McGill University

Master's Thesis

Attitude Estimation for Collision Recovery of a Quadcopter Unmanned Aerial Vehicle

---

Author:
Adrian Battiston

Supervisors:
Dr. Inna Sharf
Dr. Meyer Nahon

A thesis submitted in fulfillment of the requirements
for the degree of Master's of Engineering
in the

Aerospace Mechatronics Lab
Department of Mechanical Engineering

November 27, 2017

Abstract

The increased use of unmanned aerial vehicles (UAVs) in the public and private domains requires increased emphasis on their safe use. Recent research in the Aerospace Mechatronics Lab (AML) at McGill has worked towards a collision recovery system for quadcopter UAVs in order to enable the safe, autonomous recovery of the vehicle after a collision has occurred. This thesis continues the design of a complete collision recovery control system by investigating the effects of a collision on a quadcopter's Inertial Measurement Unit (IMU) sensors and through a comparison of conventional and novel attitude estimation algorithms with the goal of providing an improved attitude estimate during a collision. An approach to modify simulated IMU data to match the effects of a quadcopter collision on experimental data is developed. This model is then used to validate attitude estimation algorithms in simulation, prior to their evaluation using experimental data. The algorithms under investigation are a Multiplicative Extended Kalman Filter (MEKF), an Unscented Kalman Filter (UKF), a complementary filter, an H_∞ Filter, and novel adaptive varieties of the selected filters. The novel adaptive estimation algorithms are derived to better estimate the attitude during a collision. The algorithms are compared in simulated normal flight as well as during a simulated collision of a quadcopter with a wall to show which estimation algorithm provides the best quadcoptor attitude estimate in all conditions. The estimation algorithms are then evaluated experimentally, first by comparing attitude estimates generated using post-processed sensor data, then, the two estimators with the best performance are implemented onboard the quadcopter. Results show that there are observability issues experienced along the yaw axis due to the sensors available, causing degradation in gyroscope bias and yaw angle estimation. The adaptive algorithms offered some very small improvements when compared to the conventional algorithms and the robustness of H_∞ filter caused it to be detrimental to performance in this scenario. The MEKF and adaptive H_∞ filter were implemented in the collision recovery control loop and garnered an improved attitude estimate after a collision when compared to the generic algorithm which comes with the flight controller software.

Abrégé

L'utilisation augmenté des véhicules aériens sans pilote (UAVs) dans les domaines public et privé nécessite un accent accru sur leur utilisation sûre. Des recherches récentes dans le Aerospace Mechatronics Lab (AML) à McGill ont travaillé vers un système de récupération de collision pour les UAV quadrirotor afin de permettre une récupération autonome après une collision. Cette thèse poursuit la conception d'un système complet de contrôle de la récupération de collision en étudiant les effets d'une collision sur l'unité de mesure inertielle (IMU) d'un quadrirotor et par une comparaison d'algorithmes d'estimation d'attitude conventionnels et nouveaux dans le but de fournir une estimation d'attitude améliorée au cours d'une collision. Une approche pour modifier les données IMU simulées pour faire correspondre les effets d'une collision est développée. Ce modèle est utilisé pour valider les algorithmes d'estimation d'attitude en simulation avant leur évaluation à l'aide de données expérimentales. Les algorithmes à l'étude sont un filtre multiplicatif de Kalman étendu (MEKF), un filtre Kalman "unscented" (UKF), un filtre complémentaire, un filtre H_∞ et des variétés nouveau adaptables des filtres sélectionnés. Les nouveaux algorithmes d'estimation adaptative sont dérivés pour mieux estimer l'attitude lors d'une collision. Les algorithmes sont comparés dans le vol normal simulé ainsi que lors d'une collision simulée d'un quadrirotor avec un mur pour montrer quel algorithme d'estimation fournit la meilleure estimation d'attitude de quadrirotor dans toutes les conditions. Les algorithmes d'estimation sont ensuite évalués expérimentalement, d'abord en comparant les estimations d'attitude générées à l'aide de données de capteurs recemment acru, puis les deux estimateurs ayant les meilleures performances sont implémentés à bord du quadrirotor. Les résultats montrent qu'il existe des problèmes d'observabilité rencontrés le long de l'axe du lacet en raison des capteurs disponibles, ce qui provoque une dégradation dans l'estimation du biais du gyroscope et de l'angle de lacet. Les algorithmes adaptatifs ont offert des améliorations très faibles par rapport aux algorithmes classiques et la robustesse du filtre H_∞ causent une dégradation de la performance dans ce scénario. Le MEKF et le filtre adaptatif H_∞ ont été implémentés dans la boucle de contrôle et ont obtenu une estimation d'attitude améliorée après une collision par rapport à l'algorithme générique fourni avec le logiciel de contrôleur de vol.

Acknowledgements

First and foremost I would like to thank my supervisors, Professor Inna Sharf and Professor Meyer Nahon. Their guidance and the opportunities they have given me have been invaluable. Their experience never failed to enlighten my mind and lighten my workload.

This research was fuelled by two previous students in the AML, Fiona Chui and Gareth Dicker, whose enjoyable company and help made my work fly by. A special thanks is deserved by Gareth for his help as a pilot, and his endless supply of deep philosophical questions.

A big thanks to the other previous students in the AML, Khoi Tran, Waqas Khan, and Patrick Abouzakhm, who were always helpful in answering my pestering questions. Also a huge thanks to the current students in the AML, Eitan Bulka, Bassam Ul Haq, Josh Levin and Luc Sagniéres, who helped distract me and would sometimes laugh at my jokes. As well a thanks to Raja Mukherji, Adam Phillip and Professor James Forbes, for getting me started in control systems and providing guidance throughout.

Most of all, I'd like to thank my parents, for always talking me down from dropping out of school, from the first time they did it in grade three, to basically every year of my university career. My brothers, Nick and Stefan, for being inspirations, and friends through it all. My hometown brothers, Quinn, Daniel and Andrew, for keeping me level and never missing a beat. My roommates, Chris and Jay, for making our home for the past 7 years as beautiful a place as anyone's grandma could. And the rest of my Montreal family, for the love and friendship throughout this journey.

Another special thanks is deserved to one of my Montreal family, Nicholas Veenhoven, for helping me translate the abstract into French.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recently, Unmanned Aerial Vehicles (UAVs) have become more prevalent in industry and public sectors as new uses for them continue to arise. UAVs find use in search and rescue, photography, visual surveillance and are used by hobby enthusiasts all over the world [1]. The most common types of UAVs are multirotor, fixed wing, and a hybrid of the two.

Multirotor UAVs use a set of propellers to generate vertical thrust, manoeuvre and hover. The most common types of multirotors have four propellers (quadcopter) or six propellers (hexacopter). They tend to have shorter flight times and distances than other varieties of UAVs but their ability to hover and move laterally make them more manoeuvrable. Quadcopters find the majority of their use in photography and surveillance due to their ability to hover in one spot.

Fixed wing UAVs generate lift using wings and control surfaces, using one or more propellers to develop airflow over the wings. They vary in size from those with wingspans similar to medium size passenger planes, to wingspans under half a meter. Fixed wing UAVs generally have much longer flight times as the use of wings and control surfaces is more efficient for generating lift than using propellers directly for lift. They find use in delivery, search and rescue and other applications which require travelling over long distances.

Hybrid UAVs are a combination of both multirotor and fixed wing UAVs. They use the thrust from propellers to take off and land, and also have wings to generate lift while moving horizontally. Their ability to take off vertically and generate lift efficiently for long flights makes them versatile and allows many potential applications. Hybrid designs are usually more complex and will sometimes rotate their propeller housings or wings to facilitate a vertical take off. Due to the complexity of combining both multirotor and fixed wing designs, functioning hybrid style UAVs are not as well

developed as either multirotor or fixed wing UAVs, but many companies are currently working on designs for use in package delivery, medical supply delivery, and humanitarian aid.

With the use of UAVs increasing, whether they are flown autonomously or manually, crashes will occur. As federal laws on the use of UAVs around the public and in regulated airspace evolve, mitigating the effects of collisions is likely to be one of the main issues in legislation [2], [3]. While much research focuses on collision avoidance for UAVs, research is also being carried out on collision recovery [4]. With the goals of increasing public safety and protecting valuable hardware, recent efforts at McGill's Aerospace Mechatronics Lab (AML) have addressed modelling collisions between a quadcopter and environment as well as the design of collision recovery controllers [5], [6]. In general, the goal of a control algorithm is to successfully transition the system from its current state, to a desired state. A system's state can include any information about the system, such as position, orientation, temperature, etc. Control algorithms use an estimate of the current state of the system, along with a desired state to compute the actuation the system needs to achieve this desired state. The majority of conventional control algorithms do not detect collisions, nor are they designed to deal with collisions; therefore, UAV collisions can be catastrophic to the system and dangerous to bystanders. Generally, designing a control system is regarded as a conjoint problem where the overall performance of the system is dependent on the accuracy of the state estimate, and the stability and efficiency of the controller.

As the true state of a system cannot innately be known, an estimate of the state has to be computed using models and measurements of the system and known inputs to the system. Any system which requires knowledge about its state must be equipped with a set of sensors whose measurements can be combined to compute the state. The measurements provided by sensors are not perfect due to noise in the system and inaccuracy in the sensors. Also, sensors do not always provide direct measurements of the states which are useful for control of the system. State estimation algorithms combine all available measurements to obtain the optimal estimate of the system's internal state. Differences in the performance of state estimation algorithms occur as the majority of algorithms make assumptions about the type of noise the sensors experience, and approximate non-linearities in the system. Algorithms which do not make assumptions about sensor noise also exist, however they can be computationally expensive.

The previous work on collision recovery controllers at the AML focused on attitude stabilization

of a quadcopter equipped with safety bumpers or a protective frame following a collision with a wall [6]. In order to facilitate stable attitude control of UAVs, whether during normal flight or following a collision, an accurate estimate of the UAV attitude must be provided to the controller. This thesis focusses on providing an attitude estimate for quadcopters experiencing a collision with a wall.

The design and implementation of an attitude estimation algorithm is dependant on many factors. Quadcopters are highly unstable systems and require the attitude estimator and controller to run at high rates (typically 100 Hz). As quadcopters also require high thrust to weight ratios, they tend to have fairly limited computational resources - most of their weight capacity is dedicated to batteries. Due to these factors, the computational complexity of attitude estimation algorithms is an important issue for their use onboard quadcopters. The non-linearity of attitude estimation also largely influences the implementation of estimation algorithms. Another important factor in determining an effective state estimation algorithm is the sensors available onboard the vehicle. The majority of UAVs come equipped with an Inertial Measurement Unit (IMU) which is composed of a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer. In line with finding a collision recovery solution which can be implemented solely using conventional sensors onboard a quadcopter or other UAV, this thesis looks into the effects of a collision on IMU data and investigates attitude estimation algorithms that have potential to provide improved performance during a collision.

1.1 Literature Review

There are many ways in which the attitude of a vehicle can be defined or parametrized. In order to accurately estimate the attitude of a UAV, an adequate attitude parametrization must be used. The most commonly used methods to represent the attitude of an object in 3D are Euler angles, quaternions and the Direction Cosine Matrix (DCM) [7]. Euler angles are fairly common and easy to understand but experience singularities in certain orientations. As aggressive manoeuvres likely to occur during collision recovery require an attitude representation with no singularities, the use of Euler angles is avoided. Quaternions are a 4-parameter representation of attitude which have no singularities but have a unit norm-constraint that must be enforced. The DCM is a 9-parameter attitude representation which also has no singularities but the DCM must be orthogonal. As orthogonalization of a matrix is more computationally intensive than normalization of a vector, using the quaternion provides better computational efficiency [7]. Since the collision

recovery controller that has been developed in the AML is a quaternion based controller, this review focuses on quaternion based attitude estimation algorithms.

The quaternion norm-constraint and the non-linearity of attitude estimation has spurred the development of a variety of attitude estimationn algorithms [8]. Probably the most popular and widely used algorithm in attitude estimation is the Extended Kalman Filter (EKF). It uses a first order Taylor series expansion of the non-linear state and measurement equations to linearize the system about an operating point (usually the current state of the system) and then applies the Kalman Filter (KF) directly [9]. The extensive use of the EKF stems from the fact that the KF is the best linear unbiased estimator for systems with Gaussian noise. Two modified versions of the EKF have been designed to properly handle the quaternion norm-contraint. Estimating the error quaternion, which linearizes to a 3 component attitude error vector, instead of estimating the regular quaternion, limits error due the quaternion norm-constraint [10]. This version of the EKF is called the Multiplicative EKF (MEKF) as the error quaternion is multiplied by the current quaternion value to update the quaternion estimate, as opposed to the additive method used in a conventional EKF. The other method for preserving the unit norm is the norm-constrained Kalman filter derived by Zanetti et al. [11]. Zanetti et al. use Lagrange multipliers to derive a Kalman filter which enforces a norm-constraint on a subset of the state. The performance of the norm-constrained Kalman filter is shown to improve upon that of the MEKF under certain conditions.

While EKFs provide a first order approximation of a non-linear model, Particle Filters (PF) directly use the non-linear model and estimate a system's state by propagating a random set of state samples through the model in order to generate the state's posterior probability density function, from which the most probable system state can be extracted [12]. Though this technique does not lose any accuracy to linearization, or assumptions about the type of noise in the system, it can be very computationally expensive and is not always suitable for implementations on flight controllers due to their limited computing resources. The Unscented Kalman Filter (UKF) is the middle ground between the EKF and PF. It assumes the system has Gaussian noise and passes a small set of optimally chosen points through the non-linear system, which provides a state estimate accurate up to a second order approximation of the non-linear system [13]. Modified versions of the UKF, which conserve the quaternion norm-constraint by estimating an attitude error vector, instead of the quaternion, or by using Lagrange multipliers, have also been developed [14], [15]. A

version of the UKF which propagates the Cholesky decomposition of the state covariance matrix instead of the full state covariance, called the Square Root UKF (SRUKF), has been shown to increase the numerical stability of the algorithm, as well as decrease its computational complexity [16]. Some EKFs and UKFs are said to have a similar computational complexity of $\mathcal{O}(n^3)$, where n is the dimension of the state vector, but in practice the complexity of UKFs tends to be higher than that of EKFs as UKFs use multiple state samples to compute the state estimate [16]. Comparing the computational complexity of EKFs and UKFs to a PF is difficult as the complexity of a PF depends largely on the number of particles used. For smaller state vectors, fewer particles are generally needed to provide an accurate state estimate, but for larger states the number of particles can increase based on the filter design, resulting in high computational demands [17]. However, it can be said that as a UKF mimics a PF with assumptions that allow an optimally selected number of particles, PFs are more computationally complex than UKFs.

A choice required for attitude estimation algorithms is whether or not to use a dynamics model in the algorithm. Attitude estimation provides an opportunity to avoid the use of a dynamics model by propagating the attitude using a measurement of the angular velocity directly in the attitude kinematics, as opposed to algorithms which use a dynamics model to generate the angular velocity and propagate the quaternion [10]. The use of a dynamics model in an estimation algorithm can sometimes improve estimator performance if an accurate model of the system is available. However, using a dynamics model requires an approximation of the process noise covariance, which can be difficult to obtain. Using a measurement in the attitude kinematics avoids this issue as a sensor's noise covariance is much easier to approximate. Another issue with using a dynamics model is that any inaccuracies in this model could degrade performance [18]. Using a dynamics model would also require significant changes to the estimation algorithm if it were to be portable between different vehicles.

The filtering techniques mentioned above assume a stochastic system and usually provide a state estimate by making assumptions about the system's noise characteristics. The majority of non-linear techniques do not come with proofs of their optimality or stability due to their approximations of the system's non-linearities. State estimation can also be approached using the observer paradigm, which assumes a deterministic system and generally comes with provable asymptotic stability. Non-linear attitude observers have recently become popular methods for attitude estimation due to their

simplicity, small computational cost and guaranteed stability [19]–[21]. One of the more popular non-linear observers due to its efficacy is the complementary filter, named so for its resemblance to a classic complementary filter [19], [22]. The computational complexity of non-linear attitude observers tends to be significantly lower than that of KF based algorithms as non-linear observers do not involve any covariance propagation or multiplication of large matrices.

The choice of an attitude estimation algorithm depends on the computational complexity of the algorithm, the vehicle's sensors, the sensor's noise characteristics and the vehicle's use. To carry out accurate state estimation during a collision, the effects of a collision on quadcopter's sensor data must be quantified. Through examining experimental quadcopter collision data, it was found that flexibility in the quadcopter bumpers and structure causes vibrations throughout the vehicle after an impact. This manifests itself as increased noise or oscillations in the inertial sensors of the quadcopter, which decays as the vibrations are damped. With this foresight, state estimation algorithms which are robust to varying noise characteristics are explored in this thesis.

As the design of Kalman based filters allows for manipulation of the process and measurement noise covariances, adaptive filters which modify the covariance values during a crash are investigated. Sebesta and Boizot [23] derive an adaptive EKF scheme onboard a quadrotor in order to increase sensitivity to large perturbations. Their scheme uses the estimator's innovation term to trigger covariance adaptation in order to put more trust in the sensors as opposed to the dynamics model [23]. Other papers present adaptation schemes that vary the noise covariance matrices based on the innovation in order to detect and adapt to sensor and actuator faults [24]. Qi and Han develop an adaptive UKF that adapts the process noise covariance matrix using a gradient descent technique to match the measured innovation with the predicted innovation [18]. They show that the adaptive algorithm is able to provide a better estimate than the standard UKF when the system model parameters are changed during simulation.

Besides Kalman based estimation techniques which adapt the noise covariance, some filters are designed to be robust to errors and changes in the system parameters. The H_∞ filter is designed to minimize the ratio of the mean square estimation error to the sum of the mean squared initial error and mean squared system noise, providing a guaranteed bound for the H_∞ norm. There are a variety of methods to develop a filter which bounds the H_∞ norm. Xie et al. derive an H_∞ filter using algebraic Riccati equations [25]. Previous Riccati equation H_∞ estimation solutions

were not robust to model parameter errors but Xie et al. represent model uncerainties by a scaling factor making their H_∞ filter more robust. Li and Fu derive an H_∞ filter which uses Linear Matrix Inequalities (LMI) to make their filter robust to both model and noise uncertainties [26]. These derivations of H_∞ filters are very problem specific, as their derivations depend heavily on the system being estimated. The game theory formulation of the H_∞ filter (also dubbed the min-max filter) provides a state-space based filter that can be shown to reduce to a conventional Kalman filter as the H_∞ norm bound is increased towards infinity [9]. Chee and Forbes derive a norm-constrained H_∞ filter for use with attitude estimation, and show its improved performance when faced with errors in the system parameters [27].

One of the more extensive surveys of non-linear attitude estimation techniques is provided by Crassidis et al. [8]. The survey covers a wide variety of estimators and details their advantages. While the survey provides no direct comparison of estimation algorithms in of itself, its exhaustive comparison of papers sums up the general applications and performance of attitude estimation algorithms. It gives evidence that the EKF is a simple, effective and flexible tool that performs well in the majority of situations. It shows that the UKF will provide increased performance over the EKF given large initial condition errors or highly non-linear systems. The PF is outlined as an effective tool when the noise of the system is non-Gaussian, but has been relegated to these niche scenarios due to its computational complexity. Another survey provides an experimental comparison of an EKF to complementary filter and another non-linear observer [28]. It shows that the EKF outperforms the complementary filter in most cases but that it comes with a significant increase in computational complexity.

1.2 Objectives

This thesis aims to find the attitude estimation technique which provides the most accurate state estimate during a quadcopter collision. A method to model the effects of a collision on IMU sensor data is required for simulation and a model approximating these effects is developed. Novel adaptive attitude estimation algorithms as well as robust estimation algorithms are evaluated to see if they can provide improved performance when the sensors experience anomalies due to a collision. A comparison of existing attitude estimation algorithms is done along side these novel algorithms in order to evaluate their performance as well. In particular a comparison is made between an MEKF,

a UKF, two novel varieties of adaptive UKFs, a Complementary filter, an H_∞ filter and a novel adaptive H_∞ filter. While particle filters could be effective at dealing with anomalies in sensor noise due to a collision, they are not investigated here as it is likely that they are too computationally heavy to implement onboard a quadcopter. A norm-constrained version of the EKF and H_∞ filter, as well as a SRUKF are also compared to the conventional versions of each filter to see if they can provide improved performance for attitude estimation using the given hardware.

1.3 Estimator Evaluation

The estimation algorithms' performance is evaluated by comparing their performance first in simulation, and then using experimental data. In simulation, the estimators are compared in a Monte Carlo framework, during normal (i.e. collision-free) flight as well as during a collision. The evaluation of the estimators performance using experimental collision data is done using two methods. In the first method, the quadcopter's state is tracked during a collision using a Vicon motion capture system. The Vicon state estimate is then compared to attitude estimates computed by post processing the sensor data using the specified attitude estimation algorithms. The Vicon system provides a very accurate attitude estimate and is taken as the true state, allowing a computation and comparison of the error in each algorithms' attitude estimate. In the second method, a Monte Carlo framework is used to compare the best performing estimators by implementing them onboard a quadcopter with the collision recovery controller. The estimators' performance is evaluated by comparing the time it takes to recover from a collision, as a better attitude estimate should lead to a quicker and more consistent recovery. The platform used for the experiments is Navi, a custom quadcopter which uses a Pleiades Spiri frame. The quadcopter has been retrofitted with a Pixhawk flight controller, an ODROID single board computer and custom designed bumpers in order to withstand collisions.

