aligned} \tag{6.11}$$

In application, this 2nd order filter is then discretized using a bilinear transformation with a sample time of 0.01 s.

6.1.1.2 Simulation Results

To evaluate the validity of the Kalman filter with coloured noise component, simulations were run for coloured and white noises with variances $\sigma_c^2 = 0.05$ and $\sigma_w^2 = 0.001$ respectively. System disturbance variance σ_d^2 was altered as a design parameter. Figure 6.2 shows the estimator response with $\sigma_d^2 = 50$. This value of σ_d gives both a delayed response and significant error in the jerk and acceleration estimates. The value of σ_d must be increased in order to obtain a usable estimate for real-time applications.

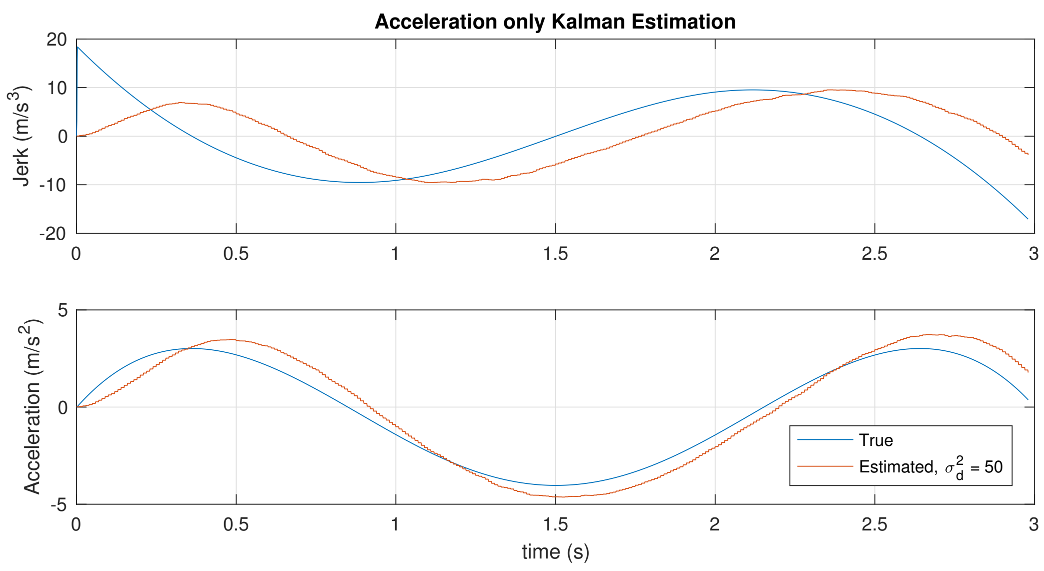

Figure 6.2 Estimated Acceleration and Jerk using Accelerometer Data, $\sigma_d^2 = 50$

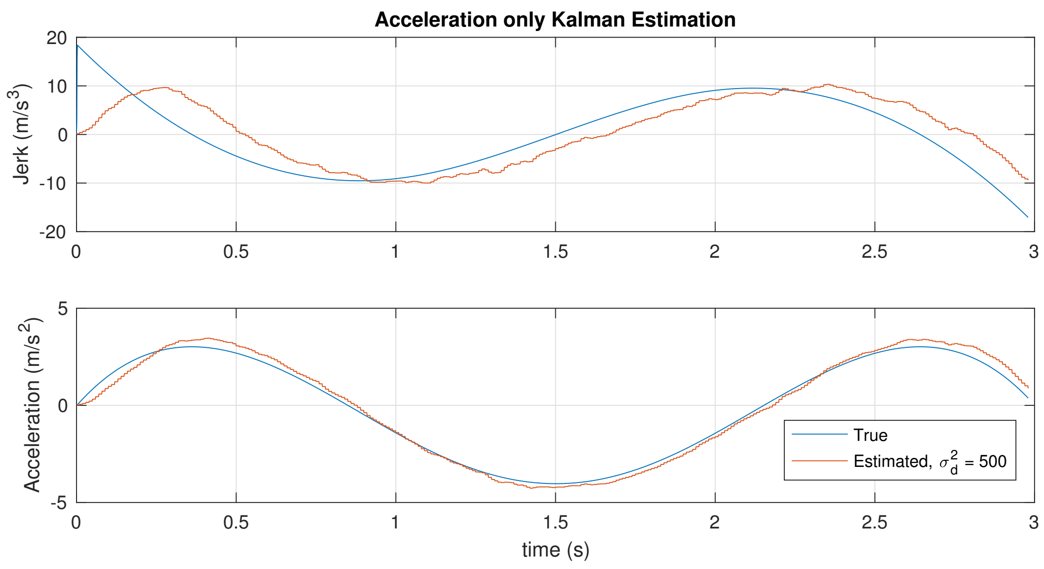

Figure 6.3 Estimated Acceleration and Jerk using Accelerometer Data, $\sigma_d^2 = 500$

Figure 6.2 shows the estimator response for $\sigma_d^2 = 500$. Here the estimator places more confidence in the acceleration measurement, giving a faster yet noisier response in both jerk and acceleration estimates.

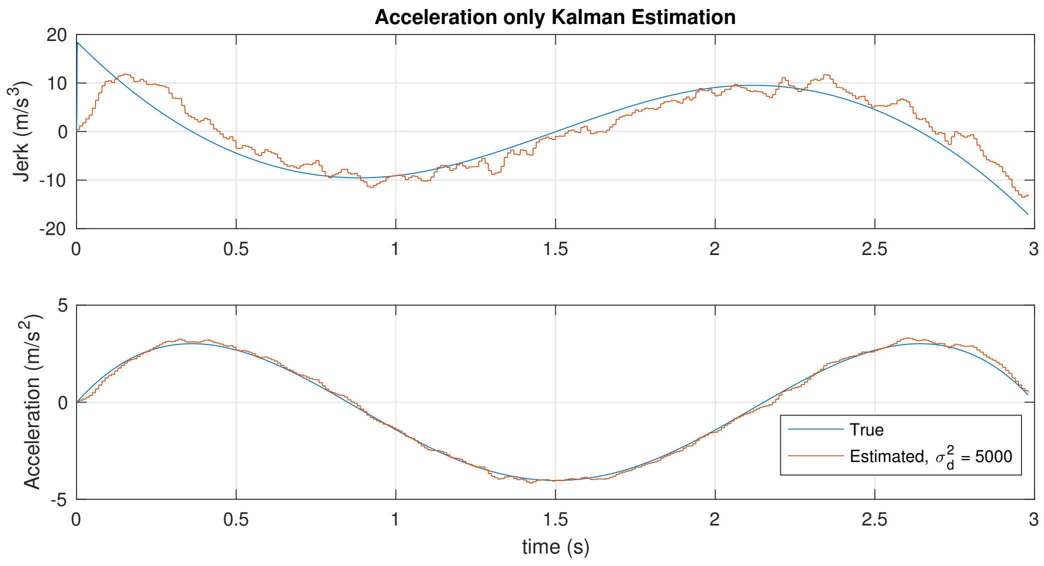

Figure 6.4 Estimated Acceleration and Jerk using Accelerometer Data, $\sigma_d^2 = 5000$

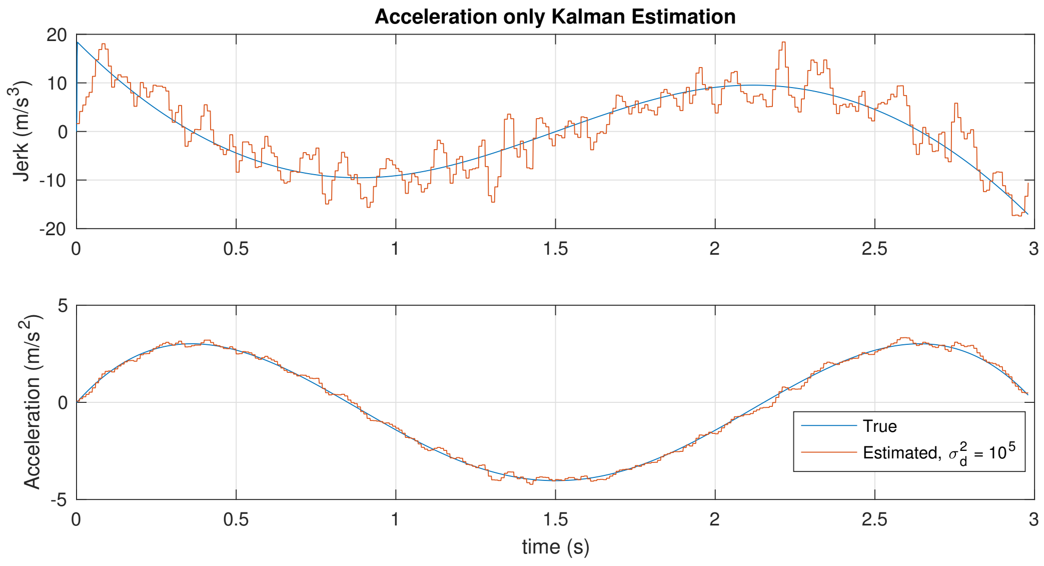

Figure 6.5 Estimated Acceleration and Jerk using Accelerometer Data, $\sigma_d^2 = 10^5$

Using only acceleration measurements, the most reliable estimate came when $\sigma_d^2 = 5000$ as seen in figure 6.4. When σ_d^2 was chosen to be higher such as in figure 6.5, the effects of measurement noise degraded the resulting estimate to the point where it became unusable. In general, however, this method was shown to be unsatisfactory for a precise and reliable estimate of the acceleration and jerk.