1.4 Thesis Organization

The thesis has the following structure: Chapter 1 introduces the thesis, its motivation and relevant background information. Chapter 2 details the dynamics, sensor, and kinematic models used in simulation and by the estimation algorithms. Chapter 3 explains the basic estimation

FIGURE 1.1: Spiri Quadcopter UAV

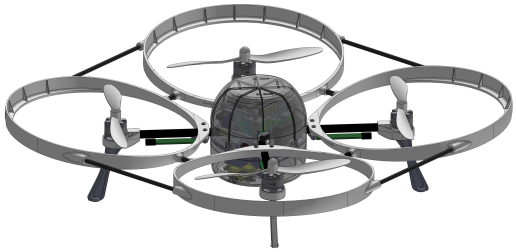

FIGURE 1.2: Custom Navi Quadcopter UAV

algorithms as well as their adaptive counterparts. Chapter 4 contains the simulation parameters and discusses the simulation results. Chapter 5 details changes in the implementation of the algorithms when using real sensor data then presents and discusses the results of the two methods used to compare experimental data. Chapter 6 concludes the thesis and discusses possible future work.

Chapter 2

System Modelling

In this chapter a dynamics model of a quadcopter is presented and the quaternion kinematics and conventions are detailed. The sensor models are stated along with an evaluation of the effect of a collision on sensor data. Modifications made to the sensor models in order to better simulate the effect of a collision are derived. While all the models given in this chapter are used for simulation purposes, the quaternion kinematics and basic sensor models are also used in formulating the attitude estimation algorithms in Chapter 3.

2.1 Quadcopter Dynamics and Kinematics

The modelling of quadcopter rigid-body dynamics is well documented [5], [29]. The MATLAB simulation used in this thesis builds upon the simulation environment developed by Chui [30] and Dicker [31] which was used in their research to develop a quadcopter collision dynamics model and a collision recovery controller. The quadcoptor dynamics are modelled and simulated using an inertial frame $\boldsymbol{\mathcal{F}}_i$ and a body frame $\boldsymbol{\mathcal{F}}_b$. The body frame is located at the center of mass (CM) of the quadcopter with the inertial frame following a North-East-Down convention. The dynamics of the quadcopter are modelled using the Newton-Euler formulation for a rigid body, and incorporate the forces arising due to impact or contact with the environment. The translational dynamics are modelled as:

$$m\dot{\mathbf{v}} + m\boldsymbol{\omega}^{\times}\mathbf{v} = \mathbf{F}_G + \mathbf{F}_T + \mathbf{F}_C \tag{2.1}$$

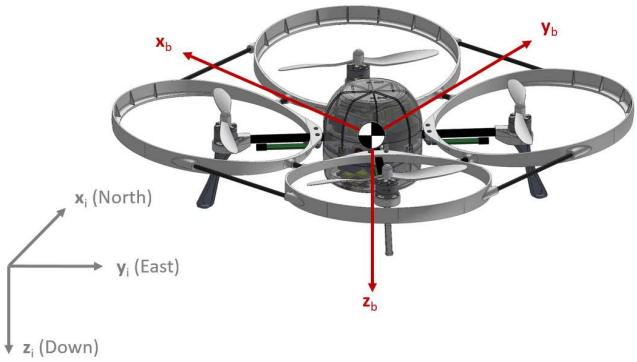

FIGURE 2.1: Coordinate frames used in dynamics and kinematics models

and the rotational dynamics are modelled with:

$$\mathbf{I}\dot{\boldsymbol{\omega}} = \sum_{j=1}^{4} \mathbf{r}_{Tj}{}^{\times}\mathbf{F}_{Tj} + \mathbf{M}_T + \mathbf{M}_\Omega + \mathbf{M}_C - \boldsymbol{\omega}^{\times}\mathbf{I}\boldsymbol{\omega} \tag{2.2}$$

where $\mathbf{F}$ and $\mathbf{M}$ are the applied forces and moments and $\mathbf{v} = [u \quad v \quad w]^\mathsf{T}$ and $\boldsymbol{\omega} = [p \quad q \quad r]^\mathsf{T}$ are the linear velocity of the CM and angular velocity of the quadcopter, all expressed in $\boldsymbol{\mathcal{F}}_b$. For the forces and moments, the subscripts G, T, Ω and C denote gravitational, thruster, gyroscopic and contact respectively. The exact definitions of these can be found in [5], and it should be noted that the moments due to aerodynamic drag and propeller flapping are neglected here. The platform specific parameters m, $\mathbf{I}$, and $\mathbf{r}_{Tj}$ are the mass, moment of inertia matrix and position of the thruster locations relative to the center of mass, respectively. The operator $^{\times}$ denotes the skew symmetric matrix and is defined as

$$\mathbf{a}^{\times} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{2.3}$$

The position of the CM of the vehicle is propagated using the velocity kinematics as:

$$\dot{\mathbf{p}} = \mathbf{C}\mathbf{v} \tag{2.4}$$

where the position, $\mathbf{p}$, of the quadcopter's CM is expressed in $\boldsymbol{\mathcal{F}}_i$ and $\mathbf{C}$ is the rotation matrix, which is defined shortly.

As already noted in Chapter 1, a quaternion is used to parametrize the attitude of the vehicle as it provides a singularity free attitude parametrization. It is singularity free partly because it is subject to the norm constraint $\mathbf{q}^\mathsf{T}\mathbf{q} = 1$. The Hamilton convention [32] is used for the quaternion which is related to the axis of rotation, $\mathbf{a}$, and angle of rotation, ϕ, by

$$\mathbf{q} = \begin{bmatrix} \cos\frac{\phi}{2} \\ \mathbf{a}\sin\frac{\phi}{2} \end{bmatrix} = \begin{bmatrix} q_0 \\ \boldsymbol{\varrho} \end{bmatrix} \tag{2.5}$$

The quaternion is propagated by the angular velocity of the vehicle as:

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{2.6}$$

where the $\otimes$ operator represents quaternion multiplication and is defined by:

$$\mathbf{q} \otimes \mathbf{p} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \mathbf{p} \tag{2.7}$$

with q_{1-3} corresponding to the components of $\boldsymbol{\varrho}$ in Eq. (2.5). The rotation matrix is derived from the quaternion using:

$$\mathbf{C}(\mathbf{q}) = (q_0^2 - \|\boldsymbol{\varrho}\|^2)\mathbf{1}_3 + 2\boldsymbol{\varrho}\boldsymbol{\varrho}^\mathsf{T} + 2q_0\boldsymbol{\varrho}^\times$$

where $\mathbf{1}_3$ denotes the 3×3 identity matrix and the above rotation matrix rotates components of a vector in $\boldsymbol{\mathcal{F}}_b$ to those in $\boldsymbol{\mathcal{F}}_i$. The quaternion which corresponds to zero rotation is $\mathbf{q} = [1, \quad 0, \quad 0, \quad 0]^\mathsf{T}$, making the corresponding rotation matrix the identity matrix. A discretized version of (2.6) based on a power series expansion is used to propagate the quaternion between time step $k-1$ and time step k in the estimation algorithms derived in Chapter 3. The discretized quaternion kinematics are:

$$\mathbf{q}_k = \boldsymbol{\Omega}(\boldsymbol{\omega}_{k-1})\mathbf{q}_{k-1} \tag{2.8}$$

where

$$\boldsymbol{\Omega}(\boldsymbol{\omega}_{k-1}) = \begin{bmatrix} \cos(0.5\psi_{k-1}\Delta t) & -\boldsymbol{\psi}_{k-1}^{\mathsf{T}} \\ \boldsymbol{\psi}_{k-1} & \cos(0.5\psi_{k-1}\Delta t)\mathbf{1}_3 - \boldsymbol{\psi}_{k-1}^{\times} \end{bmatrix} \tag{2.9}$$

and the scalar ψ, and vector $\boldsymbol{\psi}$, are computed with:

$$\psi_{k-1} = \|\boldsymbol{\omega}_{k-1}\| \tag{2.10}$$

$$\boldsymbol{\psi}_{k-1} = \sin(0.5\psi_{k-1}\Delta t)\boldsymbol{\omega}_{k-1}/\psi_{k-1} \tag{2.11}$$

Here Δt is the time between steps $k-1$ and k [10].

As the quaternion can be difficult to visualize and interpret, it is useful to convert it to a set of Euler angles. Euler angles are a 3-parameter parametrization of rotations and they correspond to a sequence of three rotations about a particular axis of the frame being rotated. The rotations can be applied in many different combinations, but here we use the convention of first applying a yaw rotation (rotation about the z-axis), a pitch rotation (rotation about the y-axis), and then a roll rotation (rotation about the x-axis). The conversion from quaternion to Euler angles is therefore [33]

$$\begin{bmatrix} \phi_E \\ \theta_E \\ \psi_E \end{bmatrix} = \begin{bmatrix} \text{atan2}\big(2(q_0q_1 + q_2q_3), \quad (q_0^2 - q_1^2 - q_2^2 + q_4^2)\big) \\ \sin^{-1}\big(-2(q_1q_3 - q_0q_2)\big) \\ \text{atan2}\big(2(q_1q_2 + q_0q_3), \quad (q_0^2 + q_1^2 - q_2^2 - q_3^2)\big) \end{bmatrix} \tag{2.12}$$

where atan2(x, y) is the inverse tangent function which appropriately computes $\tan^{-1}$(x/y) if the angle (x/y) is outside the range $-\pi/2$ and $\pi/2$.

2.2 Sensor Models

2.2.1 General Sensor Models

In order to properly validate state estimation algorithms in simulation, the data from the sensors which are used must be simulated. Sensor data is simulated by corrupting the true values which the sensors are measuring with noise. A simple and common method used to model Micro Electro Mechanical (MEM) IMUs is to corrupt the true value of the sensor with a random walk bias term and a Gaussian noise term [19], [34], [35]. More complicated error terms can be introduced, such as

misalignment, non-orthogonality, and scale factor errors, but these tend to be accounted for during sensor calibration as they are generally constant after a sensor suite is assembled [34]. Sensor noise comes from vibrations in the system, and electrical noise caused by circuitry and motors. Bias in IMU sensors can be due to many causes, such as, changes in temperature, mechanical stress, or changes in start-up conditions. Changes in sensor bias during a flight, or while a vehicle is powered on, tends to occur slowly. The largest error in a sensor bias estimate will occur when a vehicle is powered on, as a large bias could have developed while the the vehicle has been turned off and there is not an easy way to estimate initial sensor bias. Here, since the experimental flight time is short, the change in sensor bias during a flight is likely to be very small. Thus, the sensor bias during a flight can simply be modelled as a constant value and an algorithm's efficacy at estimating sensor bias can be garnered from the algorithms ability to converge toward the initial bias values.

The basic sensor models assumed in this research are:

$$\mathbf{u}_a = \dot{\mathbf{v}} + \mathbf{C}^\mathsf{T}\mathbf{g} + \mathbf{b}_a + \boldsymbol{\eta}_a \tag{2.13a}$$

$$\mathbf{u}_g = \boldsymbol{\omega} + \mathbf{b}_g + \boldsymbol{\eta}_g \tag{2.13b}$$

$$\mathbf{u}_m = \mathbf{C}^\mathsf{T}\boldsymbol{\mu} + \mathbf{b}_m + \boldsymbol{\eta}_m \tag{2.13c}$$

where $\mathbf{u}_a$, $\mathbf{u}_g$ and $\mathbf{u}_m$ are the accelerometer, gyroscope and magnetometer measurements, $\mathbf{b}_a$, $\mathbf{b}_g$ and $\mathbf{b}_m$ are the corresponding sensor biases, and $\boldsymbol{\eta}_a$, $\boldsymbol{\eta}_g$ and $\boldsymbol{\eta}_m$ are zero mean Gaussian noise variables with covariances σ_a, σ_g, and σ_m, all defined in $\boldsymbol{\mathcal{F}}_b$. In simulation, the earth's magnetic field,

$$\boldsymbol{\mu} = [.302 \quad 0 \quad .950]^\mathsf{T} \tag{2.14}$$

is defined in $\boldsymbol{\mathcal{F}}_i$ and corresponds to the magnetic field vector in the area of Montreal, Canada, computed using the International Geomagnetic Reference Field model [36]. It should be noted that in experiments, the local inertial magnetic field value is estimated using magnetometer measurements with the method discussed in Chapter 5. The gravitational vector is

$$\mathbf{g} = [0 \quad 0 \quad -g]^\mathsf{T} \tag{2.15}$$

where g is earth's gravitational acceleration constant.

2.2.2 Effects of a Collision on Sensor Data

The effects of a collision on sensor data were investigated in order to determine if the sensors registered anything beyond the conventional sensor noise, sensor bias or motion of the vehicle during a collision. The investigation showed that the inertial sensors (accelerometer and gyroscope) were significantly affected by a collision whereas the effect on the magnetometer was minimal. The gyroscope and accelerometer appeared to measure vibrations in the system caused by the collision. These sensors are more susceptible to vibrations as they are inertial sensors, as opposed to the magnetometer which is an electromagnetic sensor. In order to investigate anomalies in the sensor data caused by the collision, the components of the sensor data which correspond to the sensor bias and motion of the vehicle were removed from the sensor data. To remove these components, the sensor data was passed through a highpass FIR filter, removing the low frequency data associated with sensor bias and vehicle motion. While it is possible that passing the sensor data through a highpass filter may remove low frequency anomalies caused by the collision itself, this is thought to be unlikely. It is more likely that any anomalies in the sensor data caused by the collision occur in the high frequency range due to vibrations in the vehicle after the collision. The highpass filter was designed using MATLAB's filter design tools such that the filter transition band was between 5 Hz and 10 Hz, with the passband frequency being 10 Hz. The conventional high frequency noise which occurs in the sensors was not removed in this investigation as removing it may have resulted in the removal of anomalies caused by the collision. As the conventional sensor noise tends to have a constant mean, it is easy to tell which parts of the filtered sensor data can be attributed to conventional noise, and which are caused by the collision, without separating the data.

The sensor data used for this investigation was from a set of 72 collisions, with a wide range of impact velocities and impact angles. The sensor data for the collisions had some similar trends across many different data sets but there were also some unique anomalies, likely due to the variety of impact angles and velocities. An examination of the most frequently occurring sensor collision response was used to develop a method to corrupt the true sensor data in imitation of a collision. The effect which occurred most frequently and consistently in the inertial sensor data during and after a collision was a decaying 'vibration'. As illustrated in Figure 2.2 with the highpass filtered x-axis gyroscope data, the sensor appears to experience a decaying vibration which is accompanied by an increase in noise. In Figure 2.2, the collision occurs in the filtered data at the peak, around

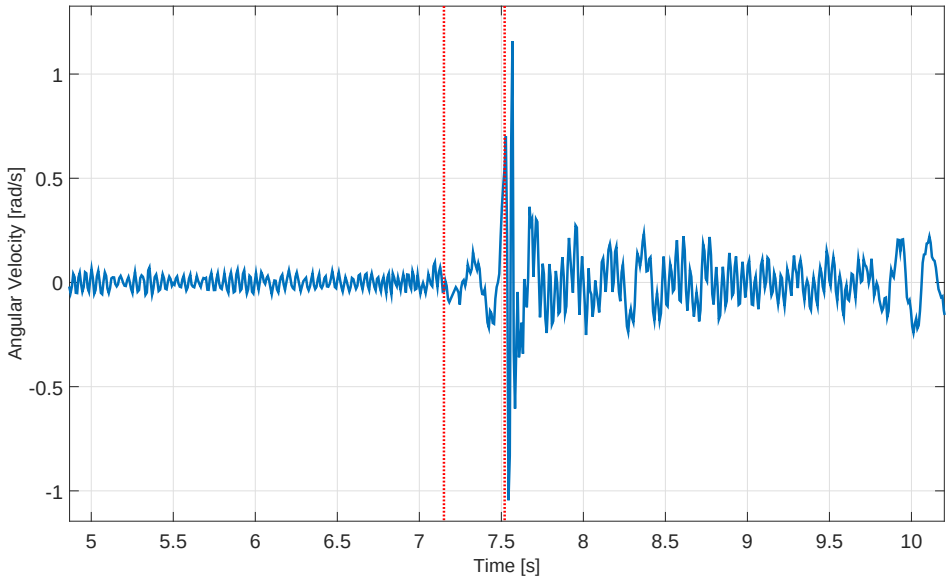

FIGURE 2.2: High pass filtered x-axis gyroscope data during collision. Dotted red lines mark the times of collision in the unfiltered data (7.2 seconds) and the phase delayed time of collision in the filtered data (7.5 seconds).

7.5 seconds (marked by a red dashed line). In the pre-filtered data, the actual collision occurs at around the 7.2 second mark (also marked by a red dashed line), however the FIR filter causes a phase shift in the data of about 0.3 seconds, causing the collision data to appear around the 7.5 second mark in Figure 2.2. In between the actual time of impact at 7.2 seconds, and the phase delayed filtered time of impact, at 7.5 seconds, an increasing oscillation of the measurement can be seen. This oscillation is likely caused by the FIR filter, as it appears in the majority of the filtered sensor data, but not in unfiltered data. After the collision the measurement oscillates and is seen to have significantly more noise than before the collision. While this type of sensor response is only shown for the x-axis of the gyroscope, it also occurs along the other axes as well as in the accelerometer data, although the noise and oscillations have varied magnitudes. In the data set illustrated in Figure 2.2 it can be seen that after the collision the noise in the measurement appears to increase, and it looks as if there are oscillations in the data. In some collisions, the oscillations are much more distinct, and less noisy, as in Figure 2.3 which shows the filtered z-axis accelerometer data from another collision.

In Figure 2.3 it can be seen that the sensor measurement experiences a distinct peak during the collision which oscillates and decays. Again you can see a phase shift of about 0.3 seconds

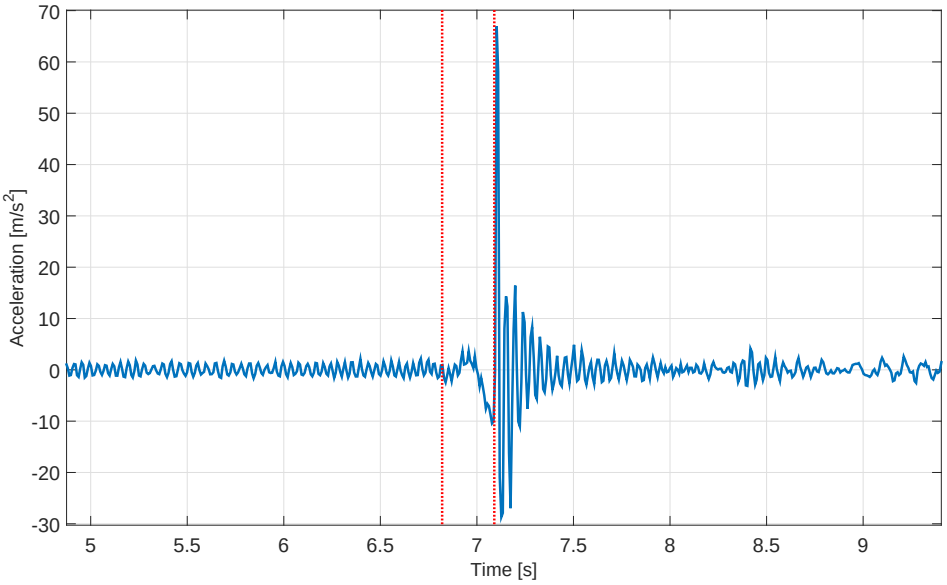

FIGURE 2.3: High pass filtered z-axis accelerometer data during collision. Dotted red lines mark the times of collision in the unfiltered data (6.85 seconds) and the phase delayed time of collision in the filtered data (7.15 seconds).

caused by the filter, with the collision occurring around 7.15 seconds in the filtered data (marked by a red dashed line), while the collision actually occurred around 6.85 seconds in the pre-filtered data (also marked by a red dashed line). The same sort of oscillations occur between the actual and phase delayed time of collision as is seen in the previous filtered data of Figure 2.2. The difference between the two data sets in Figures 2.2 and 2.3 is that the sensor data in Figure 2.3 shows a distinct oscillation which decays, whereas the measurement in Figure 2.2 is a lot noisier and the oscillations that occur are at a lower frequency. It can also be seen that the measurement in Figure 2.3 is also oscillating before and significantly after the collision (for example between 5.5-6.5 seconds). This oscillation during normal flight is thought to be caused by the bumpers and frame of the quadcopter vibrating due to the rotation of the quadcopter propellers. The kinds of anomalies described in Figure 2.3 also occur along other sensor axes, in both gyroscope and accelerometer data for some collisions.