6.1.2 Method for a Standard Discrete Linear Kalman Filter

The Kalman filter used in section 6.1.1.1 was developed using a continuous system and then discretized. Using a discrete Kalman filter can add useful information and potentially improve the overall estimate. A discrete Kalman filter is used for a state-space LTI system of the form :

$$\mathbf{x}(i+1) = \mathbf{A}\mathbf{x}(i) + \mathbf{B}\mathbf{u}(i) + \mathbf{G}\boldsymbol{\omega}(i) \tag{6.12}$$

$$\mathbf{y}(i) = \mathbf{C}\mathbf{x}(i) + \mathbf{D}\mathbf{u}(i) + \mathbf{H}\boldsymbol{\omega}(i) + \mathbf{v}(i) \tag{6.13}$$

with input $\mathbf{u}$, process noise $\boldsymbol{\omega}$ and measurement noise $\mathbf{v}$ having the following characteristics :

$$\mathbb{E}[\boldsymbol{\omega}(n)] = \mathbb{E}[\mathbf{v}(n)] = 0 \tag{6.14}$$

$$\mathbb{E}[\boldsymbol{\omega}(n)\boldsymbol{\omega}^\top(n)] = \mathbf{Q} \tag{6.15}$$

$$\mathbb{E}[\mathbf{v}(n)\mathbf{v}^\top(n)] = \mathbf{R} \tag{6.16}$$

A Kalman filter can then be constructed according to :

$$\hat{\mathbf{x}}(i+1|i) = \mathbf{A}\hat{\mathbf{x}}(i|i-1) + \mathbf{B}\mathbf{u}(i) + \mathbf{L}(i)(\mathbf{y}(i) - \mathbf{C}\hat{\mathbf{x}}(i|i-1) - \mathbf{D}\mathbf{u}(i)) \tag{6.17}$$

where the Kalman filter gain $\mathbf{L}(i)$ is found by solving the discrete Riccati equation :

$$\mathbf{L}(i) = (\mathbf{A}\mathbf{P}(i)\mathbf{C}^\top + \bar{\mathbf{N}})(\mathbf{C}\mathbf{P}(i)\mathbf{C}^\top + \bar{\mathbf{R}})^{-1} \tag{6.18}$$

$$\mathbf{M}(i) = \mathbf{P}(i)\mathbf{C}^\top(\mathbf{C}\mathbf{P}(i)\mathbf{C}^\top + \bar{\mathbf{R}})^{-1} \tag{6.19}$$

$$\mathbf{Z}(i) = (\mathbf{I} - \mathbf{M}(i)\mathbf{C}\mathbf{P}(i)(\mathbf{I} - \mathbf{M}(i)\mathbf{C})^\top + \mathbf{M}(i)\bar{\mathbf{R}}\mathbf{M}(i)^\top) \tag{6.20}$$

$$\mathbf{P}(i+1) = (\mathbf{A} - \bar{\mathbf{N}}\bar{\mathbf{R}}^{-1}\mathbf{C})\mathbf{Z}(i)(\mathbf{A} - \bar{\mathbf{N}}\bar{\mathbf{R}}^{-1}\mathbf{C})^\top + \bar{\mathbf{Q}} \tag{6.21}$$

where :

$$\bar{\mathbf{Q}} = \mathbf{G}\mathbf{Q}\mathbf{G}^\top \tag{6.22}$$

$$\bar{\mathbf{R}} = \mathbf{R} + \mathbf{H}\mathbf{Q}\mathbf{H}^\top \tag{6.23}$$

$$\bar{\mathbf{N}} = \mathbf{G}\mathbf{Q}\mathbf{H}^\top \tag{6.24}$$

$$\mathbf{P}(i) = \mathbb{E}[(\mathbf{x} - \hat{\mathbf{x}}(i|i-1))(\mathbf{x} - \hat{\mathbf{x}}(i|i-1))^\top] \tag{6.25}$$

$$\mathbf{Z}(i) = \mathbb{E}[(\mathbf{x} - \hat{\mathbf{x}}(i|i))(\mathbf{x} - \hat{\mathbf{x}}(i|i))^\top] \tag{6.26}$$

A current discrete-time estimator was implemented that makes use of the current state measurements. It takes the values of $\hat{\mathbf{x}}(i|i-1)$ and $\mathbf{y}(i)$ and generates :

$$\hat{\mathbf{x}}(i|i) = \hat{\mathbf{x}}(i|i-1) + \mathbf{M}(i)(\mathbf{y}(i) - \mathbf{C}\hat{\mathbf{x}}(i|i-1) - \mathbf{D}\mathbf{u}(i)) \tag{6.27}$$

Using this general structure, the problem could then reduced to defining the system parameters and choosing the appropriate $\mathbf{Q}$ and $\mathbf{R}$ matrices.

6.1.3 Kalman Filter using Position and Acceleration Measurements

As discussed in Introduction, the acceleration is often not the only measurement available. The next logical step to improve the jerk estimate was to include a position measurement. Two estimators were developed to use these measurements. The first was a Kalman filter considering the measurement disturbance as purely white noise, and another using the coloured noise model described in section 6.1.1.1.

6.1.3.1 White Noise Kalman Filter

The simple white noise Kalman filter was developed according to the system in figure 6.6.

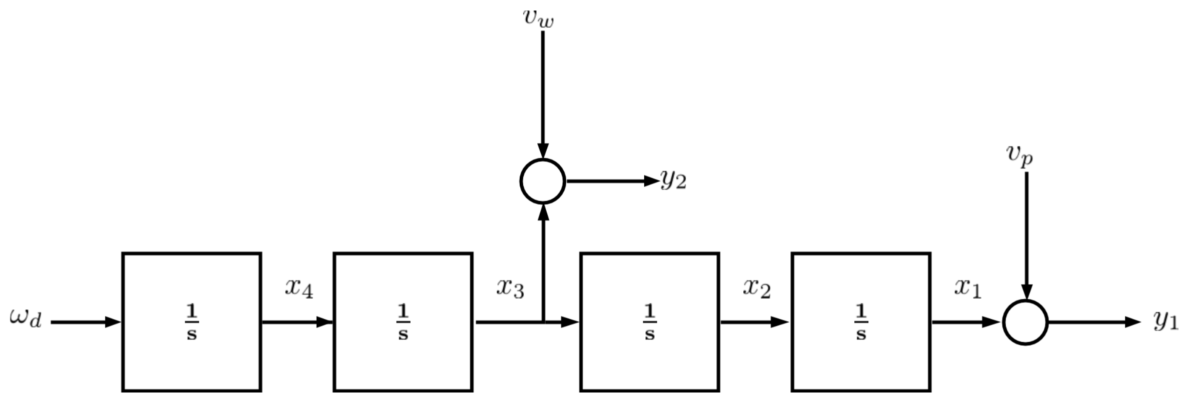

Figure 6.6 Random Walk Process with Acceleration and Position Measurement

An added benefit to using this filter as opposed to the filter developed in section 6.1.1.1 is that it provides a full estimate of the system from jerk through position. This way, a second estimator is not needed to estimate the vehicle's velocity. The state-space representation of this system is :

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{G}\omega \tag{6.28}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{v} \tag{6.29}$$

where the vectors are :

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^\top \tag{6.30}$$

$$\omega = \omega_d \tag{6.31}$$

$$\boldsymbol{v} = \begin{bmatrix} v_p & v_w \end{bmatrix}^\top \tag{6.32}$$

and the associated matrices are given by :

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{6.33}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{6.34}$$

The discrete-time equivalent is then :

$$\mathbf{x}(i+1) = \mathbf{A}_d\mathbf{x}(i) + \mathbf{G}_d\omega(i) \tag{6.35}$$

$$\mathbf{y}(i) = \mathbf{C}_d\mathbf{x}(i) + \mathbf{v}(i) \tag{6.36}$$

where the matrices are :

$$\mathbf{A}_d = \begin{bmatrix} 1 & T & \frac{T^2}{2} & \frac{T^3}{6} \\ 0 & 1 & T & \frac{T^2}{2} \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{G}_d = \begin{bmatrix} \frac{T^4}{24} \\ \frac{T^3}{6} \\ \frac{T^2}{2} \\ T \end{bmatrix} \tag{6.37}$$

$$\mathbf{C}_d = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{6.38}$$

with T the sampling period. A value of $T = 0.01s$ is used for the purposes of simulations as well as tests in subsequent sections. The system covariance matrices $\mathbf{Q}$ and $\mathbf{R}$:

$$\mathbf{Q} = \sigma_d^2 \tag{6.39}$$

$$\mathbf{R} = \text{diag}(\sigma_z^2, \sigma_a^2) \tag{6.40}$$

6.1.3.2 Coloured Noise Kalman Filter

The coloured-noise Kalman filter is augmented to include the position estimate. Its state-space representation is as follows :

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{G}\boldsymbol{\omega} \tag{6.41}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{v} \tag{6.42}$$

where the vectors are given by

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & \boldsymbol{\eta} \end{bmatrix}^\top \tag{6.43}$$

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_d & v_c \end{bmatrix}^\top \tag{6.44}$$

$$\mathbf{v} = \begin{bmatrix} v_p & v_w \end{bmatrix}^\top \tag{6.45}$$

and the matrices by

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \Phi_F \end{bmatrix}, \mathbf{G} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & \Gamma_F \end{bmatrix} \tag{6.46}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & H \end{bmatrix} \tag{6.47}$$

The discrete-time equivalent is :

$$\mathbf{x}(i+1) = \mathbf{A}_d\mathbf{x}(i) + \mathbf{G}_d\boldsymbol{\omega}(i) \tag{6.48}$$

$$\mathbf{y}(i) = \mathbf{C}_d\mathbf{x}(i) + \mathbf{v}(i) \tag{6.49}$$

with

$$\mathbf{A}_d = \begin{bmatrix} 1 & T & \frac{T^2}{2} & \frac{T^3}{6} & 0 \\ 0 & 1 & T & \frac{T^2}{2} & 0 \\ 0 & 0 & 1 & T & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & e^{\Phi_F T} \end{bmatrix}, \mathbf{G}_d = \begin{bmatrix} \frac{T^4}{24} & 0 \\ \frac{T^3}{6} & 0 \\ \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & 1 - e^{-\Gamma_F T} \end{bmatrix} \tag{6.50}$$

$$\mathbf{C}_d = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & H \end{bmatrix} \tag{6.51}$$

and

$$\mathbf{Q} = \text{diag}(\sigma_d^2, \sigma_c^2) \tag{6.52}$$

$$\mathbf{R} = \text{diag}(\sigma_z^2, \sigma_a^2) \tag{6.53}$$

6.1.3.3 Simulation Results

Subject to the same combination of coloured and white measurement noise, it was possible to obtain a very similar estimation performance with both estimators. The two estimators are shown in response to a white noise with variance $\sigma_c^2 = 0.05$ and $\sigma_w^2 = 0.001$. Figure 6.7 shows a performance improvement from results obtained in section 6.1.1.2. The estimator responses for the coloured and white noise estimators generated virtually identical results. It was concluded that the added complexity of the estimator defined in [3] did not significantly improve the quality of the estimate when adding the position measurement.

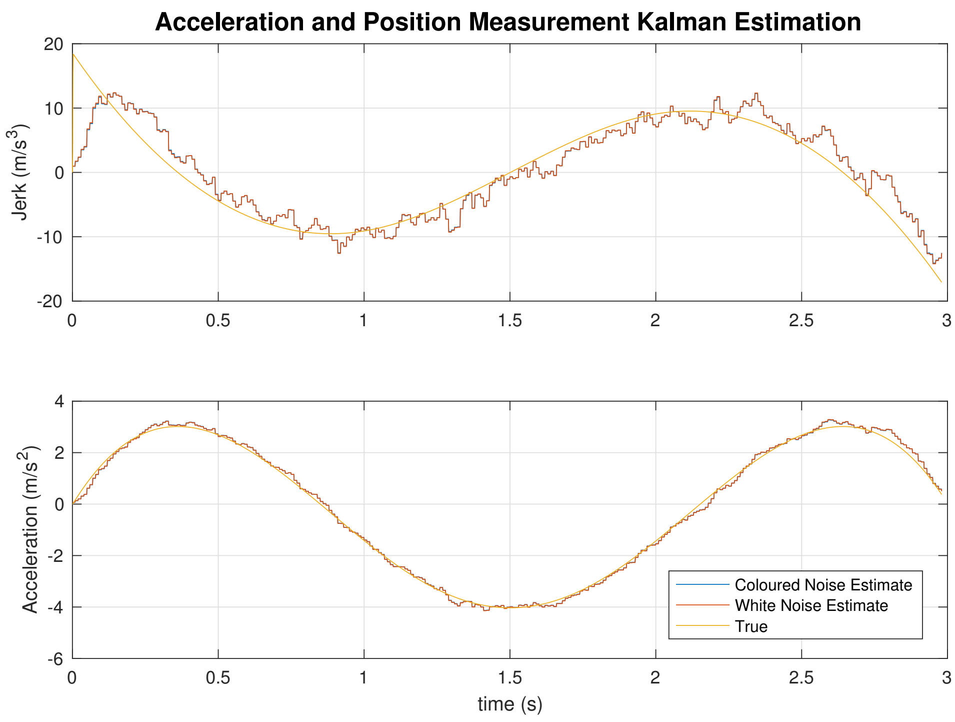

Figure 6.7 Estimated Acceleration and Jerk using Acceleration and Position Measurement.

6.1.4 Estimator with Virtual-Jerk Rate Command as System Input

The most complete version of Kalman filter that was developed used the virtual jerk rate command from the controller developed in chapter 7. the system is the same as the one described in section 6.1.3 with a slight modification of the system noise variables, i.e.,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{G}\boldsymbol{\omega} \tag{6.54}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{v} \tag{6.55}$$

with

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^\top \tag{6.56}$$

$$u = u_{vs} \tag{6.57}$$

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_{a_z} & \omega_{j_z} \end{bmatrix}^\top, \tag{6.58}$$

$$\mathbf{v} = \begin{bmatrix} v_z & v_a \end{bmatrix}^\top \tag{6.59}$$

Where ω_{a_z} and ω_{j_z} are system disturbance inputs and u_{vs} is a virtual command that would theoretically influence the fourth derivative of the position of the quadrotor. The resulting A, B, C, G matrices are :

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{6.60}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \mathbf{G} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.61}$$

The discrete-time equivalent is :

$$\mathbf{x}(i+1) = \mathbf{A}_d\mathbf{x}(i) + \mathbf{B}_d\mathbf{u}(i) + \mathbf{G}_d\boldsymbol{\omega}(i) \tag{6.62}$$

$$\mathbf{y}(i) = \mathbf{C}_d\mathbf{x}(i) + \mathbf{v}(i) \tag{6.63}$$

with

$$\mathbf{A}_d = \begin{bmatrix} 1 & T & \frac{T^2}{2} & \frac{T^3}{6} \\ 0 & 1 & T & \frac{T^2}{2} \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{B}_d = \begin{bmatrix} \frac{T^4}{24} \\ \frac{T^3}{6} \\ \frac{T^2}{2} \\ T \end{bmatrix} \tag{6.64}$$

$$\mathbf{C}_d = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \mathbf{G}_d = \begin{bmatrix} \frac{T^3}{6} & \frac{T^4}{24} \\ \frac{T^2}{2} & \frac{T^3}{6} \\ T & \frac{T^2}{2} \\ 0 & T \end{bmatrix} \tag{6.65}$$

and system covariance matrices $\mathbf{Q}$ and $\mathbf{R}$:

$$\mathbf{Q} = \mathrm{diag}(\sigma_z^2, \sigma_{v_z}^2, \sigma_{a_z}^2, \sigma_{j_z}^2) \tag{6.66}$$

$$\mathbf{R} = \mathrm{diag}(\sigma_{z_m}^2, \sigma_{a_m}^2) \tag{6.67}$$

where the system disturbance variances $\sigma^2_{j_z}$,$\sigma^2_{a_z}$,$\sigma^2_{v_z}$ and σ^2_z can be adjusted in order to account for the uncertainty of the model used. Measurement noise variances $\sigma^2_{a_m}$ and $\sigma^2_{z_m}$ are properties of the measurement devices. The main advantage to using this technique is that it incorporates yet another source of information for estimating the jerk. The amount of confidence that this virtual command actually reflects the behaviour of the object being tracked can be adjusted using the variances of ω_{a_z} and ω_{j_z}.

6.1.5 Simulation Results

This version of the estimator was then simulated using the controller developed in chapter 7. It was asked to track a reference input of 1 m from a resting position. To test its ability to reject outside disturbances, an external force was input at 20 seconds. The resulting step response can be seen in figure 6.9. In the case of this simulation, the following parameters

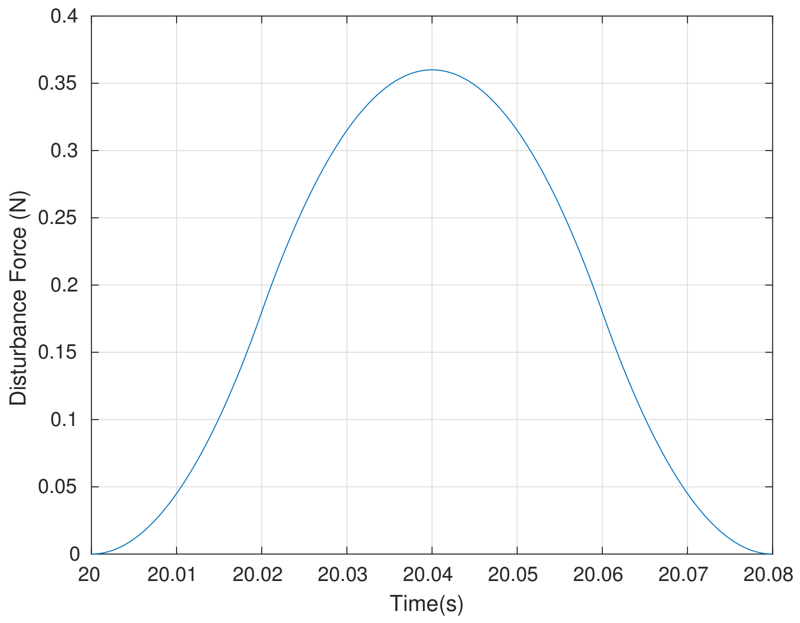

Figure 6.8 Profile of Disturbance Input for Estimator Simulation

were used. The optimal control $\mathbf{Q}$ and $\mathbf{R}$ matrices were chosen as :

$$\mathbf{Q}_{\text{LQR}} = \text{diag}(6, 10, 0, 1, 6) \tag{6.68}$$

$$\mathbf{R}_{\text{LQR}} = 1 \tag{6.69}$$

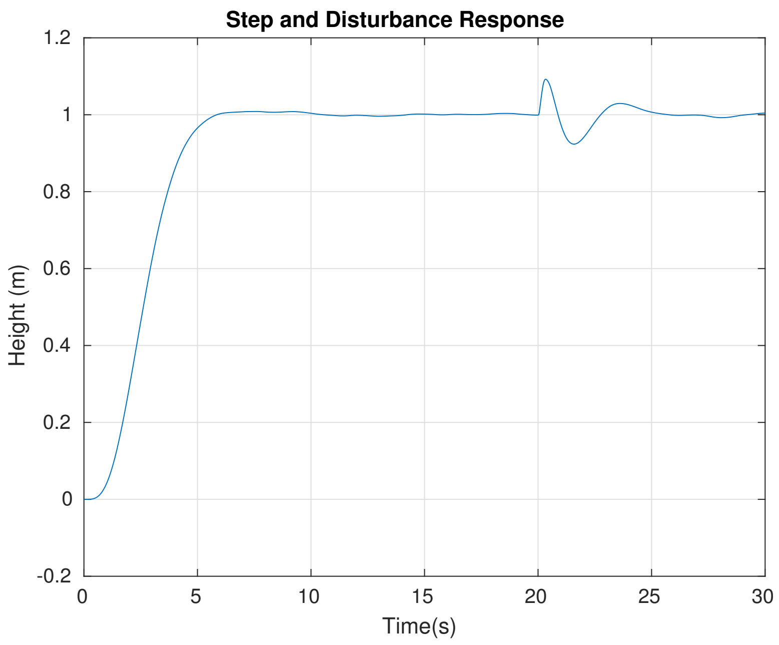

Figure 6.9 Step and Disturbance Response

which generated the following state-feedback gains :

$$\mathbf{K} = \begin{bmatrix} 5.5839 & 6.4932 & 3.4942 & 0.9676 & -2.0708 \end{bmatrix} \tag{6.70}$$

The parameters of the jerk estimators were chosen as follows for the estimator including control input :

$$\mathbf{Q}_{\text{Kal}} = \text{diag}(5 \cdot 10^{-3}, 5 \cdot 10^{-4}) \tag{6.71}$$

$$\mathbf{R}_{\text{Kal}} = \text{diag}(1.5 \cdot 10^{-8}, 3 \cdot 10^{-1}) \tag{6.72}$$

It would be interesting to compare this estimator to the one developed in the previous section. For these purposes, we establish the parameters for a jerk estimator using only acceleration and position estimates :

$$\mathbf{Q}_{\text{Kal}} = 5 \cdot 10^{3} \tag{6.73}$$

$$\mathbf{R}_{\text{Kal}} = \text{diag}(1.5 \cdot 10^{-8}, 3 \cdot 10^{-1}) \tag{6.74}$$

The resulting estimates are shown in figures 6.10 and 6.11

Figure 6.10 shows the jerk response of the first 10 seconds of the simulation. What becomes immediately obvious is the difference in level of noise between the measurement-only esti-

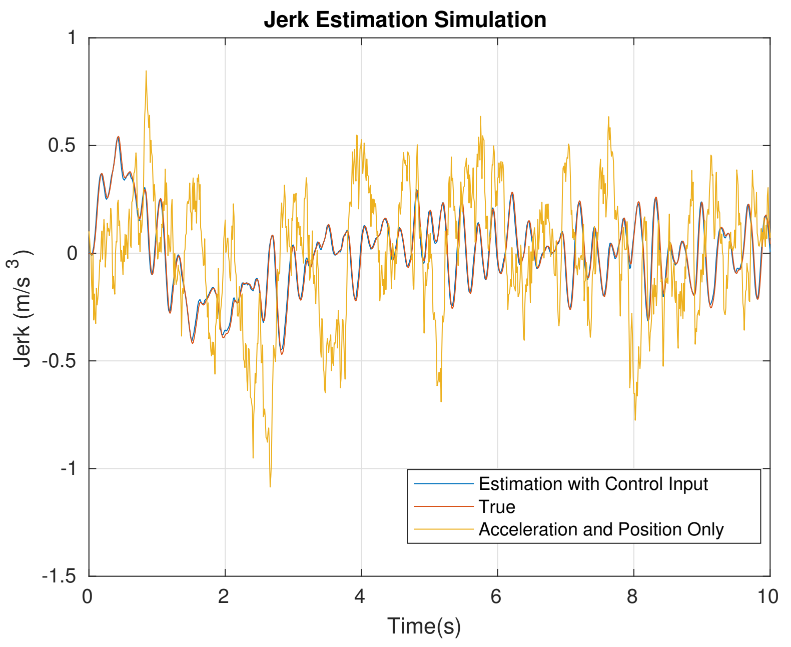

Figure 6.10 Jerk Estimation Simulation

mator and the one using the control input. The latter displays a much lower signal to noise ratio and tracks the true value more accurately.