It is thought that the anomalies observed are due to vibrations caused by the collision. A data set which shows a noisier response, similar to that of Figure 2.2, is thought to have many different modes of vibration occurring, at a variety of frequencies, resulting in the data appearing to be noisier. A data set which shows a distinct decaying oscillation, similar to Figure 2.3, is thought to

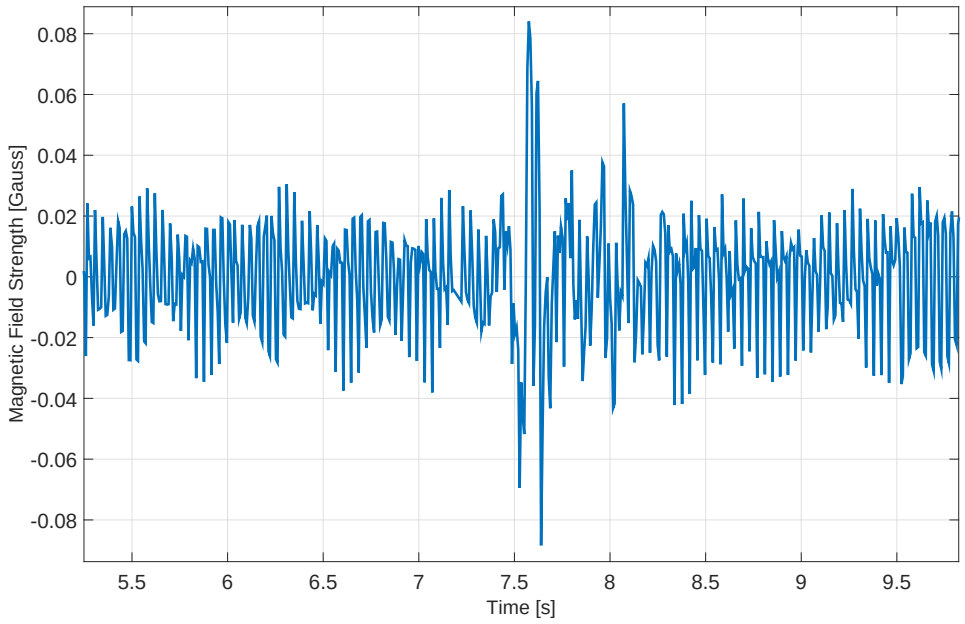

FIGURE 2.4: High pass filtered x-axis magnetometer data during collision

have one dominant vibration occurring. Since each collision occurs at different angles, velocities and at different points of impact, various modes of vibration may be excited in the vehicle, resulting in the varying sensor responses that were observed. There were also other anomalies that would occur in the sensor data, but they were much less frequent or consistent in nature.

As for the magnetometer, its data would sometimes show an abrupt increase in noise during or after a collision, as seen between 7-8.5 seconds in Figure 2.4, which shows the filtered x-axis magnetometer data, but this occurred inconsistently and no clear pattern to the anomalies in the magnetometer measurements were apparent. Any instantaneous anomalies in the magnetometer data at the time of impact, similar to those shown in Figure 2.4, were orders of magnitude smaller compared to the anomalies in the accelerometer and gyroscope.

2.2.3 Modelling Sensor Collision Anomalies for Simulation

From the sensor data it can be seen that the quadcopter appears to experience a series of decaying vibrations. A collision is likely similar to applying an impulse to the system, or a series of impulses. It is thought that the reason why the vibrations would sometimes appear clearly in inertial sensor data and other times very noisily is likely due to the natural frequency of the quadcopter,

the angular velocity of the propellers, the characteristics of the impact, and the sampling time of the sensors. Depending on the characteristics of the impact (velocity, angle, duration and point of impact) different modes of vibration would occur in the quadcopter, resulting in different effects on the sensor data.

Two methods were considered to simulate the effect of collisions on sensor data. One was to corrupt the sensor data with a set of decaying sinusoidal vibrations; the other was to increase the covariance of the Gaussian noise used to corrupt the measurements by a decaying factor. It was thought that the first method, corrupting the data with a set of sinusoids, was too complex, as the frequency, magnitude and number of sinusoids would heavily depend on the characteristics of the collision and the effect of these variables would significantly impact the results. In order to accurately model the collision in this way an extensive vibrational analysis of the quadcopter during a collision would need to be performed. The second method was chosen as it was simple to implement and adequately corrupted the sensor data during a collision. As such, the covariances for the accelerometer and gyroscope noise are modelled as

$$\sigma_a = \tilde{\sigma}_{a0} + \tilde{\sigma}_{ac} e^{-t_c/\tau} \tag{2.16}$$

$$\sigma_g = \tilde{\sigma}_{g0} + \tilde{\sigma}_{gc} e^{-t_c/\tau} \tag{2.17}$$

where t_c is the time elapsed since the crash and $t_c = \infty$ before the crash occurs, τ is a time constant, $\tilde{\sigma}_{ac}$ and $\tilde{\sigma}_{gc}$ are the maximum crash covariances, and $\tilde{\sigma}_{a0}$ and $\tilde{\sigma}_{g0}$ are the normal operational covariances for the accelerometer and gyroscope, respectively. Using this model, the sensor noise covariances are set to a standard value during normal operation. When a collision occurs, the noise covariance is increased by a value which decays to near zero over several time constants, leaving the noise covariance back at its original value. The value of the crash covariance and the rate at which it decays are kept constant between simulation runs and are chosen based on the peak values which occurred during more aggressive collisions. The magnetometer covariance is kept as a constant (which is the standard approach to modelling noise), as the effect of a collision on magnetometer data was significantly less than it was for the two inertial sensors.

The modifications to the sensor models made to more realistically simulate a collision are used

to generate sensor measurements for the simulations in Chapter 4. The simulated sensor measurements are used within the estimation algorithms derived in Chapter 3, in the place of real sensor measurements. While only the basic sensor models given in Section 2.2.1 of this chapter are used directly in the derivations of the attitude estimation algorithms presented in Chapter 3, the adaptive algorithms are specifically designed to better handle the modifications made to the sensor models due to collisions.

Chapter 3

Attitude Estimation Algorithms

In this chapter, the attitude estimation algorithms under examination are detailed. First, general information which applies to all estimators is given. Then the equations for the attitude estimation algorithms are presented in order of increasing conceptual and computational complexity. First, the complementary filter and MEKF are derived, followed by the H_∞ filter and its adaptive version. Then the equations for the UKF are given and equations for two adaptive versions, a high gain Adaptive UKF (AUKF) and a covariance matching AUKF, are derived. It should be noted that the adaptive H_∞ and UKF algorithms are specifically designed in an attempt to improve performance during a collision. In the final section of the chapter, the Pixhawk attitude estimation algorithm is presented. An important part of the evaluation of the estimation algorithms is a comparison of these algorithms to the attitude estimation algorithm that is stock in the Pixhawk firmware.

The firmware used onboard the Pixhawk is the PX4 flight stack, an open source codebase available on GitHub [37]. Another codebase is also available for use with the Pixhawk, the ArduPilot flight stack, but the PX4 flight stack is used here as it is more modular and easy to incorporate custom code into. The PX4 flight stack is also useful as it offers an estimation scheme which uses separate estimation algorithms to estimate attitude and position, making it easy to replace the PX4 attitude estimator with an alternative algorithm. As it is important to evaluate the efficacy of the PX4 attitude estimator during collisions, the PX4 attitude estimation algorithm has been extracted from the code and presented in this chapter for comparison.

An estimation algorithm which is portable between vehicles is desired; thus, the implementation of the algorithms in this thesis avoids the use of a dynamics model in order to provide portability. All state estimators considered, including the Pixhawk algorithm, use the sensor data similarly: the bias corrected gyroscope values are used in the discrete time quaternion kinematics, Eq. (2.8),

to propagate the attitude while the accelerometer and magnetometer measurements are used as attitude vector measurements. This means that a prediction of the accelerometer and magnetometer measurements are made, and a correction factor is generated based on the difference between the predicted and measured vectors. The states estimated by the algorithms are the quaternion and gyroscope bias:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{q}} \\ \hat{\mathbf{b}}_g \end{bmatrix}$$

while the angular velocity estimate is found by taking:

$$\hat{\boldsymbol{\omega}} = \mathbf{u}_g - \hat{\mathbf{b}}_g$$

Here, an overhat signifies that a variable is an estimate of the true value, and variables without overhats correspond to the true value. Of the sensor biases, only the gyroscope bias, $\mathbf{b}_g$, is estimated since the gyroscope measurement is propagated through the quaternion kinematics and bias in the sensor will accumulate over every time step, resulting in significant drift and error in the estimate. While the accelerometer and magnetometer biases have an effect on the estimate, the errors caused by them do not accumulate and only result in a slight offset. Estimating the accelerometer and magnetometer biases could also result in observability issues as the state vector would be significantly larger than the number of measurements available; therefore, these biases are not estimated.

Using the accelerometer measurement as a vector measurement can also present a problem when the vehicle is undergoing significant linear acceleration. If the vehicle is accelerating in any direction, the accelerometer measurement will be dependent on the linear acceleration as well as the gravitational vector and this could result in significant error as the current sensors do not permit a method to accurately estimate the linear acceleration of the vehicle. Thus, the accelerometer measurement is only used when the following condition is satisfied:

$$g - a_{bnd} \leq \|\mathbf{u}_a\| \leq g + a_{bnd} \quad [m/s^2]$$

where a_{bnd} is a value selected based on tuning. The bound on the accelerometer measurement is applied to all of the estimation algorithms, except for the PX4 estimator. All values used for initial conditions, tuning parameters and noise parameters in the estimation algorithms are specified in

Chapter 4 for use in simulation, and in Chapter 5 for use in experiments.

3.1 Complementary Filter

The complementary filter (named due to its similarities to a conventional complementary filter) is a non-linear observer designed by Mahony et al. [19]. The original derivation of the observer uses the DCM as its attitude parametrization but an equivalent method using quaternions is also presented, along with a comparison showing the similarity between the complementary filter and a previous quaternion observer. They present three versions of the complementary filter, two which require a direct measurement of the attitude (or a prediction of it computed based on vector measurements), and one version that directly uses vector measurements. The version used here is the third method which is termed the *explicit* complementary filter. Mahony et al. prove the local exponential stability of the error through the use of Lyapunov functions, showing that the filters are only unstable for three sets of extreme conditions: a rotation of 180 degrees about any of the three main axes, corresponding to when the error in the filter is at a maximum. The complementary filter uses a correction term based on the accelerometer and magnetometer measurements, along with bias corrected gyroscope measurement, directly in the quaternion kinematic equation. The accelerometer and magnetometer correction term is a function of the normalized measurements and a prediction of the normalized measurements. The complementary filter replaces $\boldsymbol{\omega}_{k-1}$ in (2.8) as

$$\hat{\mathbf{q}}_k = \boldsymbol{\Omega}(\mathbf{u}_g - \hat{\mathbf{b}}_g + k_p \boldsymbol{\omega}_{mes})\hat{\mathbf{q}}_{k-1} \tag{3.1}$$

where

$$\boldsymbol{\omega}_{mes} = -\text{vex}\left[\frac{k_a}{2}(\bar{\mathbf{u}}_a \hat{\mathbf{u}}_a^\mathsf{T} - \hat{\mathbf{u}}_a \bar{\mathbf{u}}_a^\mathsf{T}) + \frac{k_m}{2}(\bar{\mathbf{u}}_m \hat{\mathbf{u}}_m^\mathsf{T} - \hat{\mathbf{u}}_m \bar{\mathbf{u}}_m^\mathsf{T})\right] \tag{3.2}$$

The values k_p, k_a, and k_m are estimator gains, chosen by tuning. The vex[ ] operator is the uncross operator, which converts a skew symmetric matrix back into a vector, the inverse of $^\times$ given in equation (2.3). The estimated measurements, $\hat{\mathbf{u}}_a$ and $\hat{\mathbf{u}}_m$, are computed as

$$\hat{\mathbf{u}}_a = \mathbf{C}(\hat{\mathbf{q}}_{k-1})^\mathsf{T}\mathbf{g}/\|\mathbf{g}\| \tag{3.3a}$$

$$\hat{\mathbf{u}}_m = \mathbf{C}(\hat{\mathbf{q}}_{k-1})^\mathsf{T}\boldsymbol{\mu}/\|\boldsymbol{\mu}\| \tag{3.3b}$$

and the values $\bar{\mathbf{u}}_a$ and $\bar{\mathbf{u}}_m$ are the normalized measurements. The correction factor, $\boldsymbol{\omega}_{mes}$, is a term which is perpendicular to the error between the predicted measurements and the actual measurements, and it rotates the attitude estimate to help align the vectors. The gyroscope bias is estimated using

$$\dot{\hat{\mathbf{b}}}_g = -k_b \boldsymbol{\omega}_{mes} \tag{3.4}$$

where k_b is an estimator gain, also chosen by tuning. The bias at timestep k can be found using an Euler integration scheme:

$$\hat{\mathbf{b}}_{g|k} = \hat{\mathbf{b}}_{g|k-1} - k_b \boldsymbol{\omega}_{mes} \Delta t \tag{3.5}$$

The complementary filter requires a user selected, initial prediction of the attitude, $\hat{\mathbf{q}}_0$, and the gyroscope bias, $\hat{\mathbf{b}}_{g|0}$.

3.2 Multiplicative Extended Kalman Filter

The EKF is one of the most used state estimation algorithms due to its proven performance [8]. There are numerous variations of the EKF, none of which can be claimed as the definitive "best", although some are known to perform better in specific applications. The linear KF is the best linear unbiased estimator for systems with Gaussian noise, but in order to apply it to non-linear systems, the EKF linearizes the state transition and measurement equations about an operating point. This approximation makes the few proofs of its performance limited to local stability proofs, and proofs which require strict, difficult-to-satisfy conditions [38]. The issue with using a conventional EKF to estimate an attitude quaternion is that the additive nature of the KF measurement correction violates the quaternion norm constraint. The MEKF improves upon a conventional EKF in estimating quaternions by estimating the error quaternion instead of the full quaternion. This reduces the error due to the quaternion norm constraint as the measurement update can be applied using quaternion multiplication, as opposed to the conventional additive method. Another reason why the MEKF may perform better than a conventional EKF is that the error quaternion will usually correspond to a small rotation; thus, the magnitude of the violation of the norm constraint will be smaller than if the full quaternion is used [10]. This can also allow for a reduction in state size as the scalar part of the quaternion error derivative linearizes to zero.

One of the benefits of Kalman based filters, and stochastic estimation algorithms in general, over non-linear observers is the use of a state covariance matrix to correct the state, which can provide a direct measure of the accuracy of the current state estimate. As the MEKF estimates the error state, the state covariance matrix corresponds to the likelihood of the error state, not the likelihood of the actual state, thus its meaning is slightly more difficult to interpret.

The state vector of the MEKF is $\delta\mathbf{x} = [\delta\mathbf{q}^\mathsf{T} \quad \delta\mathbf{b}_g^\mathsf{T}]^\mathsf{T}$ where δ signifies the error between the true and the estimated state. The error quaternion is defined as:

$$\delta\mathbf{q} = \hat{\mathbf{q}}^{-1} \otimes \mathbf{q} \tag{3.6}$$

and the derivative of the error quaternion can be shown to be [10]:

$$\delta\dot{\mathbf{q}} = -\begin{bmatrix} 0 \\ \hat{\boldsymbol{\omega}}^\times \delta\boldsymbol{\varrho} \end{bmatrix} + \frac{1}{2}\delta\mathbf{q} \otimes \begin{bmatrix} 0 \\ \delta\boldsymbol{\omega} \end{bmatrix} \tag{3.7}$$

where $\delta\boldsymbol{\omega}$ is the angular velocity error and is defined as $\delta\boldsymbol{\omega} = -(\delta\mathbf{b}_g + \boldsymbol{\eta}_g)$. The MEKF predicts the current state based on the previous state and the gyroscope measurement as

$$\hat{\boldsymbol{\omega}}_k^- = \mathbf{u}_g - \hat{\mathbf{b}}_{g|k-1}^+ \tag{3.8}$$

$$\hat{\mathbf{q}}_k^- = \boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}_k^-)\hat{\mathbf{q}}_{k-1}^+ \tag{3.9}$$

$$\hat{\mathbf{b}}_{g|k}^- = \hat{\mathbf{b}}_{g|k-1}^+ \tag{3.10}$$

where the superscripts $^+$ and $^-$ denote the post-measurement correction values and pre-measurement correction values, respectively. The state covariance matrix is predicted with:

$$\mathbf{P}_k^- = \mathbf{F}_{k-1}\mathbf{P}_{k-1}^+\mathbf{F}_{k-1}^\mathsf{T} + \mathbf{Q}_{k-1} \tag{3.11}$$

where $\mathbf{P}$ is the state covariance matrix, $\mathbf{F}$ is the linearized state transition matrix and $\mathbf{Q}$ is the process noise covariance matrix. The linearized state matrix,

$$\mathbf{F}_{k-1} = \begin{bmatrix} 1 & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{3\times 1} & \mathbf{1}_3 - \Delta t \hat{\boldsymbol{\omega}}_{k-1}^{\times} & -\frac{\Delta t}{2}\mathbf{1}_3 \\ \mathbf{0}_{3\times 1} & \mathbf{0}_3 & \mathbf{1}_3 \end{bmatrix} \tag{3.12}$$

is found by discretizing and taking the first order approximation of (3.7) and (3.10), and is only used to propagate the state covariance matrix, not the state itself. The filter requires initial guesses for the quaternion, $\hat{\mathbf{q}}_0$, gyroscope bias, $\hat{\mathbf{b}}_{g|0}$, and state covariance matrix $\mathbf{P}_0$. The measurements, $\hat{\mathbf{y}}_k$, are estimated based on the current state estimate as:

$$\hat{\mathbf{y}}_k = \begin{bmatrix} \hat{\mathbf{u}}_a \\ \hat{\mathbf{u}}_m \end{bmatrix} \tag{3.13}$$

$$\hat{\mathbf{u}}_a = \mathbf{C}(\delta\mathbf{q})^{\mathsf{T}}\mathbf{C}(\hat{\mathbf{q}}_k^-)^{\mathsf{T}}\mathbf{g} \tag{3.14a}$$

$$\hat{\mathbf{u}}_m = \mathbf{C}(\delta\mathbf{q})^{\mathsf{T}}\mathbf{C}(\hat{\mathbf{q}}_k^-)^{\mathsf{T}}\boldsymbol{\mu} \tag{3.14b}$$

The term $\mathbf{C}(\delta\mathbf{q})$ is important to include in the above equations as it is used to develop the linearized measurement matrix; however in estimating the measurements, it is taken as the identity matrix. This is because the estimated error quaternion is based on the error between the estimated and actual measurements, and thus before an estimation of the measurements is made, the estimated error quaternion must correspond to zero rotation. The difference between the actual and the estimated measurements, $\boldsymbol{\epsilon}_k = \mathbf{y}_k - \hat{\mathbf{y}}_k$, is called the innovation. The innovation can be a useful measure of accuracy of the estimation algorithm as a large innovation signifies large discrepancies between the gyroscope and the other IMU sensors. The Kalman gain uses the innovation to compute the error quaternion and gyroscope bias error. The Kalman gain is computed as:

$$\mathbf{K}_k = \begin{bmatrix} \mathbf{K}_{q|k} \\ \mathbf{K}_{b|k} \end{bmatrix} = \mathbf{P}_k^-\mathbf{H}_k^{\mathsf{T}}\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathsf{T}} + \mathbf{R}_k\right)^{-1} \tag{3.15}$$

where $\mathbf{R}$ is the measurement noise covariance matrix, and $\mathbf{H}$ is developed using the linearized measurement equations. The linearized measurement matrix is shown to be [10]:

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{0}_{3\times 1} & (\mathbf{C}(\hat{\mathbf{q}}_k^-)^\mathsf{T}\mathbf{g})^\times & \mathbf{0}_3 \\ \mathbf{0}_{3\times 1} & (\mathbf{C}(\hat{\mathbf{q}}_k^-)^\mathsf{T}\boldsymbol{\mu})^\times & \mathbf{0}_3 \end{bmatrix} \tag{3.16}$$

It is important to note, as is the case in the other Kalman based filters discussed in this thesis, that $\mathbf{y}_k$, $\hat{\mathbf{y}}_k$, $\mathbf{H}$ and $\mathbf{R}$ will change sizes depending on whether the accelerometer measurement is used or not. The quaternion error and gyroscope bias error are then computed as

$$\delta\mathbf{q}_{upd}^+ = \mathbf{K}_{q|k}\boldsymbol{\epsilon}_k \tag{3.17}$$

$$\delta\mathbf{b}_{g|k}^+ = \mathbf{K}_{b|k}\boldsymbol{\epsilon}_k \tag{3.18}$$

Since the quaternion error term is assumed to be small, it can be approximated as $\delta\mathbf{q}_k^+ = [1 \quad \frac{1}{2}\delta\boldsymbol{\varrho}_{upd}^+]^\mathsf{T}$, which uses the small angle approximation on the definition of the quaternion. It is obvious that this quaternion will not conform to the unit norm constraint as the scalar part of the quaternion has a value of one, and is therefore a source of error in the MEKF. The error caused by this is what motivates the use of a norm-constrained MEKF, for which the equations are detailed in the following section. Also since the scalar portion of the error quaternion is approximated as constant, it permits a reduction of the state vector size which can reduce the computational complexity of the algorithm. In our implementation, we leave the state size as is because the full state is used in the norm-constrained MEKF, and it is useful for comparison between the two algorithms. However, in practice the actual state size is reduced by simply removing the scalar component of the error quaternion, the first column of $\mathbf{H}_k$, and the first row and column of $\mathbf{F}_{k-1}$. The quaternion and gyroscope bias estimates are then updated as

$$\hat{\mathbf{q}}_k^+ = \hat{\mathbf{q}}_k^- \otimes \delta\mathbf{q}_k^+ \tag{3.19}$$

$$\hat{\mathbf{b}}_{g|k}^+ = \hat{\mathbf{b}}_{g|k}^- + \delta\mathbf{b}_k^+ \tag{3.20}$$

TABLE 3.1: Equations for the MEKF

Prediction step:	**Correction step:**
$\hat{\boldsymbol{\omega}}_k^- = \mathbf{u}_g - \hat{\mathbf{b}}_{g\|k-1}^+$	$\hat{\mathbf{y}}_k = \begin{bmatrix} \hat{\mathbf{u}}_a \\ \hat{\mathbf{u}}_m \end{bmatrix}$, where $\hat{\mathbf{u}}_a$ and $\hat{\mathbf{u}}_m$ are found using (3.14)
$\hat{\mathbf{q}}_k^- = \boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}_k^-)\hat{\mathbf{q}}_{k-1}^+$	$\boldsymbol{\epsilon}_k = \mathbf{y}_k - \hat{\mathbf{y}}_k$
$\hat{\mathbf{b}}_{g\|k}^- = \hat{\mathbf{b}}_{g\|k-1}^+$	$\mathbf{K}_k = \begin{bmatrix} \mathbf{K}_{q\|k} \\ \mathbf{K}_{b\|k} \end{bmatrix} = \mathbf{P}_k^- \mathbf{H}_k^\mathsf{T} (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\mathsf{T} + \mathbf{R}_k)^{-1}$
$\mathbf{P}_k^- = \mathbf{F}_{k-1}\mathbf{P}_{k-1}^+\mathbf{F}_{k-1}^\mathsf{T} + \mathbf{Q}_{k-1}$	$\delta\mathbf{b}_{g\|k}^+ = \mathbf{K}_{b\|k}\boldsymbol{\epsilon}_k$
	$\delta\mathbf{q}_{upd}^+ = \mathbf{K}_{q\|k}\boldsymbol{\epsilon}_k$
	$\delta\mathbf{q}_k^+ = [1 \quad \frac{1}{2}\delta\boldsymbol{\varrho}_{upd}^+]^\mathsf{T}$
	$\hat{\mathbf{q}}_k^+ = \hat{\mathbf{q}}_k^- \otimes \delta\mathbf{q}_k^+$
	$\hat{\mathbf{b}}_{g\|k}^+ = \hat{\mathbf{b}}_{g\|k}^- + \delta\mathbf{b}_k^+$
	$\mathbf{P}_k^+ = (\mathbf{1} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{1} - \mathbf{K}_k\mathbf{H}_k)^\mathsf{T} + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^\mathsf{T}$

The quaternion is renormalized after this update, in order to correct for the norm constraint violation of the error quaternion. Finally the state covariance matrix is updated as

$$\mathbf{P}_k^+ = (\mathbf{1} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{1} - \mathbf{K}_k\mathbf{H}_k)^\mathsf{T} + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^\mathsf{T} \tag{3.21}$$

The consolidated prediction and correction equations for the MEKF are presented in Table 3.1. While in theory the state covariance matrix for the MEKF should remain symmetric, numerical errors can cause its symmetry to degrade. It is sound practice to enforce symmetry in the state covariance matrix at the end of each time step by using $\mathbf{P}_k^+ = \frac{1}{2}\left(\mathbf{P}_k^+ + \mathbf{P}_k^{+\mathsf{T}}\right)$.

3.2.1 Norm-Constrained MEKF

In order to better handle the quaternion norm constraint, Zanetti et al. use Lagrange multipliers to create a modified version of the Kalman gain which enforces the quaternion norm constraint [11]. They show that in some situations their norm-Constrained MEKF (CEKF) can give results comparable to a UKF and that it outperforms the classic MEKF. The CEKF uses the same basic algorithm as the MEKF, but recomputes a partitioned and constrained version of the Kalman gain before updating the quaternion and the state covariance matrix. It computes the constrained

TABLE 3.2: Equations for the CEKF

Prediction step:

$$\hat{\boldsymbol{\omega}}_k^- = \mathbf{u}_g - \hat{\mathbf{b}}_{g|k-1}^+$$

$$\hat{\mathbf{q}}_k^- = \boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}_k^-)\hat{\mathbf{q}}_{k-1}^*$$

$$\hat{\mathbf{b}}_{g|k}^- = \hat{\mathbf{b}}_{g|k-1}^+$$

$$\mathbf{P}_k^- = \mathbf{F}_{k-1}\mathbf{P}_{k-1}^*\mathbf{F}_{k-1}^\mathsf{T} + \mathbf{Q}_{k-1}$$

Correction step:

$$\hat{\mathbf{y}}_k = \begin{bmatrix} \hat{\mathbf{u}}_a \\ \hat{\mathbf{u}}_m \end{bmatrix}, \text{ where } \hat{\mathbf{u}}_a \text{ and } \hat{\mathbf{u}}_m \text{ are found using (3.14)}$$

$$\boldsymbol{\epsilon}_k = \mathbf{y}_k - \hat{\mathbf{y}}_k$$

$$\mathbf{K}_k = \begin{bmatrix} \mathbf{K}_{q|k} \\ \mathbf{K}_{b|k} \end{bmatrix} = \mathbf{P}_k^-\mathbf{H}_k^\mathsf{T}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^\mathsf{T} + \mathbf{R}_k)^{-1}$$

$$\delta\mathbf{b}_{g|k}^+ = \mathbf{K}_{b|k}\boldsymbol{\epsilon}_k$$

$$\delta\mathbf{q}_{upd}^+ = \mathbf{K}_{q|k}\boldsymbol{\epsilon}_k$$

$$\delta\mathbf{q}_k^+ = [1 \quad \tfrac{1}{2}\delta\boldsymbol{\varrho}_{upd}^+]^\mathsf{T}$$

Norm constraint step:

$$\mathbf{W}_k = \mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^\mathsf{T} + \mathbf{R}_k$$

$$\tilde{\epsilon}_k = \boldsymbol{\epsilon}_k^\mathsf{T}\mathbf{W}_k^{-1}\boldsymbol{\epsilon}_k$$

$$\mathbf{K}_{q|k}^* = \mathbf{K}_{q|k} + \left(\frac{1}{\|\delta\mathbf{q}_k^+\|} - 1\right)\delta\mathbf{q}_k^+\frac{\boldsymbol{\epsilon}_k^\mathsf{T}\mathbf{W}_k^{-1}}{\tilde{\epsilon}_k}$$

$$\mathbf{K}_k^* = \begin{bmatrix} \mathbf{K}_{q|k}^* \\ \mathbf{K}_{b|k} \end{bmatrix}$$

$$\mathbf{P}_k^* = (\mathbf{1} - \mathbf{K}_k^*\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{1} - \mathbf{K}_k^*\mathbf{H}_k)^\mathsf{T} + \mathbf{K}_k^*\mathbf{R}_k(\mathbf{K}_k^*)^\mathsf{T}$$

$$\delta\mathbf{q}_k^* = \mathbf{K}_{q|k}^*\boldsymbol{\epsilon}_k$$

$$\hat{\mathbf{q}}_k^+ = \hat{\mathbf{q}}_k^- \otimes \delta\mathbf{q}_k^*$$

$$\hat{\mathbf{b}}_{g|k}^+ = \hat{\mathbf{b}}_{g|k}^- + \delta\mathbf{b}_k^+$$

Kalman gain from the original one using:

$$\mathbf{W}_k = \mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^\mathsf{T} + \mathbf{R}_k \tag{3.22}$$

$$\tilde{\epsilon}_k = \boldsymbol{\epsilon}_k^\mathsf{T}\mathbf{W}_k^{-1}\boldsymbol{\epsilon}_k \tag{3.23}$$

$$\mathbf{K}_{q|k}^* = \mathbf{K}_{q|k} + \left(\frac{1}{\|\delta\mathbf{q}_k^+\|} - 1\right)\delta\mathbf{q}_k^+\frac{\boldsymbol{\epsilon}_k^\mathsf{T}\mathbf{W}_k^{-1}}{\tilde{\epsilon}_k} \tag{3.24}$$

$$\mathbf{K}_k^* = \begin{bmatrix} \mathbf{K}_{q|k}^* \\ \mathbf{K}_{b|k} \end{bmatrix} \tag{3.25}$$

where the * operator denotes the norm-constrained version of a variable. The constrained Kalman gain is then used to update the quaternion in the same manner as for the MEKF. The full algorithm for the CEKF is given in Table 3.2.

3.3 H_∞ Kalman Filter

The H_∞ or 'minmax' approach to state estimation can prove useful when an accurate model of the system is unavailable or when the noise statistics are unknown. H_∞ filters have been shown to

provide a better state estimate when there are errors in the noise and model parameters [9], [27]. There are many different approaches to developing filters which provide some bound on the H_∞ norm [39]. The minmax approach, originally derived by Yaesh and Shaked [40], uses a game theory framework where the adversary is trying to maximize an objective function using disturbances and sensor noise, and the designer is trying to minimize it.

The minmax H_∞ filter is a state space based filter which satisfies

$$\sup_{\boldsymbol{\eta}_k} \frac{\sum_{k=0}^{N} \|\mathbf{e}_k\|_2^2}{\sum_{k=0}^{N} \|\boldsymbol{\eta}_k\|_2^2} < \xi_k \tag{3.26}$$

where sup is the supremum and $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$ is the state error. Here $\boldsymbol{\eta}$ corresponds to the noise of all the sensors, so the estimator provides a bound on the ratio of the state error to the system noise. The design of the H_∞ filter is such that it reverts back to a basic MEKF when the value of ξ_k is set to ∞. The algorithm computes the Kalman gain in order to satisfy (3.26) as

$$\mathbf{D}_1 = (\mathbf{P}_k^- \mathbf{G}_k^\mathsf{T} (\mathbf{1} - \mathbf{G}_k \mathbf{P}_k^- \mathbf{G}_k^\mathsf{T})^{-1} \mathbf{G}_k + \mathbf{1}) \mathbf{P}_k^- \mathbf{H}_k^\mathsf{T} \tag{3.27}$$

$$\mathbf{D}_2 = \mathbf{H}_k \mathbf{P}_k^- \mathbf{G}_k^\mathsf{T} (\mathbf{1} - \mathbf{G}_k \mathbf{P}_k^- \mathbf{G}_k^\mathsf{T})^{-1} \mathbf{G}_k \mathbf{P}_k^- \mathbf{H}_k^\mathsf{T} + \mathbf{R}_k + \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\mathsf{T} \tag{3.28}$$

$$\mathbf{K}_k = \begin{bmatrix} \mathbf{K}_{q|k} \\ \mathbf{K}_{b|k} \end{bmatrix} = \mathbf{D}_1 \mathbf{D}_2^{-1} \tag{3.29}$$

where $\mathbf{G}$ is a user-defined matrix which incorporates the bound ξ_k and can be chosen to only apply the H_∞ bound to a subset of the state. The value of $\mathbf{G}$ is chosen as $\mathbf{G}_k = \frac{1}{\xi_k} \text{diag}(\mathbf{1}_4, \quad \mathbf{0}_{4\times 3})$ in order to only bound the quaternion. In order for the solution to this H_∞ filter to exist, the value of ξ_k must be chosen such that

$$\mathbf{1} - \mathbf{G}_k \mathbf{P}_k^+ \mathbf{G}_k^\mathsf{T} > 0 \tag{3.30}$$

$$\mathbf{1} + \mathbf{H}_k \mathbf{P}_k^+ \mathbf{H}_k^\mathsf{T} > 0 \tag{3.31}$$

where > 0 implies the matrix is positive definite and the state covariance $\mathbf{P}_k$ must also be non-singular. If these conditions are not met the filter may diverge. The state covariance update equations are also modified to enforce the H_∞ bound. As the values of $\mathbf{H}_k$ and $\mathbf{P}_k$ change at each timestep, a value of ξ_k is chosen through tuning such that it will satisfy the conditions for all likely

TABLE 3.3: Equations for the H_∞ filter

The prediction step of the H_∞ filter is the same as the MEKF (Table 3.1)

Correction step:

$$\mathbf{D}_1 = (\mathbf{P}_k^- \mathbf{G}_k^\intercal (\mathbf{1} - \mathbf{G}_k \mathbf{P}_k^- \mathbf{G}_k^\intercal)^{-1} \mathbf{G}_k + \mathbf{1}) \mathbf{P}_k^- \mathbf{H}_k^\intercal$$

$$\mathbf{D}_2 = \mathbf{H}_k \mathbf{P}_k^- \mathbf{G}_k^\intercal (\mathbf{1} - \mathbf{G}_k \mathbf{P}_k^- \mathbf{G}_k^\intercal)^{-1} \mathbf{G}_k \mathbf{P}_k^- \mathbf{H}_k^\intercal + \mathbf{R}_k + \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\intercal$$

$$\mathbf{K}_k = \begin{bmatrix} \mathbf{K}_{q|k} \\ \mathbf{K}_{b|k} \end{bmatrix} = \mathbf{D}_1 \mathbf{D}_2^{-1}$$

$$\boldsymbol{\epsilon}_k = \mathbf{y}_k - \hat{\mathbf{y}}_k$$

$$\delta \mathbf{b}_{g|k}^+ = \mathbf{K}_{b|k} \boldsymbol{\epsilon}_k$$

$$\delta \mathbf{q}_{upd}^+ = \mathbf{K}_{q|k} \boldsymbol{\epsilon}_k$$

$$\delta \mathbf{q}_k^+ = [1 \quad \tfrac{1}{2} \delta \boldsymbol{\varrho}_{upd}^+]^\intercal$$

$$\hat{\mathbf{q}}_k^+ = \hat{\mathbf{q}}_k^- \otimes \delta \mathbf{q}_k^+$$

$$\hat{\mathbf{b}}_{g|k}^+ = \hat{\mathbf{b}}_{g|k}^- + \delta \mathbf{b}_k^+$$

$$\mathbf{L}_k = (\mathbf{1} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- \mathbf{G}_k^\intercal (\mathbf{1} - \mathbf{G}_k \mathbf{P}_k^- \mathbf{G}_k^\intercal)^{-1}$$

$$\mathbf{P}_k^+ = (\mathbf{1} - \mathbf{K}_k \mathbf{H}_k + \mathbf{L}_k \mathbf{G}_k) \mathbf{P}_k^- (\mathbf{1} - \mathbf{K}_k \mathbf{H}_k + \mathbf{L}_k \mathbf{G}_k)^\intercal + \mathbf{K}_k \mathbf{R}_k (\mathbf{K}_k)^\intercal - \mathbf{L}_k \mathbf{L}_k^\intercal$$

$\mathbf{H}_k$ and $\mathbf{P}_k$. The equations for the H_∞ filter are given in Table 3.3. A norm-constrained version of the H_∞ filter has been derived by Chee and Forbes [27] but the equations will not be presented as it follows a similar methodology to the CEKF. However, the filter has been evaluated for a comparison with the conventional version in Chapter 4.

3.3.1 Novel Adaptive H_∞ Kalman Filter

The novel Adaptive H_∞ Filter ($AH_\infty F$) is proposed here with the knowledge that the H_∞ filter reduces to an MEKF as the H_∞ bound approaches infinity and that the robustness of the algorithm causes a loss in performance when the system parameters are well known [9], [27]. This is the general trade off seen with robust algorithms; there is only a performance improvement when part of the system is not accurately defined, otherwise, conventional methods can perform better. With this in mind, the $AH_\infty F$ is designed so that when the estimator has a large innovation for a significant amount of time, the H_∞ bound is lowered from infinity so that the system has some guaranteed bound on the state error. As the sensors experience increased noise and other anomalies during a collision, the innovation will be larger, which will activate the adaptation, bringing about a more robust estimate after a collision occurs. The $AH_\infty F$ activates its adaptation by summing the norm of a scaled innovation term over a series of time steps and if this sum is over a chosen threshold, it decreases the H_∞ bound from infinity. Likewise, if the summed innovation term is below the threshold, the H_∞ bound is increased towards infinity, bringing the estimator back to its original

form. The innovation sum, $\mathcal{I}_{d,k}$, is used to trigger and perform the adaptation as:

$$\mathcal{I}_{d,k} \geq \beta \rightarrow \frac{1}{\xi}_k = \min\left(\frac{1}{\xi}_{k-1} + e, \left(\frac{1}{\xi}\right)_{max}\right) \tag{3.32a}$$

$$\mathcal{I}_{d,k} < \beta \rightarrow \frac{1}{\xi}_k = \max\left(\frac{1}{\xi}_{k-1} - e, 0\right) \tag{3.32b}$$

where the innovation sum is computed by:

$$\mathcal{I}_{d,k} = \sum_{i=d+1}^{k} \Delta t \|\mathbf{\Gamma}\boldsymbol{\epsilon}_i\|_2^2. \tag{3.33}$$

The value of $k-d$ is the number of previous time steps for which the innovation is used and the value of e sets the rate at which the adaptive gain changes when the summed innovation is over a chosen threshold, β. The H_∞ bound is changed incrementally, as opposed to just having an adaptive term that is either on or off, as it is thought that this will provide a smoother transition for the adaptation. A max value for $\frac{1}{\xi}_k$ is also defined by $\left(\frac{1}{\xi}\right)_{max}$ so that the adaptation does not cause the H_∞ bound to change too much. The matrix $\mathbf{\Gamma}$ is a diagonal matrix that acts as a scaling factor in order to weight the measurement errors.

3.4 Unscented Kalman Filter

The UKF is designed to better approximate the non-linearities of a system by passing a set of optimally chosen points through the non-linear equations of the system, then computing a weighted average of the points to get an estimate of the state. UKFs have been shown to provide better state estimates for highly non-linear systems and systems with large initial estimate errors [14], [41]. The issue around using a conventional UKF to estimate quaternions is again due to violations of the quaternion norm constraint. The weighted average of the UKF sigma points used in computing the state estimate is additive in nature and cannot preserve the quaternion norm constraint. In order to avoid this, Crassidis [14] estimates an attitude error vector which can be summed without issue. The algorithm converts the attitude error vectors to and from error quaternions in order to propagate the sigma points and perform any measurement updates.

The UKF assumes the system has Gaussian noise and uses the state covariance to generate an optimal distribution of points to pass through the non-linear system equations. The sigma points at

the start of each timestep are generated by creating a modified state vector and covariance matrix using the noise mean and covariance as

$$\hat{\mathbf{x}}_k^+ = \begin{bmatrix} \delta\mathbf{p}_k^\mathsf{T} & \hat{\mathbf{b}}_{g|k}^\mathsf{T} & \hat{\boldsymbol{\eta}}^\mathsf{T} \end{bmatrix}^\mathsf{T} \tag{3.34}$$

$$\mathbf{P}_k^+ = \begin{bmatrix} \mathbf{P}_{\mathbf{x}|k} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{bmatrix} \tag{3.35}$$

where $\delta\mathbf{p}$ is the estimated error attitude vector, and $\hat{\boldsymbol{\eta}}$ is the mean of the noise variables (which is $\mathbf{0}$ in this case). At the beginning of each timestep the value of $\delta\mathbf{p}$ is assumed to be zero. Similarly to the MEKF, the state covariance matrix is dependent on the attitude error vector and is not directly related to the quaternion estimate. If the algorithm were estimating the quaternion instead of an attitude error vector, generating the sigma points would be another source of error as it is an additive process causing the quaternion sigma points to violate the quaternion norm constraint. Thus, the sigma points are generated for the attitude error vector, and converted into error quaternions. The conversions to and from the attitude error vector, $\delta\mathbf{p}$, from and to the error quaternion are given by Eqs. (3.36) and (3.37) respectively:

$$\delta\mathbf{p} = f[\delta\boldsymbol{\varrho}/(a + \delta q_0)] \tag{3.36}$$

and

$$\delta q_0 = \frac{-a\|\delta\mathbf{p}\|^2 + f\sqrt{f^2 + (1-a^2)\|\delta\mathbf{p}\|^2}}{f^2 + \|\delta\mathbf{p}\|^2} \tag{3.37a}$$

$$\delta\boldsymbol{\varrho} = f^{-1}(a + \delta q_0)\delta\mathbf{p} \tag{3.37b}$$

where the values of a and f can be changed to correspond to different attitude error vector parametrizations. Here we use a value of $f = 2(a+1)$ so that $\|\delta\mathbf{p}\|$ is equal to the error angle for small angles; then, a is chosen by tuning. The error quaternions can then be used with the actual quaternion to propagate the attitude. The sigma points are initialized at the start of each time step by

$$\boldsymbol{\mathcal{X}}_{k-1} = \left[\hat{\mathbf{x}}_{k-1}^+, \quad \hat{\mathbf{X}}_{k-1}^+ + \sqrt{(n+\lambda)\mathbf{P}_{k-1}^+}, \quad \hat{\mathbf{X}}_{k-1}^+ - \sqrt{(n+\lambda)\mathbf{P}_{k-1}^+}\right] \tag{3.38}$$

where n is the dimension of the modified state vector and λ is a function of a few parameters, acting as a scaling factor that determines the distribution of the sigma points. Its value is defined as $\lambda = \alpha^2(n + \kappa) - n$, where α determines the spread of the sigma points around the mean and κ is another scaling factor, usually set to zero [42]. The upper case version of the state vector, $\hat{\mathbf{X}}^+_{k-1}$, corresponds to an $n \times n$ matrix of state vectors, as $\hat{\mathbf{X}}^+_{k-1} = \left[\hat{\mathbf{x}}^+_{k-1}, \hat{\mathbf{x}}^+_{k-1}, ..., \hat{\mathbf{x}}^+_{k-1}\right]$. The $\sqrt{\mathbf{P}^+_{k-1}}$ operator indicates the square root of the matrix, which can be computed in a number of ways. Here we use the Cholesky decomposition of the matrix as it is a computationally efficient way to compute a square root for symmetric positive definite matrices. The sigma point matrix is segmented for use as

$$\boldsymbol{\mathcal{X}}_{k-1}(i) = \begin{bmatrix} \boldsymbol{\mathcal{X}}^{\delta p}_{k-1}(i) \\ \boldsymbol{\mathcal{X}}^{b_g}_{k-1}(i) \\ \boldsymbol{\mathcal{X}}^{\eta}_{k-1}(i) \end{bmatrix} \tag{3.39}$$

where i specifies the column of the sigma point matrix and the superscript denotes which value of the state vector it corresponds to. The attitude error vector sigma points are converted into error quaternion sigma points using (3.37); then, the error quaternion sigma points are multiplied with the current quaternion estimate as

$$\hat{\mathbf{q}}^+_{k-1}(1) = \hat{\mathbf{q}}^+_{k-1} \tag{3.40}$$

$$\hat{\mathbf{q}}^+_{k-1}(i) = \hat{\mathbf{q}}^+_{k-1} \otimes \delta\mathbf{q}^+_{k-1}(i), \quad i = 2, ..., 2n + 1 \tag{3.41}$$