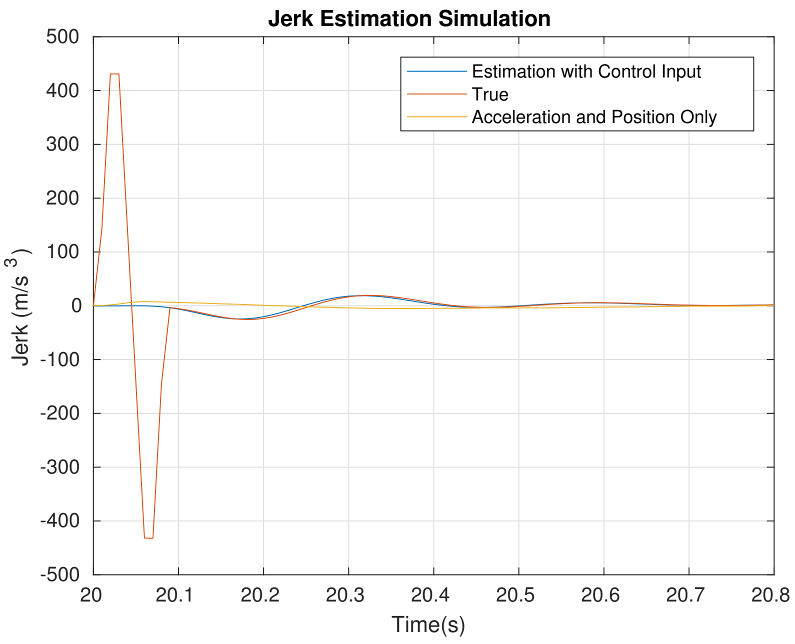

Figure 6.11 Jerk Estimation Simulation

In the seconds following the disturbance input, as presented in figure 6.10, we see the performance of both estimators when the jerk of the quadrotor is at higher levels (order of 100 vs. 0.5). Neither estimator had the capacity to respond immediately to the disturbance. However, after 0.1 seconds, The estimator with control input quickly converges to the true value, while the acceleration/position estimator fails to react in any substantial way. It can be concluded that including the control input does in fact improve the quality of the estimate of the jerk.

6.2 Estimator Validation

6.2.1 Bounded Least-Squares Offline Validation

Validating the developed controllers using actual test data instead of simulations proved to be a significant problem. The main issue in trying to determine the accuracy of the developed estimators came with the fact that there was no way to directly measure the jerk of the moving vehicle. The solution chosen was an offline technique that fitted a least-squares polynomial to the Vicon position data. This polynomial was then differentiated 3 times in order to represent the "actual" jerk of the vehicle in the z_w direction. The curve fitting problem was as follows :

$$\min_{x} \frac{1}{2} \left\| \mathbf{Cx} - \mathbf{d} \right\|^2 \tag{6.75}$$

$$s.t. : \begin{cases} \mathbf{Ax} \leq \mathbf{b} \\ \mathbf{A}_{eq}\mathbf{x} \leq \mathbf{b}_{eq} \\ \mathbf{lb} \leq \mathbf{x} \leq \mathbf{ub} \end{cases} \tag{6.76}$$

where $\mathbf{x} \in \mathbb{R}^{n+1}$ is the vector of the coefficients of the desired n^{th} order polynomial, $\mathbf{d} \in \mathbb{R}^m$ is a column vector of the m collected position data points and $\mathbf{C} \in \mathbb{R}^{m \times (n+1)}$ is the matrix corresponding to the time values of the simulation from t_1 to t_2, i.e.,

$$\mathbf{C} = \begin{bmatrix} t_1^n & t_1^{n-1} & \dots & t_1^0 \\ \vdots & & & \vdots \\ t_2^n & t_2^{n-1} & \dots & t_2^0 \end{bmatrix} \tag{6.77}$$

In order to better approximate the initial conditions of the section of data to be fitted, it was useful to apply equality constraints at the beginning and end of the desired time period.

These constraints were expressed in the $\mathbf{A}_{eq}$ and $\mathbf{B}_{eq}$ matrices :

$$\mathbf{A}_{eq} = \begin{bmatrix} nt^{n-1} & (n-1)t^{n-2} & \dots & t & 1 & 0 \\ n(n-1)t^{n-2} & & \dots & 1 & 0 & 0 \\ n(n-1)(n-2)t^{n-3} & \dots & 1 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{B}_{eq} = \begin{bmatrix} v(t) \\ a(t) \\ j(t) \end{bmatrix} \tag{6.78}$$

where t is a time value at which the velocity, acceleration or jerk are known.

6.2.2 Initial Curve-Fitting Results

To evaluate the performance of the estimator, tests were run with the MRASL's Crazyflie 2.0 nanoquadcopter. To simplify the results, the estimator was only run for the z_w direction. In order to determine the world-frame acceleration measurement equivalent, the body-frame acceleration measurements and rotated by a reliable on-board orientation estimate. Results for the offline method and the white-noise Kalman filter with position and acceleration measurements are presented in figure 6.12. It shows the Vicon data for a short section of the quadrotor's z_w trajectory overlayed with a least-squares fit computed as in section 6.2.2. The fit is not perfect, but closely follows the behaviour of the acquired data. The most notable differences are the discrepancies in the curvature of the polynomial and data at the end of the trajectory segment. This could cause a large difference in the real-time and offline estimations of the acceleration and jerk of the trajectory.

Figure 6.13 shows velocity estimates for the first derivative of the offline polynomial, a Kalman Filter using only position data and the white noise Kalman filter presented in section 6.1.3. The Kalman filter with both acceleration and position measurements generated significantly closer results to the offline result than the position only filter. This shows how using the accelerometer values would also be beneficial in situations where only the velocity and position are needed.

The initially generated polynomial was then differentiated again in order to estimate the true acceleration during the test. The dotted data in figure 6.14 corresponds to the world-frame acceleration measured by the accelerometer. The other two lines represent the real-time and offline estimates respectively. The real-time estimate follows the offline generated curve reasonably well until around 2s, when the acceleration of the polynomial rapidly increases. This difference can be attributed to a discrepancy in the bounds of the least-squared algorithm. More strict boundary conditions should be put on the generated polynomial.

The final state generated was the jerk. Figure 6.15 reveals how the measurement noise from the accelerometer has a significant effect on the jerk estimate. Furthermore, the offline gene-

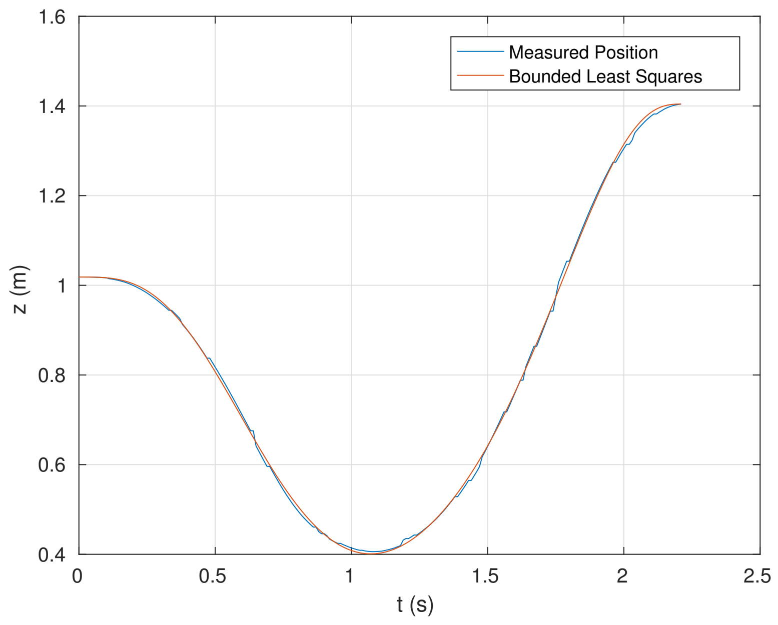

Figure 6.12 Off-Line Position Results

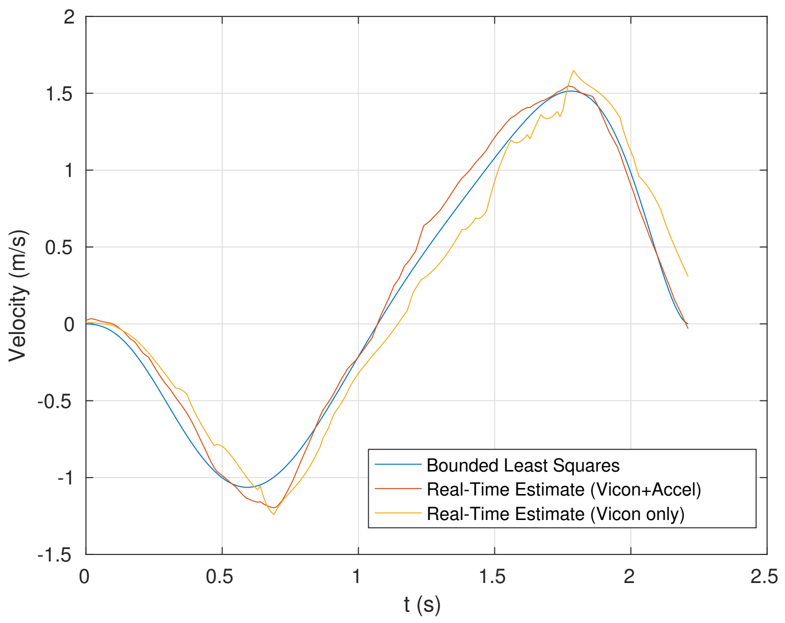

Figure 6.13 Off-Line Velocity Results

rated curve's jerk rapidly increases after $t = 2s$. The importance of the bounds was shown to indeed be substantial in finding an accurate offline estimation of the jerk. However, the 2 curves do generally follow each other, and after making certain refinements it is conceivable

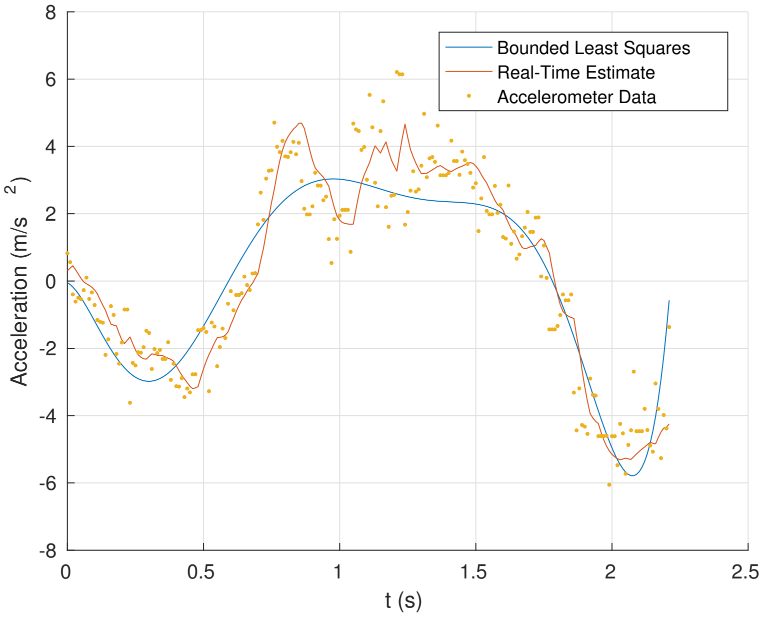

Figure 6.14 Off-Line Acceleration Results

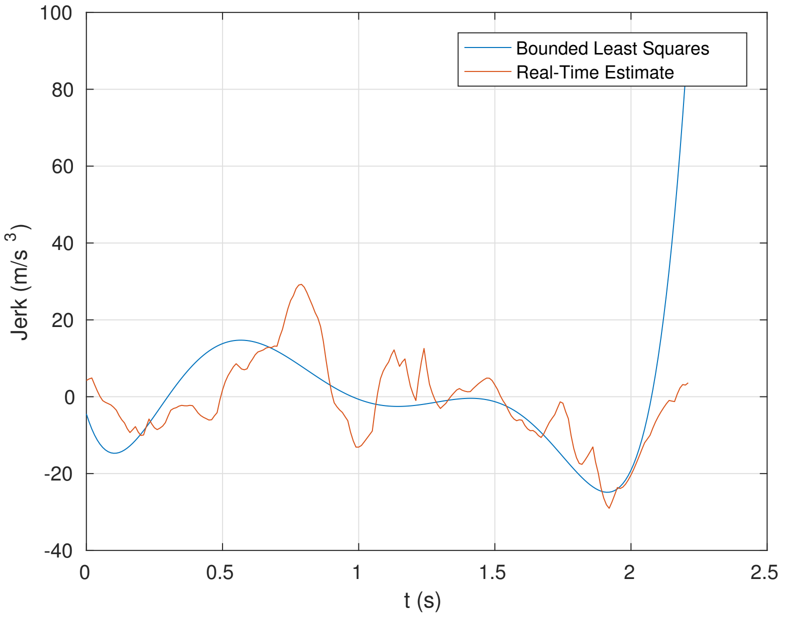

Figure 6.15 Off-Line Jerk Results

that the real-time and offline estimates could be accurate enough to be usable in a real-time control situation.