These new quaternion sigma points are propagated using the quaternion kinematics along with the gyroscope measurement, gyroscope bias sigma points, and gyroscope noise sigma points, generating a set of predicted quaternion sigma points as:

$$\hat{\omega}^+_{k-1}(i) = \mathbf{u}_g - \boldsymbol{\mathcal{X}}^{b_g}_{k-1}(i) - \boldsymbol{\mathcal{X}}^{\eta_g}_{k-1}(i) \tag{3.42}$$

$$\hat{\mathbf{q}}^-_k(i) = \boldsymbol{\Omega}\left[\hat{\omega}^+_{k-1}(i)\right]\hat{\mathbf{q}}^+_{k-1}(i) \tag{3.43}$$

These are then converted into error quaternions, which can be converted into attitude error vectors, and then these attitude error vectors are used to compute the predicted state and state covariance matrices using the conventional UKF algorithms. The error quaternion and error attitude vector

sigma points are generated by

$$\delta\mathbf{q}_k^-(i) = \hat{\mathbf{q}}_k^-(1) \otimes \hat{\mathbf{q}}_k^-(i) \tag{3.44}$$

$$\boldsymbol{\mathcal{X}}_k^{\delta p}(i) = f[\delta\boldsymbol{\varrho}_k^-(i)/(a + \delta\hat{q}_{0|k}^-)] \tag{3.45}$$

The weighted average of these propagated sigma points gives the predicted state vector

$$\hat{\mathbf{x}}_k^- = \frac{1}{n+\lambda}\left[\lambda\boldsymbol{\mathcal{X}}_k(1) + \frac{1}{2}\sum_{i=1}^{2n}\boldsymbol{\mathcal{X}}_k(i)\right] \tag{3.46}$$

which is then used with the sigma points to compute the state covariance matrix

$$\mathbf{P}_k^- = \left(\frac{\lambda}{n+\lambda} + 1 - \alpha^2 + c\right)[\boldsymbol{\mathcal{X}}_k(1) - \hat{\mathbf{x}}_k^-][\boldsymbol{\mathcal{X}}_k(1) - \hat{\mathbf{x}}_k^-]^\mathsf{T} + \frac{1}{2(n+\lambda)}\sum_{i=1}^{2n}[\boldsymbol{\mathcal{X}}_k(i) - \hat{\mathbf{x}}_k^-][\boldsymbol{\mathcal{X}}_k(i) - \hat{\mathbf{x}}_k^-]^\mathsf{T} \tag{3.47}$$

The value c is a scaling factor which weights the first sigma point in the state covariance calculation. It is usually set to $c = 2$ when the prior of the noise is assumed to be Gaussian [42]. The rest of the UKF follows the conventional UKF algorithm except the final update of the quaternion, which again requires a conversion between the attitude error vector and quaternion. The full algorithm for the UKF can be found in Table 3.4.

It should be noted that the algorithm presented here slightly differs from the implementation presented by Crassidis, [14], as he presents a simplified version of the UKF which does not include the noise covariances in the generation of sigma points. Instead, the noise covariance matrices are added to the state covariance and innovation covariance matrices during their computation, using a modified form of the noise covariance matrices which is developed based on a trapezoidal integration. By doing this, Crassidis reduces the dimensions of the state vector and sigma point matrix, which reduces the computational complexity of algorithm. Here, the conventional UKF algorithm is followed as issues were encountered in getting Crassidis' algorithm to perform consistently and no comparisons of Crassidis' simplified implementation to the conventional algorithm were found. Other reductions in computational complexity can be realized by examining the lower triangular structure of the sigma point matrix. For example, due to this structure, many of the attitude error vector sigma points are identical, therefore the corresponding error quaternion sigma points will be identical and the conversion between the two only needs to be computed once. Also, for similar

TABLE 3.4: Equations for the UKF

Initialize:

$$\hat{\mathbf{x}}_0^+ = \begin{bmatrix} \delta\mathbf{p}_0 & \hat{\mathbf{b}}_{g|0} & \hat{\boldsymbol{\eta}} \end{bmatrix}^\mathsf{T}, \qquad \mathbf{P}_0^+ = \begin{bmatrix} \mathbf{P}_{\mathbf{x}|0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{bmatrix}$$

Prediction step:

Note: at the beginning of each step $\delta\mathbf{p}_k$ is set to $\mathbf{0}$. Generate sigma points:

$$\boldsymbol{\mathcal{X}}_{k-1} = \left[\hat{\mathbf{x}}_{k-1}^+, \quad \hat{\mathbf{X}}_{k-1}^+ + \sqrt{(n+\lambda)\mathbf{P}_{k-1}^+}, \quad \hat{\mathbf{X}}_{k-1}^+ - \sqrt{(n+\lambda)\mathbf{P}_{k-1}^+}\right]$$

Convert error vector sigma points to quaternion sigma points:

$$\delta\hat{q}_{0|k-1}^+(i) = \frac{-a\|\boldsymbol{\mathcal{X}}_{k-1}^{\delta p}(i)\|^2 + f\sqrt{f^2 + (1-a^2)\|\boldsymbol{\mathcal{X}}_{k-1}^{\delta p}(i)\|^2}}{f^2 + \|\boldsymbol{\mathcal{X}}_{k-1}^{\delta p}(i)\|^2}$$

$$\delta\boldsymbol{\varrho}_{k-1}^+(i) = f^{-1}(a + \delta\hat{q}_{0|k-1}(i))\boldsymbol{\mathcal{X}}_{k-1}^{\delta p}(i), \qquad \hat{\mathbf{q}}_{k-1}^+(1) = \hat{\mathbf{q}}_{k-1}^+$$

$$\hat{\mathbf{q}}_{k-1}^+(i) = \hat{\mathbf{q}}_{k-1}^+ \otimes \delta\mathbf{q}_{k-1}^+(i)$$

Propagate quaternions using (2.8):

$$\hat{\boldsymbol{\omega}}_{k-1}^+(i) = \mathbf{u}_g - \boldsymbol{\mathcal{X}}_{k-1}^{b_g}(i) - \boldsymbol{\mathcal{X}}_{k-1}^{\eta_g}(i), \qquad \hat{\mathbf{q}}_k^-(i) = \boldsymbol{\Omega}\left[\hat{\boldsymbol{\omega}}_{k-1}^+(i)\right]\hat{\mathbf{q}}_{k-1}^+(i)$$

Compute propagated error vector sigma points:

$$\delta\mathbf{q}_k^-(i) = \hat{\mathbf{q}}_k^-(1) \otimes \hat{\mathbf{q}}_k^-(i), \qquad \boldsymbol{\mathcal{X}}_k^{\delta p}(i) = f[\delta\boldsymbol{\varrho}_k^-(i)/(a + \delta\hat{q}_{0|k}^-)], \qquad \boldsymbol{\mathcal{X}}_k^{b_g}(i) = \boldsymbol{\mathcal{X}}_{k-1}^{b_g}(i)$$

Compute propagated state estimate and covariance:

$$\hat{\mathbf{x}}_k^- = \tfrac{1}{n+\lambda}\left[\lambda\boldsymbol{\mathcal{X}}_k(1) + \tfrac{1}{2}\textstyle\sum_{i=1}^{2n}\boldsymbol{\mathcal{X}}_k(i)\right]$$

$$\mathbf{P}_k^- = \left(\tfrac{\lambda}{n+\lambda} + 1 - \alpha^2 + c\right)[\boldsymbol{\mathcal{X}}_k(1) - \hat{\mathbf{x}}_k^-][\boldsymbol{\mathcal{X}}_k(1) - \hat{\mathbf{x}}_k^-]^\mathsf{T} + \tfrac{1}{2(n+\lambda)}\textstyle\sum_{i=1}^{2n}[\boldsymbol{\mathcal{X}}_k(i) - \hat{\mathbf{x}}_k^-][\boldsymbol{\mathcal{X}}_k(i) - \hat{\mathbf{x}}_k^-]^\mathsf{T}$$

Correction step:

$$\boldsymbol{\gamma}_k(i) = \begin{bmatrix} \mathbf{C}\left[\hat{\mathbf{q}}_k^-(i)\right]^\mathsf{T}\mathbf{g} \\ \mathbf{C}\left[\hat{\mathbf{q}}_k^-(i)\right]^\mathsf{T}\boldsymbol{\mu} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\mathcal{X}}_{k-1}^{\eta_a}(i) \\ \boldsymbol{\mathcal{X}}_{k-1}^{\eta_m}(i) \end{bmatrix}, \qquad \hat{\mathbf{y}}_k = \frac{1}{n+\lambda}\left\{\lambda\boldsymbol{\gamma}_k(1) + \tfrac{1}{2}\textstyle\sum_{i=1}^{2n}\boldsymbol{\gamma}_k(i)\right\}$$

Compute innovation and cross covariances:

$$\mathbf{P}_k^{yy} = \left(\tfrac{\lambda}{n+\lambda} + 1 - \alpha^2 + c\right)[\boldsymbol{\gamma}_k(1) - \hat{\mathbf{y}}_k][\boldsymbol{\gamma}_k(1) - \hat{\mathbf{y}}_k]^\mathsf{T} + \tfrac{1}{2(n+\lambda)}\textstyle\sum_{i=1}^{2n}[\boldsymbol{\gamma}_k(i) - \hat{\mathbf{y}}_k][\boldsymbol{\gamma}_k(i) - \hat{\mathbf{y}}_k]^\mathsf{T}$$

$$\mathbf{P}_k^{xy} = \left(\tfrac{\lambda}{n+\lambda} + 1 - \alpha^2 + c\right)[\boldsymbol{\mathcal{X}}_k(1) - \hat{\mathbf{x}}_k^-][\boldsymbol{\gamma}_k(1) - \hat{\mathbf{y}}_k]^\mathsf{T} + \tfrac{1}{2(n+\lambda)}\textstyle\sum_{i=1}^{2n}[\boldsymbol{\mathcal{X}}_k(i) - \hat{\mathbf{x}}_k^-][\boldsymbol{\gamma}_k(i) - \hat{\mathbf{y}}_k]^\mathsf{T}$$

Compute Kalman gain and update state:

$$\mathbf{K}_k = \mathbf{P}_k^{xy}(\mathbf{P}_k^{yy})^{-1}, \qquad \hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k)$$

Use $\delta\mathbf{p}_k^+$ with (3.37) to generate error quaternion $\delta\mathbf{q}_k^+$ then update quaternion and the state covariance:

$$\hat{\mathbf{q}}_k^+ = \hat{\mathbf{q}}_k^-(1) \otimes \delta\mathbf{q}_k^+, \qquad \mathbf{P}_k^+ = \mathbf{P}_k^- - \mathbf{K}_k\mathbf{P}_k^{yy}\mathbf{K}_k^\mathsf{T}$$

reasons as the MEKF it is good practice to enforce state covariance matrix symmetry at the end of each time step as $\mathbf{P}_k^+ = \frac{1}{2}\left(\mathbf{P}_k^+ + \mathbf{P}_k^{+\mathsf{T}}\right)$.

3.4.1 Square Root UKF

The SRUKF improves upon the conventional UKF by propagating the Cholesky decomposition of the state covariance instead of the actual state covariance matrix. This ensures that the state covariance matrix will be symmetric positive definite, increasing numerical accuracy of the algorithm [13]. The SRUKF uses a QR decomposition of the weighted sigma points in order to update the Cholesky factor of the state covariance. It then performs a Cholesky update (or downdate) on the R matrix of the QR decomposition, which is required to be done separately as the weight of the first sigma point vector might be negative (which could cause issues with the QR decomposition). The SRUKF implemented here follows the algorithm used in [43], which more closely resembles the algorithm of the conventional UKF, than the original derivation of SRUKF presented by Van Der Merwe and Wan [13]. This is mostly due to issues encountered while getting Van Der Merwe and Wan's implementation to perform consistently.

The SRUKF initializes the Cholesky decomposition $\mathbf{S}_{\mathbf{x}|0}^+ = \sqrt{\mathbf{P}_{\mathbf{x}|0}}$ and generates the sigma points at each time step as

$$\mathbf{S}_{k-1}^+ = \begin{bmatrix} \mathbf{S}_{\mathbf{x}|k-1}^+ & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sqrt{\mathbf{Q}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \sqrt{\mathbf{R}} \end{bmatrix} \tag{3.48}$$

$$\boldsymbol{\mathcal{X}}_{k-1} = \left[\hat{\mathbf{x}}_{k-1}^+, \quad \hat{\mathbf{x}}_{k-1}^+ + \sqrt{(n+\lambda)}\mathbf{S}_{k-1}^+, \quad \hat{\mathbf{x}}_{k-1}^+ - \sqrt{(n+\lambda)}\mathbf{S}_{k-1}^+\right] \tag{3.49}$$

Here, the values of $\sqrt{\mathbf{Q}}$ and $\sqrt{\mathbf{R}}$ are simple to compute as the matrices are diagonal. Then, the Cholesky decomposition is updated using the propagated sigma points and predicted state by

$$\mathbf{S}_k^{upd} = qr\left\{\sqrt{\frac{1}{2(n+\lambda)}}\left[\boldsymbol{\mathcal{X}}_k(i) - \hat{\mathbf{x}}_k^-\right]\right\}, \quad i = 2:2n+1 \tag{3.50a}$$

$$\mathbf{S}_k^- = cholupdate\left\{\mathbf{S}_k^{upd}, \quad \left[\boldsymbol{\mathcal{X}}_k(1) - \hat{\mathbf{x}}_k^-\right], \quad \sqrt{\frac{\lambda}{n+\lambda} + 1 - \alpha^2 + c}\right\} \tag{3.50b}$$

where $qr\{\mathbf{A}\}$ gives the $\mathbf{R}$ matrix of a QR decomposition of $\mathbf{A}$ and $cholupdate\{\mathbf{R}, \mathbf{X}, b\}$ gives the Cholesky factor of $\mathbf{A} + b\mathbf{X}\mathbf{X}^\mathsf{T}$ where $\mathbf{R} = chol\{\mathbf{A}\}$. The measurement sigma points are computed in the same manner as for the UKF, but the innovation covariance is instead computed using a QR decomposition as

$$\mathbf{S}_k^{yy-} = qr\left\{\sqrt{\frac{1}{2(n+\lambda)}}\left[\boldsymbol{\gamma}_k(i) - \hat{\mathbf{y}}_k\right]\right\}, \quad i = 2:2n+1 \tag{3.51a}$$

$$\mathbf{S}_k^{yy} = cholupdate\left\{\mathbf{S}_k^{yy-}, \quad \left[\boldsymbol{\gamma}_k(1) - \hat{\mathbf{y}}_k\right], \quad \sqrt{\frac{\lambda}{n+\lambda} + 1 - \alpha^2 + c}\right\} \tag{3.51b}$$

Then the Kalman gain is computed as

$$\mathbf{K}_k = \mathbf{P}_k^{xy} / (\mathbf{S}_k^{yy\mathsf{T}}) / \mathbf{S}_k^{yy} \tag{3.52}$$

and the Cholesky decomposition of the state covariance is updated using

$$\mathbf{U} = \mathbf{K}_k \mathbf{S}_k^{yy} \tag{3.53}$$

$$\mathbf{S}_{\mathbf{x}|k}^{+} = cholupdate\left\{\mathbf{S}_{\mathbf{x}|k}^{-}, \quad \mathbf{U}, \quad 1\right\} \tag{3.54}$$

where $\mathbf{S}_{\mathbf{x}|k}^{-}$ is the block of $\mathbf{S}_k^{-}$ corresponding to the state.

3.4.2 Novel High Gain Adaptive UKF

The novel high gain AUKF is based on an adaptive EKF designed by Sebesta et al. [23]. They show their estimator is more responsive to aggressive manoeuvres and disturbances than a conventional EKF. However, their design is computationally demanding as it requires integrating an initial value problem several times in order to modify the adaptive gain. In the simplified version derived here, the same innovation measure that is used in the AH_∞ filter is used to trigger the adaptive part of the filter. A single adaptive gain term, G_k, is used to scale the noise covariance matrices. The scaling method is designed to increase the noise covariances for the gyroscope and

accelerometer. The value of G_k changes at each time step based on

$$\mathcal{I}_{d,k} \geq \beta \rightarrow G_k = \min\left(G_k + e, G_{max}\right) \tag{3.55a}$$

$$\mathcal{I}_{d,k} < \beta \rightarrow G_k = \max\left(G_k - e, 1\right) \tag{3.55b}$$

where $\mathcal{I}_{d,k}$ is found using (3.33), and the other variables have the same effects as their counterparts in (3.32). The adaptive filter then replaces $\mathbf{R}_k$ and $\mathbf{Q}_k$ with

$$\mathbf{Q}_k^G = \Delta_G \mathbf{Q}_k \tag{3.56}$$

$$\mathbf{R}_k^G = \delta_G \mathbf{R}_k \tag{3.57}$$

where Δ_G and δ_G were chosen as

$$\Delta_G = \text{diag}[G_k, G_k, G_k, G_k^{-1}, G_k^{-1}, G_k^{-1}] \tag{3.58}$$

$$\delta_G = \text{diag}[G_k, G_k, G_k, 1, 1, 1] \tag{3.59}$$

so that they increase the noise covariance for the inertial sensors. This method also decreases the noise covariance of the gyroscope bias, as this was found to yield better results.

3.4.3 Novel Covariance Matching Adaptive UKF

The second novel adaptive method modifies the measurement noise covariance matrix $\mathbf{R}$ by adding a term based on the difference between the measured innovation covariance and the estimated innovation covariance. This method modifies the measurement noise covariance using the following terms:

$$\mathbf{R}_k^G = \mathbf{R}_k + \frac{1}{d} \sum_{i=k-d}^{k} \left(\tilde{\mathbf{S}}_i\right) \tag{3.60}$$

where

$$\mathbf{S}_k = (\boldsymbol{\epsilon}_k \boldsymbol{\epsilon}_k^\mathsf{T} - \mathbf{P}_k^{yy}) \tag{3.61}$$

$$\tilde{\mathbf{S}}_k = \begin{bmatrix} \tilde{s}_{11} & \dots & \tilde{s}_{1j} \\ \vdots & \ddots & \vdots \\ \tilde{s}_{i1} & \dots & \tilde{s}_{ij} \end{bmatrix} \tag{3.62}$$

$$\tilde{s}_{ij} = \begin{cases} s_{ij} & \text{for} \quad s_{ij} \geq 0 \quad \& \quad i = j \\ 0 & \text{for} \quad s_{ij} < 0 \quad | \quad i \neq j \end{cases} \tag{3.63}$$

The value of d again sets the number of previous time steps for which data is used and the same value of d is used in simulation for all adaptive algorithms. Equation (3.60) modifies the measurement noise covariance matrix using the average of the difference between the predicted and measured innovation covariance matrices. Equations (3.61)-(3.63) compute the difference between the predicted and measured innovation covariance matrices, and then sets any off-diagonal or negative entries to zero. While this does not adapt the gyroscope noise covariance, it still modifies the accelerometer noise covariance to account for the increase in noise from crashing. This method will also slightly affect the magnetometer noise covariance, but in theory should keep it around the same value. Since this method uses the innovation covariance $\mathbf{P}_k^{yy}$ to modify $\mathbf{R}_k$, it requires less tuning than the high gain AUKF which needs more user-tuned parameters.

3.5 PX4 Complementary Filter

The PX4 flight stack is an open source professional autopilot software which offers a split architecture for providing attitude and position estimates, allowing the attitude estimator to be easily replaced and compared with custom code. The PX4 attitude estimator is a type of non-linear observer, also dubbed a complementary filter in the code comments. As it is open source code, the documentation for the estimation algorithm is scarce so the algorithm presented here has been reverse engineered from the PX4 code.