CHAPTER 7 JERK-AUGMENTED CONTROL

This chapter outlines the development of a controller that makes use of the estimation techniques presented in chapter 3. Controllers based on the reduced linearized model of a quadrotor are created in order to smoothly track reference inputs in x, y and z directions. Section 7.1 outlines the theoretical basics of the jerk-augmented controller and applies them to control the height of a quadrotor. Section 7.2 presents the optimal control techniques used to determine the desired gains for the jerk-augmented controller. These techniques are then expanded to control the x and y position of the vehicle in section 7.3.

7.1 Jerk-Augmented Height Control of a Quadrotor UAV

As discussed in introduction, limiting the jerk of a vehicle could be of interest for passenger vehicle applications. However, the jerk does not explicitly appear in the mathematical model presented in chapter 3. In order to be able to include the vertical acceleration and jerk of the quadrotor in the control design, the system must be augmented. To this end, we define a virtual command u_{vs} which delivers a virtual snap command to the quadrotor :

$$u_{vs} \equiv \frac{\mathrm{d}^2}{\mathrm{d}t^2}\left(\frac{\Delta_f}{m}\right) \tag{7.1}$$

The reduced height model presented in equation 3.13 is then augmented to include u_{vs} :

$$\underbrace{\begin{bmatrix} \Delta\dot{z} \\ \Delta\dot{v}_z \\ \Delta\dot{a}_z \\ \Delta\dot{j}_z \end{bmatrix}}_{\Delta\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \Delta z \\ \Delta v_z \\ \Delta a_z \\ \Delta j_z \end{bmatrix}}_{\Delta\mathbf{x}} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{\mathbf{B}} u_{vs} \tag{7.2}$$

which is in canonical controllable form. One notable element of this system is that it is unstable without a feedback of the jerk. This can be shown by analyzing the characteristic equation of the system with state feedback. The virtual snap command is defined as a function of the augmented state-variables :

$$u_{vs} = -\mathbf{K}\Delta\mathbf{x} \tag{7.3}$$

with $\mathbf{K} = \begin{bmatrix} k_z & k_{v_z} & k_{a_z} & k_{j_z} \end{bmatrix}$. The resulting closed-loop system has the following characteristic equation :

$$\begin{aligned} P(s) &= \det(s\mathbf{I}_4 - (\mathbf{A} - \mathbf{BK})) \\ &= s^4 + k_{j_z}s^3 + k_{a_z}s^2 + k_{v_z}s + k_z \end{aligned}$$

which, by the Routh-Hurwitz criterion, cannot be stable if $k_{j_z} = 0$ [8]. In order to ensure null steady-state error, feedback by integral-action can be introduced as well :

$$\underbrace{\begin{bmatrix} \Delta\dot{z} \\ \Delta\dot{v}_z \\ \Delta\dot{a}_z \\ \Delta\dot{j}_z \\ \Delta\dot{z}_i \end{bmatrix}}_{\Delta\dot{\mathbf{x}}_{aug}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}_{aug}} \underbrace{\begin{bmatrix} \Delta z \\ \Delta v_z \\ \Delta a_z \\ \Delta j_z \\ \Delta z_i \end{bmatrix}}_{\Delta\mathbf{x}_{aug}} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}}_{\mathbf{B}_{aug}} u_{vs} \tag{7.4}$$

The expression for u_{vs} is then :

$$u_{vs} = -\mathbf{K}\Delta\mathbf{x}_{aug} \tag{7.5}$$

However, in practice, only the vertical force f can be commanded. The resulting value of f can be calculated according to :

$$\Delta f = \iint m \cdot u_{vs}\mathrm{d}t \tag{7.6}$$

$$f(0) = f_e = -mg \tag{7.7}$$

$$\dot{f}(0) = \ddot{f}(0) \equiv u_{vs}(0) = 0 \tag{7.8}$$

The overall control structure can be seen in figure 7.1.

7.2 Gain Calculation using Optimal Control Theory

The matrix gain $\mathbf{K}$ must then be chosen in a such way that the jerk of the quadrotor can be limited. In order to achieve this, LQR[1] optimal control method is used. The following performance criterion is minimized in order to find suitable feedback gains :

$$J = \frac{1}{2}\int_0^\infty (\mathbf{x}_{aug}^\top \mathbf{Q}\mathbf{x}_{aug} + u_{vs}^\top R u_{vs})\mathrm{d}t \tag{7.9}$$

1. Linear Quadratic Regulator.

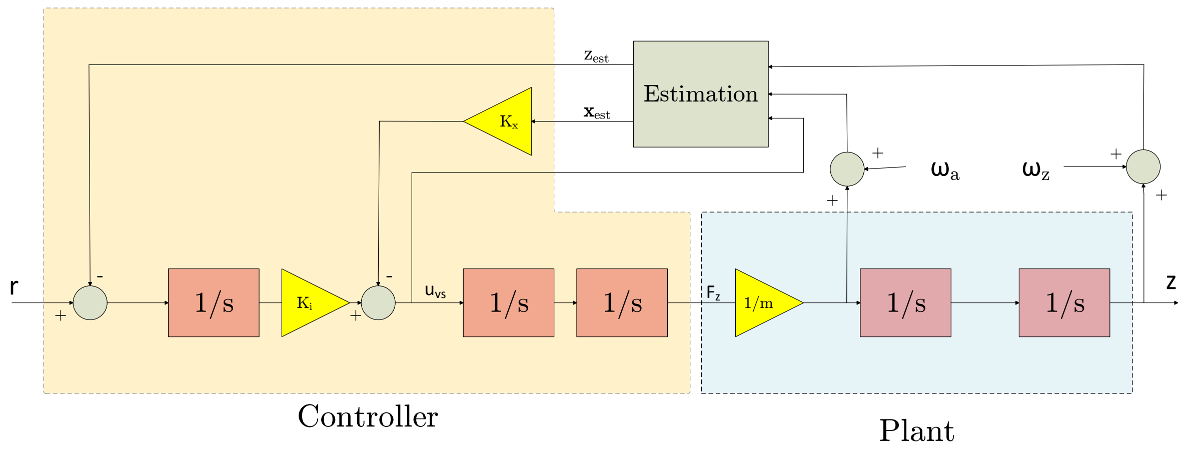

Figure 7.1 Jerk-augmented Control Structure

where $\mathbf{Q} \geq 0$ is a positive semi-definite matrix corresponding to a weighting on the augmented state variables and $R > 0$ is a positive scalar representing a weighting on the control input u_{vs}. With the augmented state vector $\mathbf{x}_{aug}$, the acceleration as well as the jerk experienced by the quadrotor can be explicitly used as design parameters. The resulting gains are calculated by solving the algebraic Riccati equation :

$$\mathbf{PA} + \mathbf{A}^{\top}\mathbf{P} - \mathbf{PB}R^{-1}\mathbf{B}^{\top}\mathbf{P} + \mathbf{Q} = \mathbf{0} \tag{7.10}$$

with $\mathbf{P}$ is a positive definite matrix. The feedback gain matrix $\mathbf{K}$ is then given by

$$\mathbf{K} = R^{-1}\mathbf{B}^{\top}\mathbf{P} \tag{7.11}$$

Using this technique, a desired system response can be achieved in order to actively punish high levels of jerk. For instance, figure **??** shows the simulated step response for two controllers. The first controller is a state-feedback controller that only uses feedback of the position and velocity of the quadrotor. The second is a jerk-augmented controller that was tuned using the techniques presented in this section. Both are tuned to give an overdamped response with settling time of 6 seconds. The first plot of figure **??** shows the position response of each controller configuration. The experienced jerk as well as the integral of the square of the jerk are presented in the second and third plots respectively. What becomes clear is that for the same settling time, the jerk-augmented controller has the capacity to produce significantly lower values of jerk.

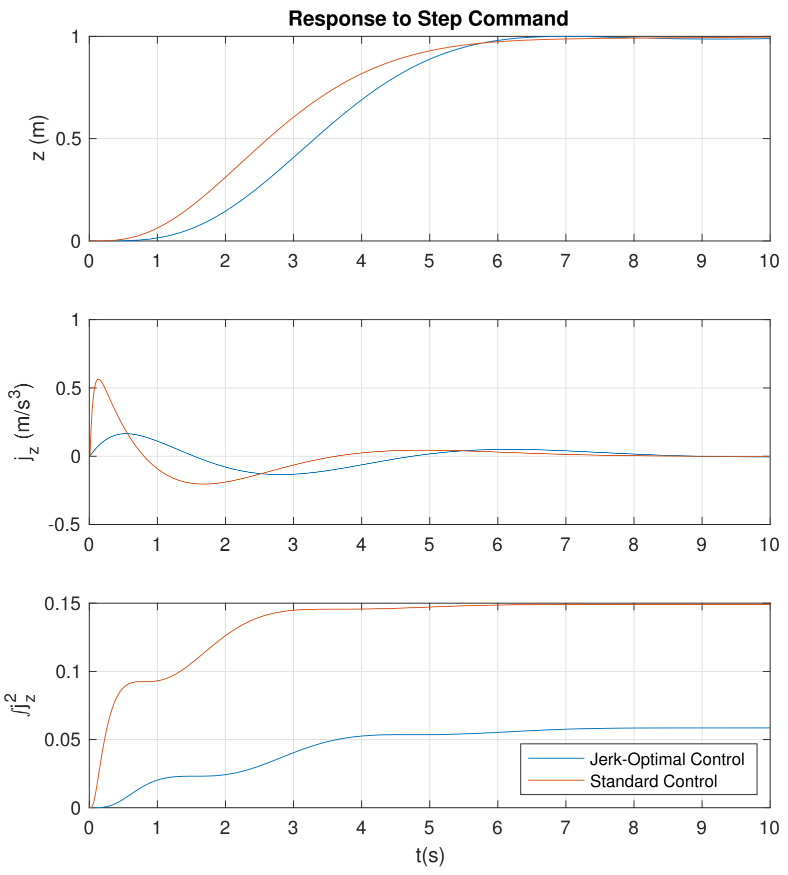

Figure 7.2 Simulated Response to Step Input

7.3 Jerk-Augmented Control in the x- and y-Axis

The controller developed in sections 7.1 and 7.2 can also be used to limit the translational jerk in the x and y directions. To do so, the reduced lateral and longitudinal models of the quadrotor are examined. The reduced longitudinal model is presented in equation 7.12.