The PX4 complementary filter is structured similarly to the complementary filter detailed in Section 3.1. One main difference is that the PX4 filter uses the magnetometer reading to only correct the vehicle's yaw. This is done in some estimator implementations as the magnetic field

reading can be significantly influenced by high currents and metallic objects near the vehicle which can catastrophically corrupt the pitch and roll estimates [44], [45]. This practice can be wasteful if there are no significant disturbances to the magnetic field reading, but can be beneficial to stability if there are. As the roll and pitch of the vehicle can be ascertained using the accelerometer, the magnetometer measurement can be used to correct the less critical of the three Euler angles, the yaw, which generally can not be obtained from accelerometer data alone. The PX4 filter is also different in that it uses the accelerometer reading regardless of whether or not the vehicle is experiencing any linear accelerations. While this would provide more measurements to use for estimation purposes, the measurements would be less accurate in this application.

The PX4 filter propagates the quaternion almost identically to (3.1) as

$$\hat{\mathbf{q}}_k = \boldsymbol{\Omega}(\mathbf{u}_g + \hat{\mathbf{b}}_g + \boldsymbol{\omega}_{mes})\hat{\mathbf{q}}_{k-1} \tag{3.64}$$

however, with a simpler correction term:

$$\boldsymbol{\omega}_{mes} = k_a \hat{\mathbf{u}}_a^{\times} \bar{\mathbf{u}}_a + k_m \hat{\boldsymbol{\mu}}_b \tag{3.65}$$

The values k_a and k_m are tuning gains, $\hat{\mathbf{u}}_a$ is the predicted gravitational vector (computed as in equation (3.3a)) and $\hat{\boldsymbol{\mu}}_b$ is a correction term generated from the magnetometer reading. In particular, the magnetometer correction term is computed using the knowledge that the inertial x-axis lines up with magnetic north. The correction term rotates the magnetometer reading into the inertial frame (Eq. (3.66)), and then uses the tan inverse of its x and y components to find the error between the predicted and actual inertial frames (Eq. (3.67)). It then rotates the correction factor into the body frame in order to apply it (Eq. (3.68)):

$$\hat{\boldsymbol{\mu}}_i = \mathbf{C}(\hat{\mathbf{q}}_{k-1})\bar{\mathbf{u}}_m \tag{3.66}$$

$$e_{\boldsymbol{\mu}} = \text{atan2}(\hat{\mu}_{i|2}, \quad \hat{\mu}_{i|1}) \tag{3.67}$$

$$\hat{\boldsymbol{\mu}}_b = \mathbf{C}(\hat{\mathbf{q}}_{k-1})^{\mathsf{T}} \begin{bmatrix} 0 \\ 0 \\ -e_{\boldsymbol{\mu}} \end{bmatrix} \tag{3.68}$$

where $\hat{\mu}_{i|1}$ and $\hat{\mu}_{i|2}$ refer to the first and second values of $\hat{\boldsymbol{\mu}}_i$. The gyroscope bias is then updated by

$$\hat{\mathbf{b}}_{g|k} = \hat{\mathbf{b}}_{g|k-1} + k_b \boldsymbol{\omega}_{mes} \Delta t \tag{3.69}$$

where k_b is a tuning gain.

The PX4 implementation uses special provisions for when the vehicle undergoes fast rotations. In particular, the PX4 filter does not update the gyroscope bias if the magnitude of the gyroscope measurement is greater than 10 deg/s and it modifies the magnetometer gain k_m, such that

$$k_m = k_m * \min(\|\mathbf{u}_g\|/0.873, \quad 10) \tag{3.70}$$

if the magnitude of the gyroscope measurement is greater than 50 deg/s (or 0.873 rad/s). The motivation behind increasing the yaw correction gain for high vehicle spin rates originates from experiments conducted by the PX4 developers. These experiments showed that misalignment errors in the gyroscope created large yaw deviations if using the gyroscope to propagate the quaternion and the vehicle is spinning at high rates; it was found that in this situation better results could be obtained by increasing magnetometer gain. The results of these experiments are unpublished in a shorthand Dropbox document [46].

Chapter 4

Validation in Simulation

The seven estimation algorithms presented in detail in Chapter 3 are first validated in simulation, prior to testing them with actual sensor data or using them in the control loop of a quadrotor UAV. This is done for a number of reasons: testing in simulation is vital to ensure the algorithms perform as expected especially considering that quadcopters are inherently unstable and a poor attitude estimate can result in catastrophic failures. As a large number of estimation algorithms are under investigation it is not viable to implement and test them all experimentally. This is particularly true in the case of attitude estimation for collision recovery, where the more experiments are performed, the more likely the quadcopter will be damaged. Testing with simulated data is also important for debugging the algorithms, as using simulated values makes it easier to control the data, and track down where issues may originate.

The two main stochastic factors which affect the performance of state estimation algorithms are sensor noise and Initial Condition (IC) error. Comparing the estimation algorithms using a single or just a few sets of data does not always provide an accurate representation of how well the estimation algorithms perform compared to each other, as the randomness may result in uncharacteristic performance by certain algorithms. The estimation algorithms need to be validated and compared across a large number of data sets, with a wide variety of IC errors. The Monte Carlo method compares algorithms by averaging the performance of an algorithm over a large number of data sets, using randomized values for ICs, disturbances in the system, or other stochastic variables which may affect their performance. As the number of data sets increases, the average performance across all data sets should converge towards a value from which the performance of the estimation algorithms can be compared reliably.

In the simulations presented in this chapter, the performance of the estimation algorithms is

compared when faced with random initial attitude estimate errors, random initial sensor biases and random sensor noise. Errors in the initial state estimate can come from a variety of causes. Generally, the initial attitude estimate for a vehicle is either based on an assumption about the vehicle's initial orientation, or it combines the very first set of sensor readings to form a guess of the vehicle's orientation. Many issues can contribute to initial attitude estimate errors such as start up anomalies in the sensors, noise in the sensors, the vehicle may not be completely still during initialization, or assumptions about the attitude may be wrong. The vehicle is also usually assumed to have zero initial bias in its sensors, which generally is not true. Sensor bias can develop from a change in temperature as the vehicle is moved from indoors to outdoors, from shade to sun, or as the electronics heat up over the course of operation. Sensor bias can also exist due to mechanical strain on the sensor, or calibration errors.

To validate and compare the estimation algorithms, three simulation scenarios are investigated. In the first two simulation scenarios, the quadrotor takes off and then flies in a 2m square trajectory over 60 seconds, then hovers for 15 seconds and the simulation ends. The first square trajectory scenario has lower IC errors, whereas the second scenario has higher IC errors. The first scenario compares the estimation algorithms performance in normal flight under what can be considered as average initialization conditions (such as moving the quadcopter from indoors to outdoors on a summer day), and the second scenario compares them under extreme initialization conditions (such as moving from indoors to outdoors on a cold winter day). In the third simulation scenario, the quadrotor is given the same low IC error as in the first scenario, it flies along the same square trajectory but then crashes into a wall. The quadcopter impacts the wall at 60 seconds in the collision scenario, and the simulation is ended after 75 seconds for consistency with scenarios one and two. A scenario ending in a collision but with high IC error is not investigated as the hypothetical scenario involving a collision is one where the UAV has been in flight for a while prior to it; thus, the attitude estimate has had time to converge. In the simulations, the IC errors are generated from zero mean Gaussian distributions where the standard deviation is chosen based on the Pixhawk sensor data sheets, experimental sensor data and assumptions about the scenario. All three scenarios are simulated using the quadrotor parameters and system dynamics from [5] and for the third scenario, using the collision recovery controller from [6]. The trajectory and attitude controller for normal flight are quaternion based waypoint controllers derived in [47]. In order to have a consistent

TABLE 4.1: Sensor Noise Characteristics

$\tilde{\sigma}_{a0} =$	$1\ [m/s^2]$	$\tilde{\sigma}_{ac} =$	$14\ [m/s^2]$
$\tilde{\sigma}_{g0} =$	$0.0018\ [rad/s]$	$\tilde{\sigma}_{gc} =$	$0.1\ [rad/s]$
$\sigma_m =$	$0.007\ [G]$	$\tau =$	$0.25\ [sec]$

comparison of the estimators, the controllers use the true quadrotor states as feedback as opposed to values from an estimator. Comparing the algorithms while using each algorithm's estimate in the controller feedback loop will be done in experiments in Section 5.3.

The initial attitude estimate error was chosen as $\hat{\mathbf{q}}_0 = [\cos(\phi_0) \quad \mathbf{a_0}\sin(\phi_0)]^\mathsf{T}$ such that $\phi_0 = \mathcal{N}(0, \tilde{\sigma}_\phi)$ degrees and $\mathbf{a}_0 = \mathbf{a}'_0/\|\mathbf{a}'_0\|$ where the three components of $\mathbf{a}'_0$ are chosen such that $a'_{0i} = \mathcal{U}(-1, 1)$, with $\mathcal{U}$ being the uniform distribution. The rotation axis is chosen from the uniform distribution as the initial attitude estimate error is equally likely to be in any direction. The initial sensor bias values are chosen as $\mathbf{b}_{s|0} = \mathcal{N}(0, \tilde{\sigma}_{b_s})$ where $s \in \{a, g, m\}$ for the accelerometer, gyroscope or magnetometer biases respectively. The noise characteristics used in the sensor models are given in Table 4.1. The values for the ICs which vary for each scenario are given in Table 4.2 and the values for the ICs which are constant across all scenarios are given in Table 4.3. The values of the tuned parameters for each estimator are summarized in Table 4.4. Tunable parameters were chosen based on the estimators' performance in the low IC scenarios. They were chosen by using a small set of constant IC values, and then varying the tunable parameters through certain ranges. A better method for tuning the parameters would be to perform a set of Monte Carlo simulations for each set of parameter gains investigated, but this would require a multitude of simulations which is beyond the scope of this work. By using a constant set of IC error values while varying the parameters, the effect of the parameters could be compared using a smaller set of data. The parameters which provided the best results were then used for all simulation scenarios. The estimation algorithms are provided data at a rate of 100Hz in simulation, which was chosen based on the sensor rates of the logged data onboard the PixHawk flight controller.

4.1 Results

The results of the simulations are shown in Figures 4.1–4.6 and Tables 4.5–4.10. Figures 4.1, 4.3 and 4.5 present the average Root Mean Square (RMS) total angle error and the average RMS Euler angle errors across all scenarios. The upper and lower graphs in each of these figures present

TABLE 4.2: Scenario specific IC statistics

Scenario		$\tilde{\sigma}_\phi \quad [deg]$	$\tilde{\sigma}_{b_g} \quad [rad/s]$
1	Low Initial Error, Square	1	0.01
2	High Initial Error, Square	5	0.04
3	Low Initial Error, Crash	1	0.01

TABLE 4.3: ICs for all scenarios

CEKF and H_∞ filters	
$\mathbf{P}_{x\|0}$ =diag[.025, .025, .025, .025 .001, .001, .001]	
$\mathbf{Q}$ =diag[$\tilde{\sigma}_{g0}\mathbf{1}_{1\times4}$, $\tilde{\sigma}_{b_g}\mathbf{1}_{1\times3}$]	$\mathbf{R}$ =diag[$\tilde{\sigma}_{a0}\mathbf{1}_{1\times4}$, $\sigma_m\mathbf{1}_{1\times3}$]
UKFs	
$\mathbf{P}_{x\|0}$ = diag[.025, .025, .025, .001, .001, .001]	
$\mathbf{Q}$ =diag[$\tilde{\sigma}_{g0}\mathbf{1}_{1\times3}$, $\tilde{\sigma}_{b_g}\mathbf{1}_{1\times3}$]	$\mathbf{R}$ =diag[$\tilde{\sigma}_{a0}\mathbf{1}_{1\times3}$, $\sigma_m\mathbf{1}_{1\times3}$]
All filters	
$\tilde{\sigma}_{b_a} = 0.05 \quad [m/s^2]$	$\tilde{\sigma}_{b_m} = 0.005 \quad [Gs]$

TABLE 4.4: Estimator specific parameters

Complementary Filter	$k_P = 1$	$k_a = 0.15$
	$k_m = 0.25$	$k_b = 0.25$
UKF	κ = -6	$a = 1$
	$\alpha = 0.5$	$c = 2$
H_∞ filter	$\frac{1}{\xi} = 0.1$	
High Gain AUKF	$G_{max} = 5$	$d = 15$
	$e = 0.25$	$\beta = 0.25$
	$\mathbf{\Gamma}$ = diag[$\mathbf{1}_{1\times3}$, 30 $\mathbf{1}_{1\times3}$]	
AH_∞ filter	$e = 0.01$	$d = 15$
	$\left(\frac{1}{\xi}\right)_{max} = 0.2$	

the same data, but the lower graph omits the results of the complementary filter in order to provide better scaling of the graph for the comparison of the other algorithms. Figures 4.2, 4.4 and 4.6 give the average RMS gyroscope bias error. Each of the scenarios uses data from 200 simulated flights to compute the averages. For a given simulation run, the RMS Euler angle error is the RMS of the difference between the true and estimated Euler angles. Tables 4.5, 4.7 and 4.9 present the standard deviation of the RMS total angle and RMS Euler angle error averages corresponding to the results in Figures 4.1, 4.3 and 4.5, respectively. Tables 4.6, 4.8 and 4.10 present the standard deviation of the RMS gyroscope bias error averages which are presented in Figures 4.2, 4.4 and 4.6, respectively. The RMS total angle error, $\delta\phi$, is calculated as

$$\delta\phi_k = 2\cos^{-1}(\delta q_{0,k}) \tag{4.1a}$$

$$\delta\phi = \sqrt{\sum_{k=1}^{N} \frac{\delta\phi_k^2}{N}} \tag{4.1b}$$

and the RMS gyroscope bias error is calculated as the RMS of the difference between the true and estimated gyroscope bias for each of the x-, y-, and z-axes of the quadcopter.

4.1.1 Scenario One Results

Figures 4.1–4.2 and Tables 4.5–4.6 present the attitude estimation error statistics for scenario one - a square trajectory with low IC errors - where it can be seen that the complementary filter provides the least accurate attitude estimate, by a significant margin, with average total attitude errors over 10°. The H_∞, MEKF and UKF filters provide similar attitude estimates, with average total attitude errors of 2.66°, 2.40° and 2.23°, respectively. The H_∞ filter performed slightly worse than the MEKF, which is expected under normal flight conditions due to its robust nature. While the H_∞ filter has a slightly worse average estimation error, Table 4.5 shows that its average estimation error has the lowest standard deviation. This means the H_∞ filter's robustness slightly degrades the accuracy of algorithm, but slightly improves its consistency. Table 4.5 also shows that the MEKF has a slightly lower standard deviation than the UKF and that the complementary filter has a much larger standard deviation than the other algorithms. The MEKF is shown to have the best gyroscope bias estimate but by a small margin compared to the H_∞ and UKF gyroscope bias estimates. The standard deviations of the average gyroscope bias errors are very similar for

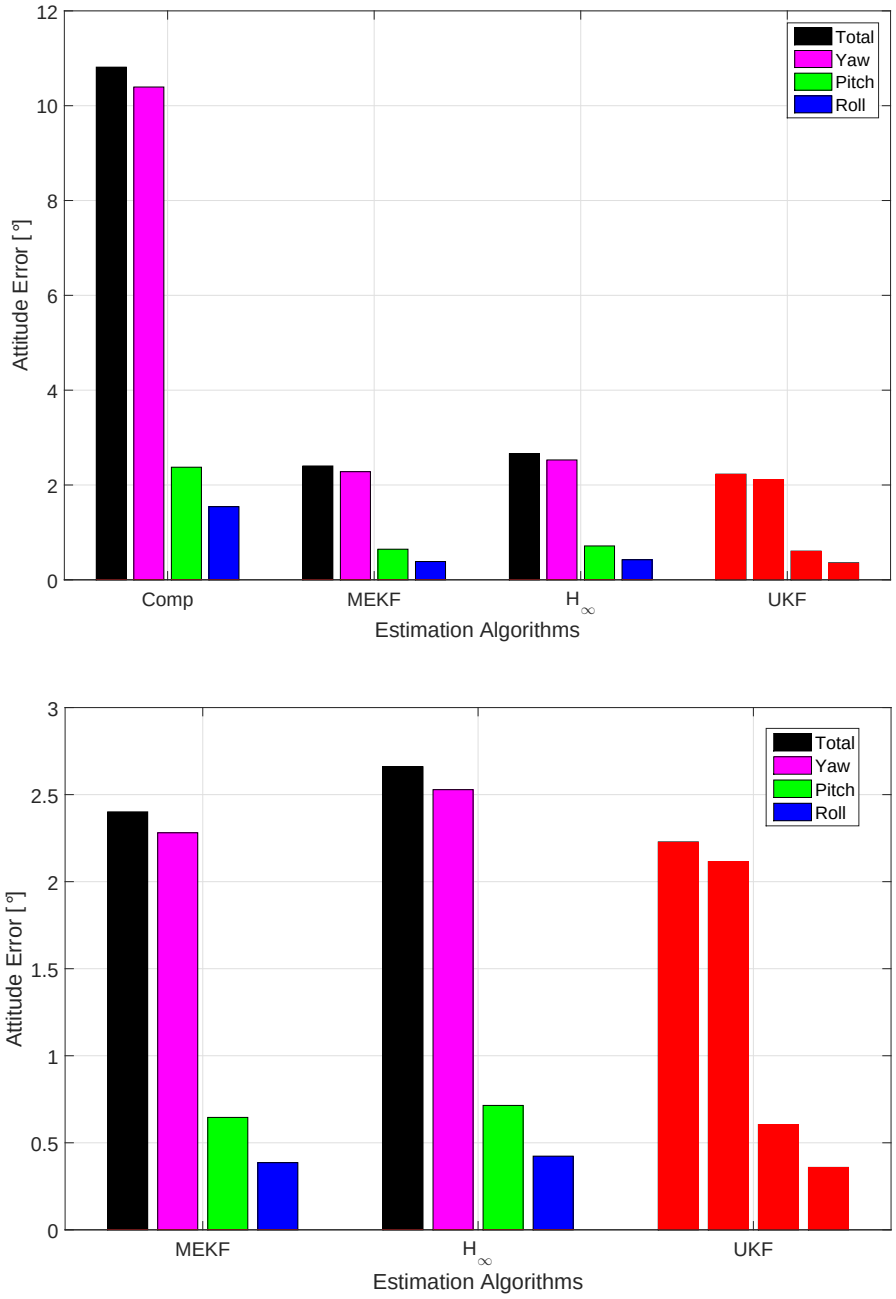

FIGURE 4.1: Average RMS total and Euler angle error for scenario 1 (low ICs, no collision) of 200 simulated flights. Red bars correspond to the lowest error. The upper and lower graphs contain the same data but the lower figure does not include the complementary filter data for clarity

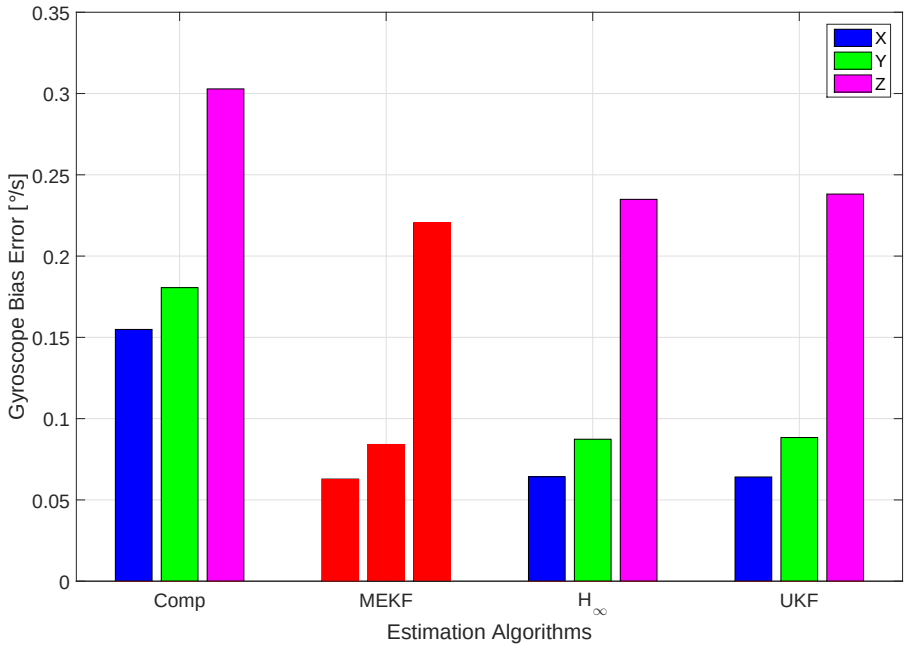

FIGURE 4.2: Average RMS gyroscope bias error for scenario 1 (low ICs, no collision) of 200 simulated flights. Red bars correspond to the lowest error

the Kalman based filters, with the UKF performing slightly better, while the complementary filter has the highest standard deviation. The results of the adaptive algorithms were not significantly different from the results of the basic estimation algorithms during the square trajectory, and are therefore omitted from Figures 4.1 and 4.2.