$$\begin{bmatrix} \Delta\dot{x} \\ \Delta\dot{v}_x \\ \Delta\dot{\theta} \\ \Delta\dot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta v_x \\ \Delta\theta \\ \Delta q \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1/I_{yy} \end{bmatrix} M_y \tag{7.12}$$

Similarly, The lateral reduced model can be expressed as in equation 7.13 :

$$\begin{bmatrix} \Delta\dot{y} \\ \Delta\dot{v}_y \\ \Delta\dot{\phi} \\ \Delta\dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta v_y \\ \Delta\phi \\ \Delta p \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1/I_{xx} \end{bmatrix} M_x \tag{7.13}$$

In the linearized space, the x and y positions of the quadrotor are indirectly related to the input moments M_y and M_x respectively. In practice, position control and attitude control are often treated as separate problems. This is due to the availability of orientation measurements at a much higher frequency than the off-board position measurements. Consequently, the reduced x and y models can be further separated into attitude and position components. Equations 7.14 and 7.15 demonstrate this with the longitudinal model.

$$\begin{bmatrix} \Delta\dot{x} \\ \Delta\dot{v}_x \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta v_x \end{bmatrix} + \begin{bmatrix} 0 \\ -g \end{bmatrix} \Delta\theta, \tag{7.14}$$

$$\begin{bmatrix} \Delta\dot{\theta} \\ \Delta\dot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\theta \\ \Delta q \end{bmatrix} + \begin{bmatrix} 0 \\ 1/I_{yy} \end{bmatrix} \Delta M_y \tag{7.15}$$

The system in equation 7.15 can be tuned to have a desired step response so that a reference angle can be achieved according to a desired dynamic. This behaviour can then be modeled as an actuator dynamic when considering the system in equation 7.14. Taking this into account, a virtual snap command in the x direction can be defined :

$$u_{vs} \equiv \frac{\mathrm{d}^2}{\mathrm{d}t^2}\left(-g\Delta\theta\right) \tag{7.16}$$

The separated longitudinal model is then augmented to include the virtual snap command :

$$\begin{bmatrix} \Delta\dot{x} \\ \Delta\dot{v}_x \\ \Delta\dot{a}_x \\ \Delta\dot{j}_x \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta v_x \\ \Delta a_x \\ \Delta j_x \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_{vs} \tag{7.17}$$

and finally, as in the case with height control :

$$\Delta\theta = \iint \frac{u_{vs}}{-g}\,\mathrm{d}t \tag{7.18}$$

$$\theta(0) = \theta_e = 0 \tag{7.19}$$

$$\dot{\theta}(0) = \ddot{\theta}(0) \equiv u_{vs}(0) = 0 \tag{7.20}$$

The identical method can then be applied for control in the y direction. The result is a controller that can actively limit the translational jerk of a multirotor UAV in all directions. It is structured in such a way that it can be easily implemented with remote-control type quadrotors such as the Crazyflie 2.0.

7.4 Test Results

Testing the developed controller and estimator proved to be a complex undertaking. This is because it relied on several co-dependent components that had yet to be validated individually. Results in chapter 3 showed that estimation of the jerk was at least possible in principle, but the most complete version of the estimator that incorporated the control input had yet to be implemented. The jerk-augmented controller performed adequately in simulation, but was was shown to be affected by measurement noise. The most difficult practical obstacle in testing the proposed estimator and controller was finding the correct parameters for the real-time jerk estimator. Nevertheless, the proposed solution was successfully implemented in the MRASL. The details of the performed tests are presented in this section.

The controllers were then tuned for the performed tests. To control the height of the Crazyflie 2.0, the jerk-optimal controller was implemented in Simulink and developed for discrete real-time operation. For the purpose of the initial tests, the optimal control $\mathbf{Q}$ and R matrices were chosen as :

$$\mathbf{Q} = \mathrm{diag}(6, 15, 0, 0.1, 6) \tag{7.21}$$

$$R = 1 \tag{7.22}$$

which generated the following state-feedback gains :

$$\mathbf{K} = \begin{bmatrix} 5.7382 & 6.1429 & 2.3634 & 0.4845 & -2.2632 \end{bmatrix} \tag{7.23}$$

In order to avoid drift in the x-y plane in the initial tests, hover controllers developed as in chapter 3 were used to set the desired yaw, pitch and roll angles.

7.4.1 Height Response to Reference Input

The first tests were performed to demonstrate the feasibility of the developed components. The jerk-augmented controller was only implemented to control the height of the quadrotor, while standard state-feedback controllers were used to eliminate drift in the x-y plane. After a takeoff period to distance the vehicle from areas where ground effect was significant, the quadrotor was commanded to track a reference input height of 2 m, as shown in figure 7.3.

The first subplot of figure 7.3 shows the measured height of the quadrotor alongside the expected simulation result and bounded least-squares fit. All three curves follow each other very closely. The similarity between the measured height and the simulation curve validates the overall controller structure and demonstrates the applicability of such a controller in real-world uses. The quality of the least-squares fit ensures that its derivatives will produce precise offline estimates of the Crazyflie's velocity, acceleration and jerk.

The second subplot shows the real-time estimate of the velocity of the quadrotor with the first-derivative of the offline curve. Except for some noise in the real-time estimation, the two curves show almost identical behaviour. We conclude that the real-time velocity estimate is more than adequate.

The real-time estimate, second derivative of the least-squares curve and accelerometer data are presented in the third subplot of figure 7.3. The real-time estimate, presented in red, followed the offline estimate relatively well. Even though a significant amount of noise manifested itself in the real-time estimate, it was at substantially lower levels than if the raw accelerometer data was used directly. This shows an improvement over a direct IMU approach to measuring acceleration in real-time.

The fourth subplot shows an initial attempt to show the validity of the real-time estimate of the jerk. The third derivative of the offline curve is superimposed on the real time estimate and simulation values to a step input. Unfortunately, the signal to noise ratio of the jerk signal in this version of the test is too low to make a conclusion with respect to the validity of the estimator for real-time values of the jerk. However, when comparing the offline curve to the simulation curve, the implemented controller generated lower values of jerk than expected in simulation. This shows that the controller still has the capacity to limit the jerk of a quadrotor. To further investigate and show the quality of the real-time jerk estimator, a different test configuration must be used.

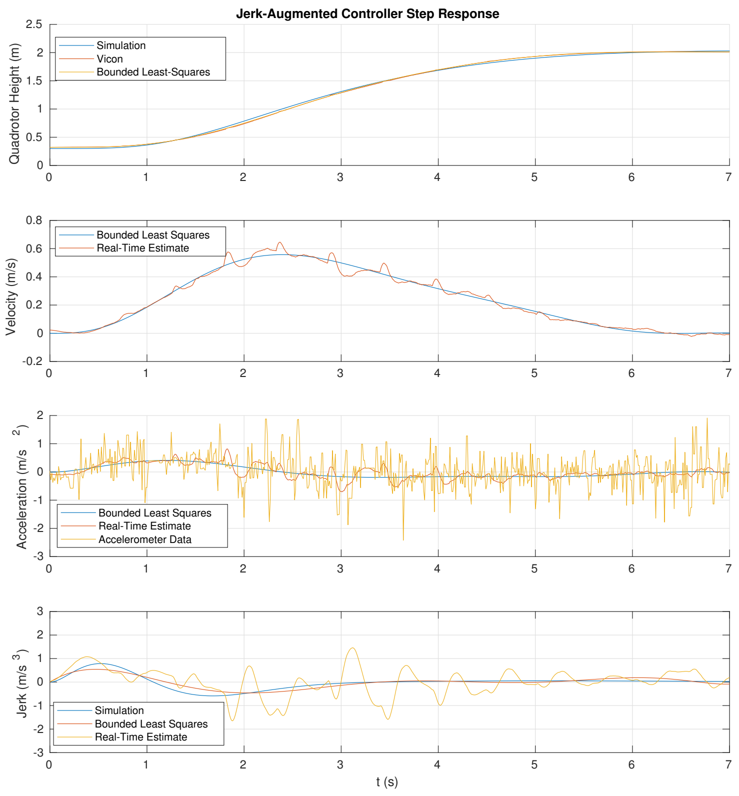

Figure 7.3 Test Response to Step Input

7.4.2 Tests in the x-y Plane

A validation of the real-time estimate of the jerk generated inconclusive results because the levels of jerk experienced by the Crazyflie 2.0 were so low that they fell below a level that could be evaluated visually. As a result, an experimental setup was produced in order to produce artificially high levels of jerk. The jerk-augmented controller was first implemented for control in the x-y plane and tuned to react quickly with no limitation of the jerk. To do so, the optimal control $\mathbf{Q}$ and R matrices were chosen as in equation 7.24. Because of the symmetry of the reduced lateral and longitudinal models, the same weighting parameters were used :

$$\mathbf{Q} = \text{diag}(6, 0, 0, 0, 9) \tag{7.24}$$

$$R = 10 \tag{7.25}$$

which corresponds to a weighting only on the position and integral error of position. The vehicle's x-y velocities, accelerations and jerk are ignored in the performance criterion. This generated the following state-feedback gains for control in the x direction :

$$\mathbf{K} = \begin{bmatrix} -2.1824 & -2.2697 & -1.4060 & -0.5333 & 0.9239 \end{bmatrix} \tag{7.26}$$

for the y direction, the calculated gains were simply the negative values of the above gains. The on-board attitude controller as used in chapter 3 was implemented to receive desired angle commands. The controller was set to give a second order response with double poles at -20. A reference input was created and sent to follow a Lissajou curve defined as follows :

$$\begin{aligned} x &= \sin(t) \\ y &= \cos(2t) \\ z &= 1 \end{aligned} \tag{7.27}$$

As this is not a trajectory tracking controller, the quadrotor was not expected to follow the curve exactly. The goal was to make it change direction quickly enough to generate measurable levels of jerk. The behaviour of the y position of the Crazyflie can be seen in figure 7.4.