It can be seen that for all estimation algorithms the predominant attitude estimate error is along the yaw component and that the estimate of the z component of gyroscope bias is significantly worse compared to the others. The largest standard deviation in the errors also occurs along the yaw component of the attitude and z component of the gyroscope bias. This is likely due to the observability issues caused by the alignment of the gravitational vector with the body z-axis which occurs frequently in this maneouvre. Hong et al. found that the gyroscope bias along the direction of gravity tends to be unobservable when the vehicle is not experiencing large changes in attitude or acceleration, on a vehicle equiped with an accelerometer, gyroscope and single GPS [48]. As can be deduced from the results here, the use of a magnetometer as well as an accelerometer as attitude measurements give a fully observable system. However, as the accelerometer vector tends to be aligned with the yaw axis, it is likely that the pitch and roll angles are more observable than the rotation along the yaw axis. Another factor that could contribute to the yaw estimate error is

TABLE 4.5: Standard deviation of RMS angle error for entire flight of scenario 1. Bold values signify the lowest values.

Algorithm	Standard deviation [°]			
	Total	Yaw	Pitch	Roll
Complementary	8.73	8.60	1.70	1.09
MEKF	0.475	0.450	0.131	0.078
H_∞	**0.436**	**0.413**	**0.120**	**0.073**
UKF	0.530	0.503	0.143	0.089

the direction of the magnetic field vector. Since the value of the magnetic field vector used here has a large z component, noise in the horizontal components of the magnetometer would have a larger effect, as the signal to noise ratio would be smaller for the horizontal components of the magnetic field. Future work could investigate if the use of a magnetic field vector from near the equator, which tends to have a much smaller downward component can reduce the yaw error. It is also noticeable that the pitch estimate is consistently slightly worse than the roll estimate, and the y-axis gyroscope bias estimate is slightly worse than the x-axis estimate. The exact reason for this is unknown, but some hypotheses are suggested. One possibility is that the direction in which the magnetic field points also causes a smaller signal to noise ratio along the y-axis of the vehicle, resulting in a larger error in the pitch and y-axis gyroscope bias estimate. Another possibility is that it is a result of how error is propagated through the 3 sequential rotations of the Euler angles; since the yaw rotation estimate has large errors, this error might cause error in the next rotation in the sequence, the pitch estimate. However, a more thorough analysis is required to confirm any of these hypotheses.

4.1.2 Scenario Two Results

The results for scenario two are shown in Figures 4.3–4.4 and Tables 4.7–4.8. The main difference from the first scenario is that the complementary filter has very poor performance in estimating both attitude and gyroscope bias, and appears to directly propagate the increase in the initial errors, with its total attitude estimate error exceeding 45°. The MEKF and the H_∞ filter both show an increase of about 40% in the error in their estimates compared to scenario one, with total attitude

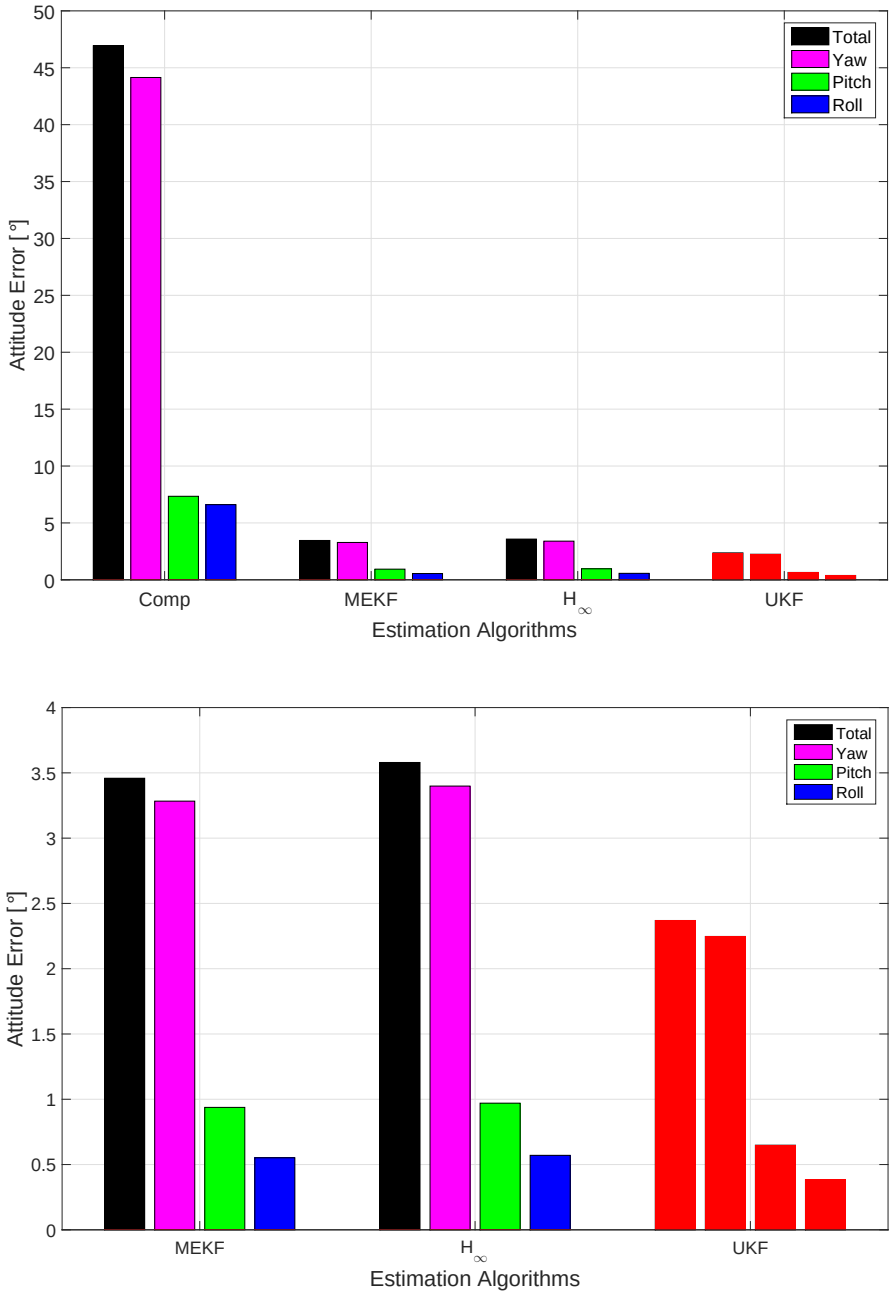

FIGURE 4.3: Average RMS total and Euler angle error for scenario 2 (high ICs, no collision) of 200 simulated flights. Red bars correspond to the lowest error. The upper and lower graphs contain the same data but the lower figure does not include the complementary filter data for clarity

TABLE 4.6: Standard deviation of RMS gyroscope bias error for entire flight of scenario 1. Bold values signify the lowest values.

Algorithm	Standard deviation [°/s]		
	X	Y	Z
Complementary	0.089	0.095	0.235
MEKF	0.025	0.029	0.074
H_∞	0.024	0.028	**0.069**
UKF	**0.021**	**0.026**	0.074

estimate errors of 3.46° and 3.58°, respectively. The UKF shows a slight increase in error compared to scenario one, with a total attitude estimate error of 2.37°, but with a more significant increase in the gyroscope bias estimate error. These results are mirrored in the standard deviations of the attitude errors as the complementary filter shows a very large increase, the MEKF and H_∞ filter standard deviations almost triple, and the UKF attitude error standard deviation only increases slightly compared to scenario one. The standard deviations of the gyroscope bias follows the same trends between estimation algorithms as seen in scenario one, but all of the values are around 4 times larger than those seen in scenario one. These trends agree with the observations in literature that the UKF is better at dealing with IC error and gyroscope bias compared to other estimation algorithms [14].

4.1.3 Scenario Three Results

Figures 4.5–4.6 and Tables 4.9–4.10 give the error and standard deviation values from scenario three, the collision trajectory. The figures contain two sets of values for the attitude and gyroscope bias error: the average RMS error for the entire manoeuvre and the average RMS error during the crash only. As we are investigating how the crash scenario affects the performance of each estimator, the average RMS error during the crash is computed as the error from the instant the quadrotor contacts the wall until the point when the attitude is stabilized by the collision recovery controller, which tends to take approximately 0.6 seconds. The tables also contain standard deviation data for the RMS error of the entire run and the RMS error of only the crash. In the tables and figures, $AUKF_1$ corresponds to the high gain AUKF and $AUKF_2$ corresponds to the covariance matching AUKF. In order to improve readability, Figure 4.5 does not show the values for the average RMS

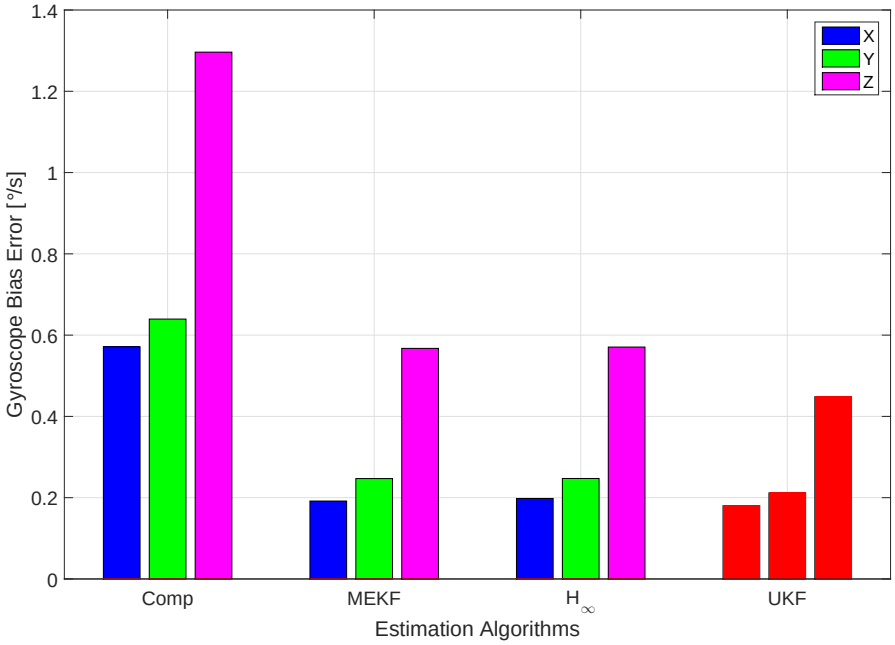

FIGURE 4.4: Average RMS gyroscope bias error for scenario 2 (high ICs, no collision) of 200 simulated flights. Red bars correspond to the lowest error

TABLE 4.7: Standard deviation of RMS angle error for entire flight of scenario 2. Bold values signify the lowest values.

Algorithm	Standard deviation [°]			
	Total	Yaw	Pitch	Roll
Complementary	38.82	33.88	4.51	5.51
MEKF	1.34	1.28	0.367	0.222
H_∞	1.19	1.13	0.324	0.199
UKF	**0.684**	**0.649**	**0.188**	**0.106**

TABLE 4.8: Standard deviation of RMS gyroscope bias error for entire flight of scenario 2. Bold values signify the lowest values.

Algorithm	Standard deviation [°/s]		
	X	Y	Z
Complementary	0.368	0.372	1.07
MEKF	0.115	0.129	0.353
H_∞	0.117	0.127	0.336
UKF	**0.108**	**0.103**	**0.217**

total angle error and instead these are shown in Figure 4.7, which also includes the total angle error from scenario one, so a direct comparison can be made.

It can be seen that the trends for the average attitude and gyroscope bias error, for the entire manoeuvre match those of scenario one. The standard deviations also follow similar overall trends to those seen in scenario one. The AH_∞ filter performs slightly better than the regular H_∞ filter and performs similarly to the MEKF. The AH_∞ filter provides a slightly lower standard deviation during the collision recovery than the MEKF or H_∞ filter, but has a slightly larger standard deviation during the overall flight. The UKF algorithms provide the best average error, with the high gain AUKF performing slightly better than the conventional UKF, and the covariance matching AUKF not showing any significant improvement over the UKF. The UKF algorithms all have similar standard deviations during the entire run and collision recovery. All algorithms have an increase in the standard deviation along the yaw and pitch components during collision recovery compared to the standard deviations for the entire flight. The increase in the standard deviation for the UKF based algorithms is less than the increase in the EKF algorithms, though the standard deviations for the EKF and H_∞ filters are lower for the total run. The standard deviation for the roll component actually decreases slightly when compared to the standard deviation in the error for the entire run. It can also be seen that the standard deviation of the gyroscope bias estimate error decreases during collision recovery. This shows that the algorithms are more consistent in estimating the roll of the vehicle during collision recovery, but are less consistent in estimating yaw and pitch, with the consistency of the UKF algorithms being the least affected by the collision. The results in Figure 4.7 show that the majority of the algorithms have smaller total angle error for scenario three than scenario one when comparing the total angle error for the entire run. It also shows that the majority of algorithms perform slightly better during the collision than during the complete manoeuvre. The only estimator which performs noticeably worse during the collision is the complementary filter.

During the collision, the complementary filter produces a worse estimate for all three of the Euler angles compared to the corresponding estimates for the entire run. It is likely that the increase in the complementary filter error can be attributed to the increase in sensor noise during the collision. As the complementary filter is a deterministic observer, its stability analysis assumes the system has no noise and it would appear that here, the high noise values cause an increase in error in its estimate.

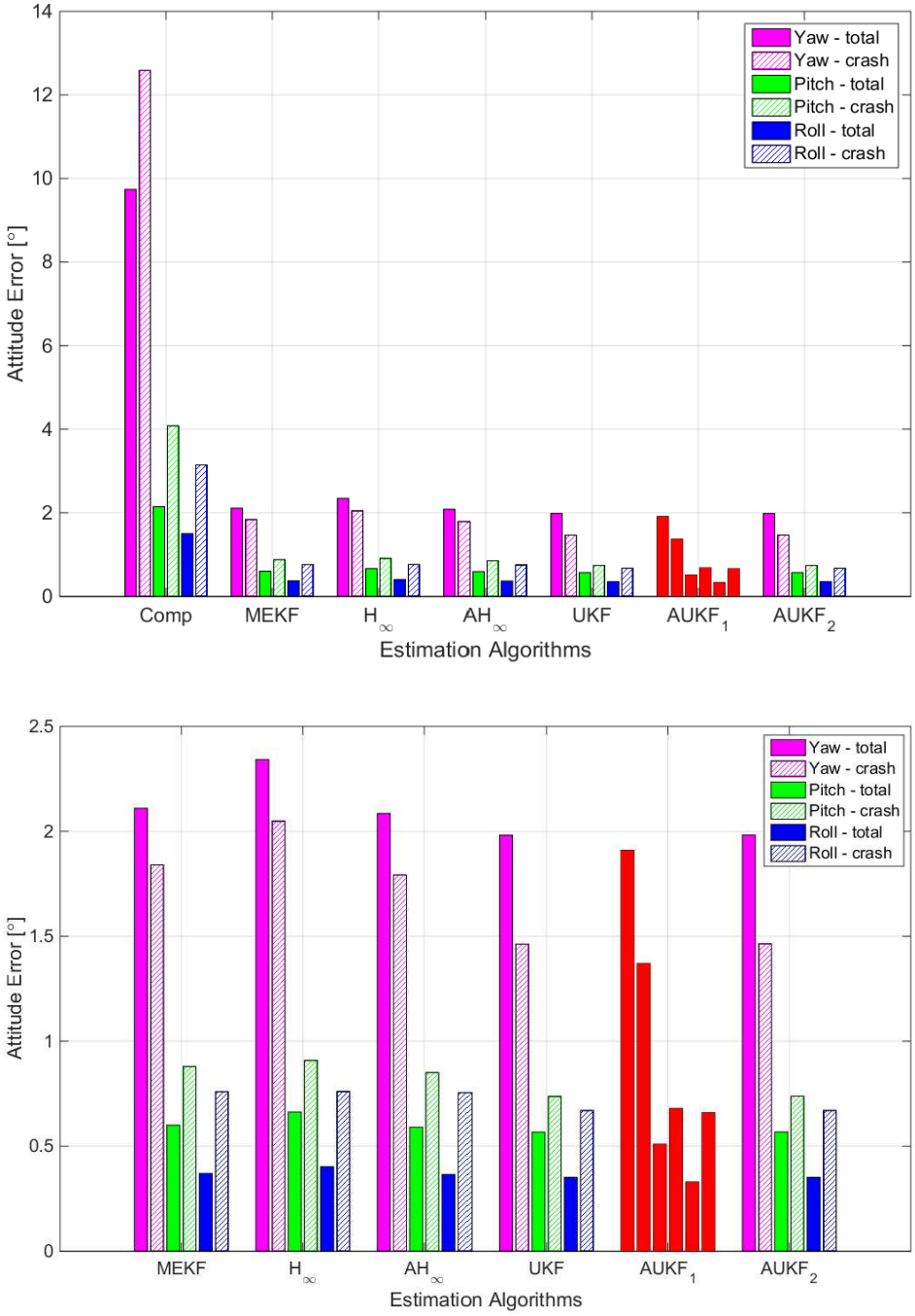

FIGURE 4.5: Average RMS Euler angle error for scenario 3 (low ICs, collision at end of run) of 200 simulated flights. Red bars correspond to the lowest error. The upper and lower graphs contain the same data but the lower figure does not include the complementary filter data for clarity

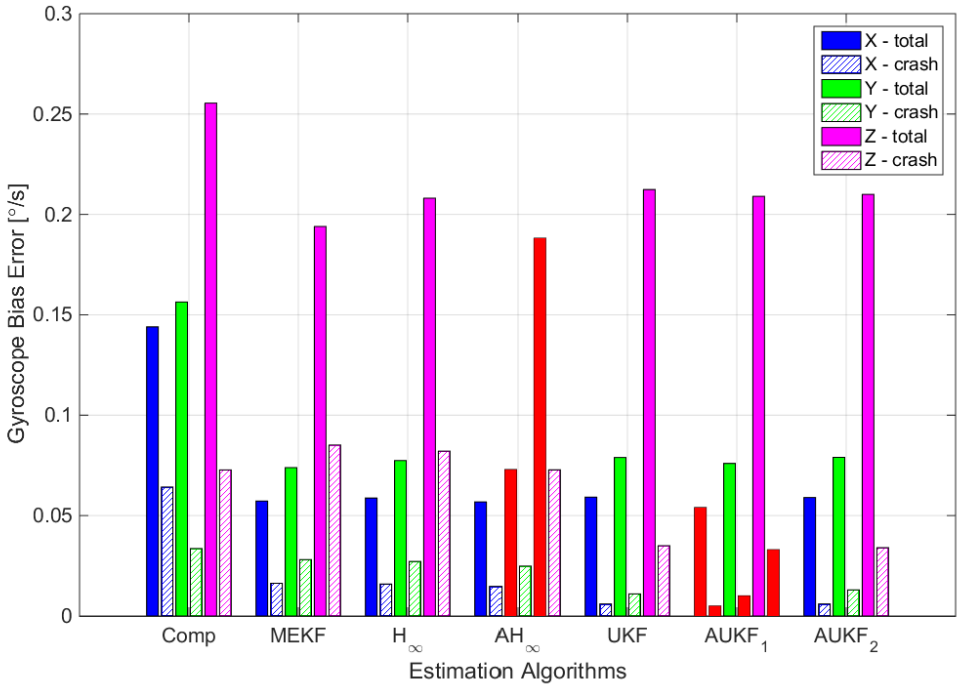

FIGURE 4.6: Average RMS gyroscope bias error for scenario 3 (low ICs, collision at end of run) of 200 simulated flights. Red bars correspond to the lowest error

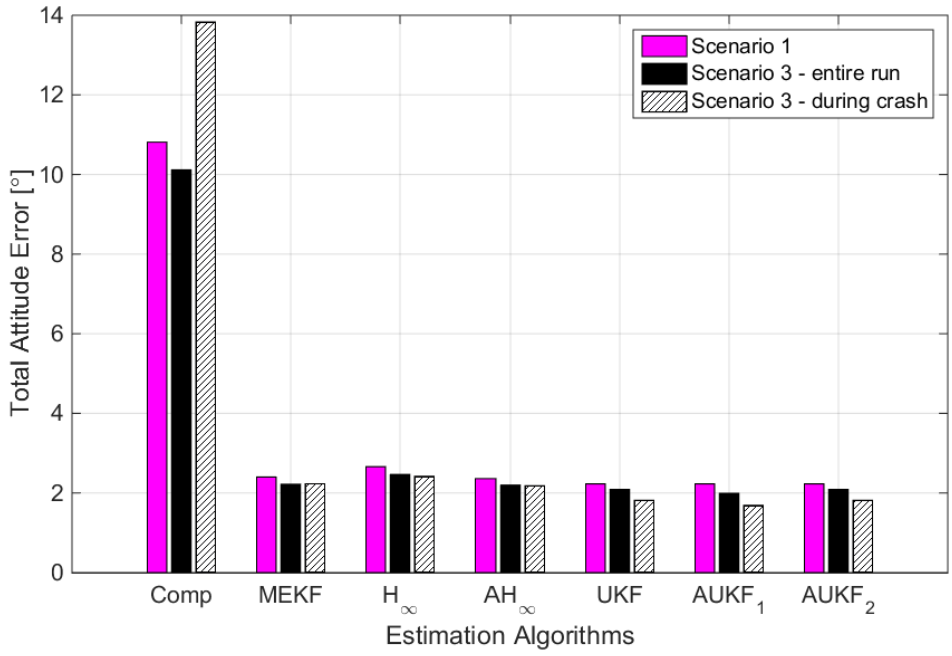

FIGURE 4.7: Comparison of average RMS total angle error for scenario 1 (low ICs, no collision) and scenario 3 (low ICs, collision at end of run)

TABLE 4.9: Standard deviation of RMS angle error for entire flight and crash of scenario 3. Bold values signify the lowest values.