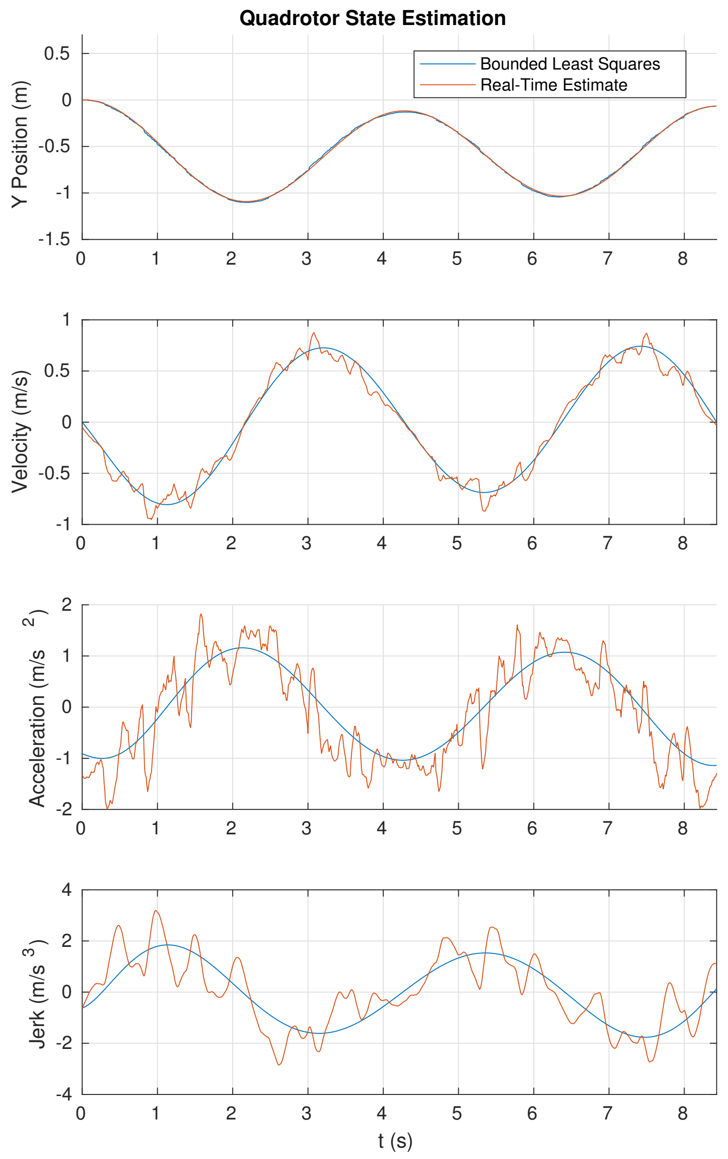

Figure 7.4 Test Response to Step Input

As in the step response test, the position data was collected and fit with a least-squares polynomial to validate the estimation techniques that were used. Because of the sinusoidal nature of the generated reference input, the quadrotor did indeed change direction at higher velocities and frequencies. The offline approximation of the experienced velocity of the quadrotor in the y direction can be seen to be closely followed by the real-time estimate. For the acceleration, the estimator had a similar performance to the step input case. The main difference seen in this test configuration is in the subplot representing the jerk of the quadrotor in the y axis. The offline curve shows a distinctly sinusoidal behaviour at higher magnitudes than in the step input case. At these higher levels of jerk, the performance of the real-time estimator is revealed to successfully track the actual values of the jerk with some significant levels of noise. However, even with this noise, the quadrotor remained stable and was able to respond adequately to a changing reference input. We conclude that even though the level of noise in the jerk and acceleration signals was relatively high, the real-time jerk estimation using an LKF produced satisfactory and usable results. Implementing these techniques with sensors that are more isolated from vibration could drastically improve the precision of resulting state-estimates.

7.4.3 Comparison of Jerk Levels

One of the main purposes of the jerk-augmented controller developed in this chapter is to limit the levels of jerk of a quadrotor. In order to validate this property, another set of tests was performed with the goal of reproducing simulation results shown in figure 7.2. Two controllers were tuned to have a settling time of close to 5 seconds. The first controller followed a standard state-feedback architecture and was tuned using LQR techniques. The second was a jerk-augmented controller tuned as in section 7.1. The quadrotor was then commanded to track a height of 1.5m, producing results as shown in figure 7.5 and 7.6.

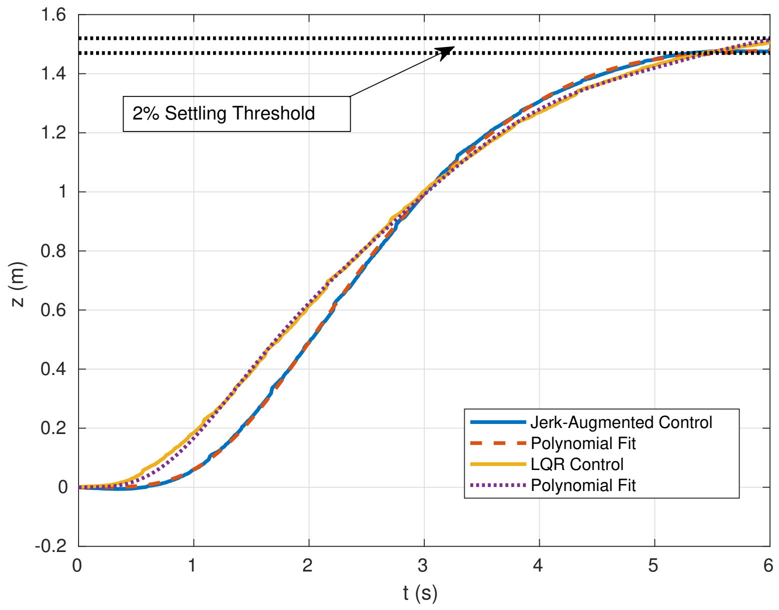

Figure 7.5 Position Response Comparison

Figure 7.5 shows the position response to a reference input of 1.5m. Both LQR and jerk-augmented controllers cross into the 2 percent settling threshold at similar times, with the jerk augmented controller reaching slightly before the LQR controller. This implies that in terms of settling time, these two controllers perform at equivalent levels. The true difference between these controllers becomes apparent when analyzing the jerk of the quadrotor over this period.

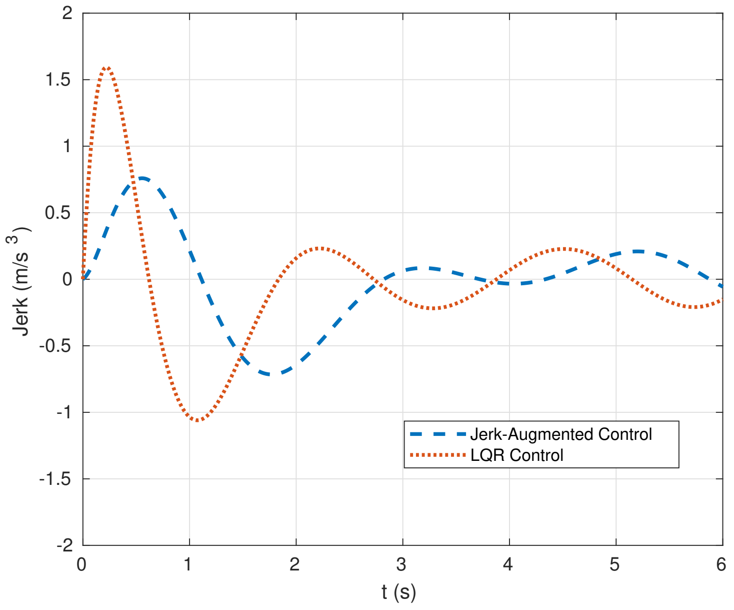

Figure 7.6 Jerk Response Comparison

Figure 7.6 Shows the calculated offline curves corresponding to the levels of jerk experienced by both controllers. The jerk of the reference-tracking trajectory produced by the LQR controller exceeded $1.5m/s^3$ while the jerk-augmented controller limited levels of jerk to well below $1m/s^3$. This shows the jerk-augmented controller's ability to limit a quadrotor's jerk while respecting constraints corresponding to settling time. This result is promising and could lead to potential future applications where limitation of the jerk is necessary.

CHAPTER 8 CONCLUSION

8.1 Synthesis of Work

This work aimed to explore the field of trajectory generation and control. Firstly, a reproduction of the important work presented in [1] was conducted in the MRASL. Results showed that the theoretically simple approach used enabled the reproducible and precise execution of aggressive manoeuvres through windows. The tests also demonstrated the capabilities of the MRASL's flight arena as well as the potential of the Crazyflie 2.0 quadrotor used.

The remainder of this work contains four main contributions. The first is the development and experimental validation of a real-time jerk estimator for quadrotors. Of the multiple possible solutions presented, the most effective implementation of real-time jerk estimation was a linear Kalman filter that made use of position and acceleration measurements as well as input commands. The result was an estimator that accurately tracked the jerk of a quadrotor UAV in real-time with a precision that was high enough to be used for state-feedback. The second contribution is an offline estimation technique by bounded least-squares curve-fitting that gives a reasonable approximation of the actual jerk experienced by a vehicle. By setting reasonable boundary conditions of a high-order polynomial curve, this method proved to be a reasonably precise way to evaluate an otherwise unmeasurable quantity. The third contribution is a jerk-augmented LQR controller that can explicitly limit the jerk of the height of a quadrotor. This controller was combined with the estimator in order to show the possible practical applications of both elements. Using simulations that included realistic values of measurement noise and actuator response, the controller showed promising results.

The final and most important contribution of this work was the validation of the estimator and controller during experimental tests. Several works in the field of trajectory generation and control have produced control architectures that require the feedback of the jerk of a UAV in order to be stable. However, no work could be found that validates these architectures with testing. To the best of our knowledge, the experimental validation presented in this work is the first time that a feedback of the jerk has been used for the control of a quadrotor UAV.

8.2 Limitations of the Proposed Solution

Some of the limitations of the solution treated in chapter 5 are well-documented. The main drawback of the sequential composition technique used is the need for a multi-trial iterative refinement process in order to ensure the accuracy of the aggressive window manoeuvres.

A second limitation of the sequential composition method is that it offers no guarantee of the quadrotor's ability to stabilize after passing through the window. Different values for the overall thrust delivered during the attitude control phase as well as different gain values for the soft hover controller must be simulated and tested in order to come up with a reliably stable end phase. Augmenting the overall thrust decreased the precision of the manoeuvres while decreasing the overall thrust made it more difficult for the quadrotor to stabilize. In terms of the gains of the soft-hover controller, a similar behaviour was observed. Tuning the gains to have a short settling time caused a saturation of the motors, while "softening" the controller to have a longer settling time caused the quadrotor to drop closer to the ground before stabilizing. If the controller's settling time was too long, the quadrotor would hit the ground before stabilizing.

The main limitation of the proposed jerk estimation techniques was its basis in the linear approximation of the quadrotor model. In operations far outside of this linearization, the estimator would fail to accurately estimate the jerk experienced by the quadrotor. However, this simple estimator structure was chosen because of the limited availability of experimental evaluation of jerk estimation techniques. This estimator was intended to be a proof of concept for future applications of jerk feedback.

Both the estimator and jerk-augmented controller were shown to be somewhat sensitive to accelerometer noise. The operation of the UAV used during experiments was therefore limited by the high level of measurement noise (variances on the order of $\sigma^2 = 0.3$) of the on-board accelerometers. In order improve the behaviour of the controller, and to enable the estimator to place a higher weighting on the acceleration measurements, it would be necessary to use a quadrotor with vibration isolation of the on-board IMU.

8.3 Future Improvements

Future work could focus on the implementation of control elements presented in chapters 6-7 towards the execution of the manoeuvres presented in chapter 5. An example of this could include the calculation of feasible minimal-snap trajectories as presented in [23] and their execution by a feedback-linearization controller as presented in [54]. The latter work is a nonlinear controller that requires a feedback of the jerk in order to be stable and guarantees the convergence of a quadrotor onto a sufficiently smooth 3D trajectory. These proposed improvements could eliminate the need for the iterative process outlined in [1].

REFERENCES

[1] Daniel Mellinger, Vijay Kumar, and Nathan Michael. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *International Journal of Robotics Research*, 2012.

[2] Carlos Luis and Jerome Le Ny. Design of a trajectory tracking controller for a nano-quadcopter. *CoRR*, abs/1608.05786, 2016.

[3] Shin-Ichi Nakazawa, Tadashi Ishihara, and Hikaru Inooka. Real-time algorithms for estimating jerk signals from noisy acceleration data. *International Journal of Applied Electromagnetics and Mechanics*, 18 :149–163, 2003.

[4] Robert Mahony, Kumar Vijay, and Peter Corke. Multirotor aerial vehicles. *IEEE ROBOTICS & AUTOMATION MAGAZINE*, pages 20–32, September 2012.

[5] Markus Hehn and Raffaello D'Andrea. A flying inverted pendulum. In *2011 IEEE International Conference on Robotics and Automation*, pages 763–770, May 2011.

[6] Kostas Alexis, George Nikolakopoulos, Anthony Tzes, and Leonidas Dritsas. *Coordination of Helicopter UAVs for Aerial Forest-Fire Surveillance*, pages 169–193. Springer Netherlands, Dordrecht, 2009.

[7] Alexandre Borowczyk, Duc Tien Nguyen, André Phu-Van Nguyen, Dang Quang Nguyen, David Saussié, and Jerome Le Ny. Autonomous Landing of a Multirotor Micro Air Vehicle on a High Velocity Ground Vehicle. *IFAC-PapersOnLine*, 50(1) :10488–10494, 2017.

[8] Richard C. Dorf and Robert Bishop. *Modern Control Systems.* Prentice Hall, Upper Saddle River, New Jersey, 12th edition, 2011.

[9] Wilson J. Rugh. *Linear System Control.* Prentice Hall, Upper Saddle River, New Jersey, 2nd edition, 1996.

[10] Hassan K. Khalil. *Nonlinear Systems.* Prentice Hall, Upper Saddle River, New Jersey, 2nd edition, 1996.

[11] Chi-Tsong Chen. *Analog and Digital Control System Design.* Oxford University Press, New York, New York, 2006.

[12] Duc Tien Nguyen, David Saussié, and Lahcen Saydy. Robust Self-Scheduled Fault-Tolerant Control of a Quadrotor UAV. *IFAC-PapersOnLine*, 50(1) :5761–5767, 2017.

[13] Eitan Bulka Tran, Nguyen Khoi and Meyer Nahon. Quadrotor control in a wind field. In *IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015.

[14] Holger Voos. Nonlinear control of a quadrotor micro-uav using feedback-linearization. In *IEEE International Conference on Mechatronics*, 2009.

[15] Samir Bouabdallah and Roland Siegwart. Backstepping andsliding-mode techniques applied to an indoor micro quadrotor. In *IEEE International Conference on Robotics and Automation*, 2005.

[16] Federico Augugliaro, Emanuele Zarfati, Ammar Mirjan, and Raffaello D Andrea. Knot-tying with Flying Machines for Aerial Construction. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 5917–5922, 2015.

[17] Ammar Mirjan, Federico Augugliaro, Raffaello D'Andrea, Fabio Gramazio, and Matthias Kohler. Building a bridge with flying robots. In D. Reinhardt, R. Saunders, and J. Burry, editors, *Robotic Fabrication in Architecture, Art and Design*. Springer,Cham, 2016.

[18] Hyunsoo Yang and Dongjun Lee. Dynamics and Control of Quadrotor with Robotic Manipulator. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5544–5549, 2014.

[19] Benoit Landry. Planning and control for quadrotor flight through cluttered environments. Master's thesis, Massachussets Institute of Technology, 2014.

[20] William Hanna. Modelling and control of an unmanned aerial vehicle. Bachelor's Thesis, Charles Darwin University, 2014.

[21] Julian Forster. System Identification of the Crazyflie 2.0 Nano Quadrocopter. Bachelor's Thesis, ETH Zurich, 2015.

[22] Gabriel M. Hoffmann, Steven L. Waslander, and Claire J. Tomlin. Quadrotor helicopter trajectory tracking control. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

[23] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 3509215, pages 2520–2525, 2011.

[24] Justin Thomas, Giuseppe Loianno, Morgan Pope, Elliot W. Hawkes, Matthew A. Estrada, Hao Jiang, Mark R. Cutkosky, and Vijay Kumar. Planning and control of aggressive maneuvers for perching on inclined and vertical surfaces. In *ASME International Design Engineering Technical Conferences*, 2015.

[25] Taeyoung Lee, Melvin Leoky, and N. Harris McClamroch. Geometric tracking control of a quadrotor uav on se(3). In *Proceedings - IEEE Conference on Decision and Control*, 2010.

[26] Markus W. Achtelik, Simon Lynen, Margarita Chli, and Roland Siegwart. Inversion based direct position control and trajectory following for micro aerial vehicles. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2933–2939, Nov 2013.

[27] Marcelino M De Almeida and Maruthi Akella. New numerically stable solutions for minimum-snap quadcopter aggressive maneuvers. In *2017 American Control Conference*, 2017.

[28] Davide Falanga, Elias Mueggler, Matthias Faessler, and Davide Scaramuzza. Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision. In *Proceedings - IEEE International Conference on Robotics and Automation*, 2017.

[29] Mark Mueller, Markus Hehn, and Raffaello D'Andrea. A computationally efficient motion primitive for quadrocopter trajectory generation. *IEEE Transactions on Robotics*, 31(6), 2015.

[30] Jing Yu, Zhihao Cai, and Yingxun Wang. The minimum jerk trajectory generation of a quadrotor based on the differential flatness. In *IEEE Chinese Guidance, Navigation and Control Conference*, pages 1061–1066, 2016.

[31] T Rakgowa, Eng Kiong Wong, Kok Swee Sim, and M E Nia. Minimal Jerk Trajectory For Quadrotor VTOL Procedure. In *IEEE International Symposium on Robotics and Intelligent Sensors*, pages 284–287, 2015.

[32] Michael Neunert, Cedric De Crousaz, Fadri Furrer, Mina Kamel, Farbod Farshidian, Roland Siegwart, and Jonas Buchli. Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking. In *Proceedings - IEEE International Conference on Robotics and Automation*, 2016.

[33] Sarah Tang and W Valentin. Aggressive Flight With Suspended Payloads Using Vision-Based Control. *IEEE Robotics and Automation Letters*, 3(2) :1152–1159, 2018.

[34] Cameron C Taylor and Jacobus A. A. Engelbrecht. Acceleration-based Control of a Quadrotor with a Swinging Payload. In *Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)*, 2016.

[35] Philipp Foehn, Davide Falanga, Naveen Kuppuswamy, Russ Tedrake, and Davide Scaramuzza. Fast Trajectory Optimization for Agile Quadrotor Maneuvers with a Cable-Suspended Payload. In *Proc. Robot., Sci. Syst*, Boston, Ma, 2017.

[36] Ivana Palunko, Rafael Fierro, and Patricio Cruz. Trajectory Generation for Swing-Free Maneuvers of a Quadrotor with Suspended Payload : A Dynamic Programming

Approach. In *IEEE International Conference on Robotics and Automation*, pages 2691–2697, 2012.

[37] Gokhan Alcan and Mustafa Unel. Robust Hovering Control of a Quadrotor Using Acceleration Feedback. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017.

[38] Hugh Durrant-Whyte and Thomas C. Henderson. *Multisensor Data Fusion*, pages 585–610. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[39] Shaohua Wang and Ying Yang. Quadrotor aircraft attitude estimation and control based on Kalman filter. In *Proceedings of the 31st Chinese Control Conference*, volume 1, pages 5634–5639, 2012.

[40] Matías Tailanián, Santiago Paternain, Rodrigo Rosa, and Rafael Canetti. Design and implementation of sensor data fusion for an autonomous quadrotor. In *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, pages 1431–1436, 2014.

[41] Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor. *Robotics : Science and Systems*, 2013.

[42] Robert C. Leishman, John C. MacDonald, Randal W. Beard, and Timothy W. McLain. Quadrotors and accelerometers : State estimation with an improved dynamic model. *IEEE Control Systems*, 34(1) :28–41, 2014.

[43] Vps Naidu. Evaluation of Acceleration and Jerk Models in Radar and IRST Data Fusion for Tracking Evasive Maneuvering Target. In *AIAA Aerospace Sciences Meeting and Exhibit*, 2008.

[44] Andrea Bisoffi, Francesco Biral, Mauro Da Lio, and Luca Zaccarian. Longitudinal Jerk Estimation of Driver Intentions for Advanced Driver Assistance Systems. *IEEE/ASME Transactions on Mechatronics*, 2017.

[45] Mohammad Ali Badamchizadeh, Iraj Hassanzadeh, and Mehdi Abedinpour Fallah. Extended and unscented kalman filtering applied to a flexible-joint robot with jerk estimation. *Discrete Dynamics in Nature and Society*, 2010.

[46] Makoto Yamakado, Jyunya Takahashi, Shinjiro Saito, Atsushi Yokoyama, and Masato Abe. Improvement in vehicle agility and stability by G-Vectoring control. *Vehicle System Dynamics*, 48(SUPPL. 1) :231–254, 2010.

[47] John-Jairo Jairo Martinez and Carlos Canudas-de Wit. A safe longitudinal control for adaptive cruise control and stop-and-go scenarios. *IEEE Transactions on Control Systems Technology*, 15(2) :246–258, 2007.

[48] Abdelaziz Benallegue, Abd El-Kader Mokhtari, and Leonid Fridman. High-order sliding-mode observer for a quadrotor UAV. *International Journal of Robust and Nonlinear Control*, 2006.

[49] Milad Bayat and M A Amiri Atashgah. An Augmented Strapdown Inertial Navigation System using Jerk and Jounce of Motion for a Flying Robot. *The Journal of Navigation*, 70 :907–926, 2017.

[50] Chiu Liu and Thomas W Kennedy. Human judgment and analytical derivation of ride quality. *Transportation Science*, 33(3) :290–297, 1999.

[51] Manikandasriram Srinivasan Ramanagopal, André Phu Van Nguyen, and Jerome Le Ny. A Motion Planning Strategy for the Active Vision-Based Mapping of Ground-Level Structures. *IEEE Transactions on Automation Science and Engineering*, 15(1) :356–368, 2018.

[52] Jun Ichi Toji and Hiroyuki Ichihara. Formation control of quadrotors with extended feedback linearization based on consensus problem. In *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, pages 3477–3480, 2017.

[53] Abd El-Kader Mokhtari, Abdelaziz Benallegue, and Yury Orlov. Exact linearization and sliding mode observer for a quadrotor unmanned aerial vehicle. *International Journal of Robotics and Automation*, 21(1) :39–49, 2006.

[54] Dong Eui Chang and Yongsoon Eun. Global Chartwise Feedback Linearization of the Quadcopter with a Thrust Positivity Preserving Dynamic Extension. *IEEE Transactions on Automatic Control*, 2017.