Algorithm	Standard deviation [°]					
	Yaw		Pitch		Roll	
	Total	Crash	Total	Crash	Total	Crash
Complementary	8.27	10.65	1.51	1.88	1.09	0.520
MEKF	0.448	1.19	0.128	0.298	0.078	**0.058**
H_∞	**0.418**	1.26	**0.120**	0.325	**0.074**	0.060
AH_∞	0.470	1.173	0.134	0.290	0.082	**0.058**
UKF	0.555	**0.984**	0.155	**0.264**	0.101	0.100
$AUKF_1$	0.571	1.00	0.160	0.271	0.104	0.101
$AUKF_2$	0.555	0.986	0.155	**0.264**	0.101	0.100

TABLE 4.10: Standard deviation of RMS gyroscope bias error for entire flight and crash of scenario 3. Bold values signify the lowest values.

Algorithm	Standard deviation [°/s]					
	X - total	X - crash	Y - total	Y - crash	Z - total	Z - crash
Complementary	0.081	0.024	0.088	0.025	0.204	0.066
MEKF	0.023	0.006	0.026	0.013	0.063	0.045
H_∞	0.023	0.006	0.025	0.012	**0.060**	0.043
AH_∞	0.023	0.006	0.026	0.011	0.065	0.040
UKF	**0.020**	**0.003**	**0.023**	**0.007**	0.069	**0.020**
$AUKF_1$	**0.020**	**0.003**	**0.023**	**0.007**	0.069	0.021
$AUKF_2$	**0.020**	**0.003**	**0.023**	**0.007**	0.069	**0.020**

The MEKF also has a slightly less accurate estimate during the collision than during the entire run, but the difference is negligible at 0.01°. Both the H_∞ and AH_∞ filters show a slight improvement in their total attitude estimate error during the collision compared to the error during the entire run. While there is an improvement in the H_∞ filter's estimate during the collision, the H_∞ filter still provides an estimate which is worse than the MEKF, and the improvement garnered by the AH_∞ filter is not significant, only 0.02° better than the MEKF result.

There is a more significant improvement in the UKFs estimates during the collision compared to the other estimation algorithms' performance during the collision. Since the collision occurs after the quadcopter completes the square trajectory, it could be that the decrease in error during the collision is because the estimators have had time to converge. However, there is also a decrease in the error for the total manoeuvre of scenario three compared to scenario one, as seen from Figure 4.7, implying that the aggressive motion of the crash slightly improves performance. This slight improvement can also be seen in the gyroscope biases when comparing the average gyroscope bias errors for the entire run of scenario three, Figure 4.6, to that of scenario one, Figure 4.2. The results show that, for the Kalman based algorithms, the aggressive motion of the quadrotor during a collision improves the yaw and gyroscope bias estimates, while the attitude estimates along the pitch and roll are slightly degraded. While the pitch and roll estimates do get worse when a collision occurs, the increase in overall performance corroborates the earlier hypotheses that the alignment of the gravitational and magnetic vector with the down axis contributes significantly to the estimation errors in the yaw attitude component and gyroscope bias. It is hypothesized that the shift in alignment away from the z axis of the gravitational and magnetic vectors during the large attitude changes caused by a collision improves the observability of the biases and yaw angle, resulting in an overall more accurate attitude estimate. However, while there is a decrease in the overall average attitude estimate error during the collision for most algorithms, there is also an increase in the standard deviation of the attitude estimate. This implies that there is a larger spread of data during the collision, and that while on average there is a slight improvement across all flights, the attitude estimates become less consistent - some collisions see an improvement in the attitude estimate while some estimates become worse. It is possible that whether or not there is an improvement in the attitude estimate depends on the motion during the collision. Some motions resulting from collision recovery may improve the overall observability of the system more than

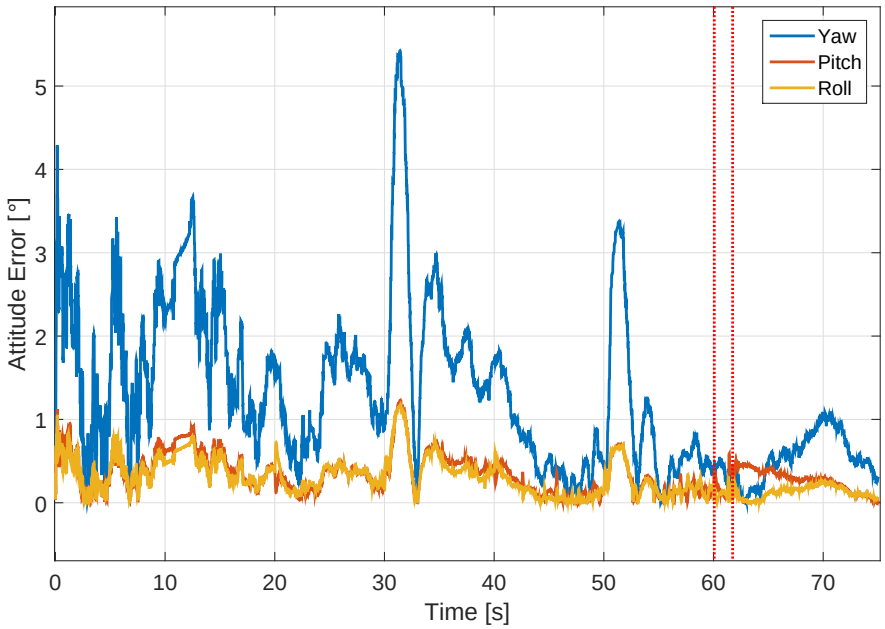

FIGURE 4.8: Absolute Euler angle error for UKF during a single simulation flight. First dashed red line marks the impact with the wall and the second marks when attitude is stabilized

others. While it is beyond the scope of this thesis, it could be worthwhile to look into whether there is a pattern that correlates to whether specific collision recovery motions, or control inputs associated with collision recovery, result in an improved attitude estimate after a collision.

Figure 4.8 shows how the Euler angle error for the UKF develops over time during a single simulation run. In this flight it can be seen that the Euler angle error increases and decreases sporadically through the flight, likely as the vehicle flies through specific manoeuvres. Over the duration of the entire run the yaw, pitch and roll errors can be seen to decrease, with the decrease in the pitch and roll error being less significant than that of the yaw. This can likely be attributed to the gyroscope bias estimate converging for all components. From the point when the impact of the collision occurs (the first dotted red line in the figure) until the point when the attitude is stabilized by the collision recovery controller (the second dotted line in the figure) a slight decrease in the attitude estimate error can be seen. Towards the end of the collision recovery manoeuvre the roll and pitch error increase. After the attitude is stabilized and the vehicle is hovering, the yaw error begins to increase, while the roll and pitch errors steadily decrease. While this does corroborate the hypothesis that the motion of the collision temporarily increases observability of the yaw, it can be

seen that the error is very noisy, and the short term trends of a single flight such as this can only be taken as meaningful due to the results of the Monte Carlo simulations. It is also interesting to note that the errors incurred due to the motion of the vehicle flying along the square trajectory can be just as large as the error at the start of the simulation due to initial attitude and bias estimate errors.

While some estimation algorithms did perform better during the collision, the adaptive algorithms did not provide as significant of an improvement as was hoped for. The fact that the H_∞ filter performed worse than the MEKF, and the AH_∞ filter provided such a small improvement implies that the collision does not cause significant enough noise problems in the sensor data to warrant the robust trade offs of the H_∞ filter. While the high gain AUKF did show a slight improvement over the UKF, it is possible that all three adaptive algorithms would show a more significant improvement over their non-adaptive counterparts if a dynamics model is used in the prediction step of the algorithms. The adaptive high gain EKF presented by Sebesta et al. [23] on which the high gain UKF is based, shows improved performance during aggressive manoeuvres using a dynamics model in the prediction step. Similarly Chee and Forbes [27] show the H_∞ filter improves performance when the dynamics model used has large errors. It is possible that the adaptation or robustness of these algorithms generates the most improvement when the estimator is impacted by unmodelled disturbances or large modelling errors. Thus, by only using sensor measurements in these algorithms, which was partially done to avoid these modelling errors, the improvement in performance expected from adaptive or robust techniques is smaller or negligible.

Another factor that could reduce the effect of the AUKFs is the thresholding of the accelerometer measurement. Since the accelerometer measurement is not used when its magnitude is over a certain threshold, the adaptive algorithms would have less of an effect as the thresholding would cause the accelerometer measurement to be ignored when there is significant noise or changes in linear acceleration, as there are during the collision. Since one main goal of the adaptive algorithms is to increase the noise covariance for the inertial sensors, the fact that the accelerometer measurement is not used during the collision could limit their effectiveness.

It is also worth noting that the adaptive algorithms required a significant amount of tuning in order to garner this small improvement in performance. Since the adaptive algorithms require more tuning parameters, a set of well-tuned parameters that yielded good results in the adaptive

algorithms was difficult to find. Furthermore, it was found that moderately small changes away from these well-tuned parameters would actually cause a decrease in performance when compared to the results of the basic UKF. This implies that small changes to the vehicle or system would require extensive retuning or, if the changes occur without the knowledge of the user, it could result in a poorer state estimate. This factor would make the use of an adaptive algorithm somewhat impractical for the given scenario. Although their improvement during simulation was small, there are many unmodelled factors during a collision that may further validate the use of the H_∞ or adaptive algorithms. As a collision is chaotic and hard to model, the comparison of adaptive and conventional algorithms using experimental data may give different results.

4.2 Discussion

The simulation results showed that the aggressive motion of a collision increased observability, improving the state estimate more than the destabilizing effect of the increase in sensor noise. It was found that the Kalman based filters were much better at estimating gyroscope bias and dealing with IC error than the complementary filter, and that the UKF performed the best in these regards. The robust trade off of the H_∞ algorithm proved detrimental to state estimation in the scenarios examined and the adaptive algorithms provided minimal improvements compared to their conventional counterparts. In addition to the results presented in Section 4.1, an investigation was carried out into various modifications to the presented algorithms, as proposed in literature, to evaluate the effect of these modifications on their performance. Two specific modifications are explored in the following sections and the discussion is ended with several remarks on the computational complexity of the filters.

4.2.1 Norm-Constrained Filters

An examination of norm-constrained versions of the MEKF and H_∞ algorithms was carried out, to see if the norm-constrained versions would yield any improvement over the conventional ones. Zanetti et al. [11] show that in some situations their CEKF gives results comparable to a UKF and that it outperforms the classic MEKF. In [11] it was found that the CEKF provided a significant improvement over the MEKF when the sensor update rate was set at 1 Hz. The norm-constrained

versions of the MEKF and H_∞ filters were implemented in the present research and compared in the given simulation scenarios of Section 4.1. It was found that while the CEKF and norm-constrained H_∞ filter did provide an improvement over their conventional counterparts, the improvement was minor, on the order of a hundredth of a degree. It is thought that the CEKF provided a significant improvement in [11] because of a very low update rate of 1 Hz, compared to the 100 Hz rate used here. At an update rate of 1 Hz the estimation algorithms would have significantly larger error quaternions and having larger error quaternions would result in larger error due to renormalization. Here, as the sensor update rate is 100 times that used by Zanetti et al., the error quaternions are much smaller, and as such, deviations from the norm constraint are less significant, resulting in only a minor performance improvement.

The CEKF derived by Zanetti et al. uses the MEKF for its structure, but the MEKF already provides an improvement over a conventional EKF by estimating the error quaternion. The MEKF is termed an indirect filter as it reduces error due to the norm constraint by estimating the error quaternion instead of the full quaternion [10]. A CEKF which estimates the full quaternion, termed a direct CEKF, was also implemented in this study to see if enforcing the norm constraint while estimating the direct attitude quaternion matched the performance increase provided by estimating the error quaternion. It was found that the MEKF still provides a significantly better attitude estimate than the direct CEKF.

4.2.2 Normalization of Attitude Vector Measurements

The effect of normalizing the accelerometer and magnetometer measurements in order to use them as attitude vector measurements was also investigated. Since the only important part of these measurements is vector orientation, any changes in magnitude could have a negative impact when they are used to update the attitude estimate. Therefore, by normalizing the measurements, the effects of magnitude changes are removed. Changes in the magnitude of the measurements could be caused by a number of factors, such as, changes in the accelerometer magnitude due to linear accelerations in the vehicle, or changes in the magnetometer measurement due to external magnetic fields. In simulation, as the method used to corrupt the magnetometer measurement will not affect its average magnitude, only changes in accelerometer measurement magnitude would have a significant effect. The design of the complementary filter requires the accelerometer and

magnetometer readings to be normalized by default, so here, only the effect on the performance of the Kalman based filters is being discussed. While it was initially thought that normalization of the measurements would improve the accuracy of the estimation algorithms, the results proved to the contrary. It was found that when the accelerometer and magnetometer measurements were normalized, the performance of the filters was much less consistent compared to versions which did not normalize the measurements. It is thought that this reduction in performance is due to the fact that the process of normalization changes the noise characteristics of the measurements. The Kalman based filters work on the assumption that the noise in the system is Gaussian, while normalization of the measurements would change not only the mean and covariance of the noise, but the probability density of the measurement noise altogether. Review of previous research which used normalized measurements in Kalman based filters shows that some effort has gone into developing methods which combine normalized measurements to compute an attitude estimate, then use the estimate directly in EKFs [49], [50]. The methods specifically discuss modifications to the covariance matrices in order to better reflect the normalized noise. Investigation of these methods could be worth while in the future as differences in magnitude between the predicted and actual measurements could have a significant effect on attitude estimation.

4.2.3 Computational Complexity

Another important consideration in the choice of a state estimation algorithm is the computational complexity of the method. Using the timing features of MATLAB, a relationship of the algorithm complexities relative to each other can be determined for the specific scenarios and algorithm implementations. For the estimation algorithm formulations used here, it was found that the MEKF and H_∞ filters took approximately twice as long to execute as the complementary filter, and the UKFs took about 13.5 times as long as the complementary filter. The SRUKF was also implemented as a variation on the UKF algorithms, as discussed in Chapter 3, in order to compare computation time and any increases in numerical accuracy. In all cases, it was found to provide a negligible decrease in attitude estimation error (on the order of a hundredth of a degree) but with a slight gain in the computational speed.

The rate at which the estimation algorithms were executed was also investigated to evaluate its affect on the overall performance of each algorithm, and if the rate had any effect on the general

trends found in simulation. The performance of the algorithms when executed at 200 Hz was found to follow the same trends as at 100 Hz, but with a small decrease in the RMS attitude error for all estimators (approximately 0.2 degrees). It is likely that the relationship between estimation accuracy and estimator update rate is non-linear, and increasing the update rate beyond 200 Hz is not likely to result in significant gains in performance. Evaluating the estimators' performance at a lower rate than 100 Hz was not considered worthwhile as this rate is easily obtainable on conventional processing hardware on current UAVs. In a situation where computing resources are limited, yet sensor data is available at a fast rate, using a less computationally expensive algorithm, such as the complementary filter, with a higher update rate might prove more effective than using a more complex algorithm at a slower rate. As is discussed in the following chapter, an implementation of the UKF algorithm onboard the Pixhawk flight controller was not possible due to overloading of the CPU. While it is not pursued in this work, this helps to show that a reordering of the relative performance of the algorithms based on the execution rate would be useful.

Chapter 5

Implementation and Experimental Validation

The simulation results of Chapter 4 give insights into how the estimation algorithms will perform when they are implemented using actual sensor data or in the control loop of a quadcopter. The main issue with validating state estimation algorithms experimentally is that the true values of the states are not innately known, so error in an algorithm's state estimate can be difficult to measure. In order to validate an estimation algorithm experimentally, a method to accurately measure a vehicle's state is needed, or methods must be devised to use another measure to compare algorithms. In line with accurately measuring a vehicle's state, the AML is equipped with a Vicon motion capture system, which uses infra-red light and cameras to track a constellation of infra-red reflective tracking balls, providing a very accurate estimate of the position and orientation of the constellation. An alternative method to measure the effectiveness of an attitude estimation algorithm during a collision is thought to be the time required to recover from a collision; if the collision recovery controller has a less accurate attitude estimate, it will likely take longer to recover from a collision. Thus, we compare the algorithms using two methods: post-processing sensor data to compare an algorithm's attitude estimate to the attitude estimate of the Vicon motion capture system (Section 5.2), and comparing the time taken to stabilize the vehicle after a collision when using an algorithm in the control loop (Section 5.3). Each of these methods to validate the algorithms has sources of error but together provide a reasonable evaluation of the algorithms' performance. The first section of this chapter, 5.1, is dedicated to the discussion of the methods used to initialize and implement the algorithms when using real sensor data.

5.1 Implementations with Real Sensor Data

Some aspects of the algorithms need to be handled differently when implemented using real sensor data compared to when they are implemented in simulation. One of the main issues is the initial state estimate. Since there is no information about the initial gyroscope bias, the initial estimate for the gyroscope bias is taken to be zero. However, an initial estimate for the quaternion can be approximated from the sensor values. Here the initial attitude estimate is computed using the method which is implemented in the PX4 flight stack, but with a slight modification to increase the accuracy of this initial estimate. A more accurate estimate of the initial attitude is required here, as this estimate is used to compute an estimate of the Earth's magnetic field in the inertial frame, a value which is not needed for the Pixhawk algorithm. An accurate value of Earth's magnetic field in the inertial frame is required since it is used by the algorithms presented in Chapter 3 to predict the magnetometer measurement in each estimation cycle. Therefore, using an inaccurate inertial magnetic field value would result in a systematic error in the algorithms.

As the magnetic field is different in different parts of the world, an accurate value of the magnetic field specifically in the flight region is required. This can be achieved in two ways: using the global position of the vehicle with a model of Earth's magnetic field, or using the attitude of the vehicle with the body frame magnetometer measurement. If the vehicle travels a significantly large distance during its flight, the global position of the vehicle, usually taken from a GPS, can be input into a model of Earth's magnetic field to compute the field at the vehicle's location. This method can have systematic errors resulting from the accuracy of the magnetic field model. The other method is to rotate the magnetometer reading from the body frame into the inertial frame to compute an inertial magnetic field estimate. This method could also be used for longer flights, but the value of Earth's magnetic field in the inertial frame would need to be recomputed after travelling a certain distance. The method is also subject to error from an inaccurate attitude estimate or noise in the magnetometer. Since there is no GPS data available, and the magnetic field is not expected to change significantly because our flights will be short in duration and range, the second method is used and an approximation of the magnetic field is computed based on magnetometer measurements. As all of the experiments take place in the AML, a variation of the first method was considered, which uses a predetermined inertial magnetic field vector based on the location of the AML. This method was ruled out as it limits the location and portability of the algorithms,

requiring any other implementations to update the magnetic field at each new flight location.

For the experiments conducted for this thesis, the initial attitude estimate is computed by taking the average of the first 20 accelerometer and magnetometer measurements in order to approximate the magnetic field and gravitational vector in the body frame. This is the main modification to the PX4 method, as the PX4 method only used one measurement from each sensor to approximate these values. Using multiple values to make the initial attitude estimate reduces the error due to sensor noise. They are computed by:

$$\hat{\boldsymbol{\mu}}_b = \frac{1}{20}\sum_{k=0}^{20} \mathbf{u}_{m|k} \tag{5.1a}$$

$$\hat{\mathbf{g}}_b = \frac{1}{20}\sum_{k=0}^{20} \mathbf{u}_{a|k} \tag{5.1b}$$

These values are then used to form the basis vectors of a coordinate system, from which the rotation matrix is computed as:

$$\bar{\mathbf{k}} = \frac{-\hat{\mathbf{g}}_b}{\|\hat{\mathbf{g}}_b\|} \tag{5.2}$$

$$\mathbf{i} = \hat{\boldsymbol{\mu}}_b - (\hat{\boldsymbol{\mu}}_b^{\mathsf{T}}\bar{\mathbf{k}})\bar{\mathbf{k}} \tag{5.3}$$

$$\bar{\mathbf{i}} = \frac{\mathbf{i}}{\|\mathbf{i}\|} \tag{5.4}$$

$$\bar{\mathbf{j}} = \bar{\mathbf{k}} \times \bar{\mathbf{i}} \tag{5.5}$$

$$\hat{\mathbf{C}}_0 = \begin{bmatrix} \bar{\mathbf{i}}^{\mathsf{T}} \\ \bar{\mathbf{j}}^{\mathsf{T}} \\ \bar{\mathbf{k}}^{\mathsf{T}} \end{bmatrix} \tag{5.6}$$

Then the initial quaternion estimate can be extracted from the value of $\hat{\mathbf{C}}_0$, following the methods in [51], and an estimate of the Earth's magnetic field in the inertial frame can be computed as

$$\hat{\boldsymbol{\mu}}_i = \hat{\mathbf{C}}_0 \hat{\boldsymbol{\mu}}_b \tag{5.7}$$

which is used in place of $\boldsymbol{\mu}$ when predicting the magnetometer measurement. Since the estimated values for the gravitational vector and magnetic field are approximated by averaging the values over

20 timesteps, the vehicle is required to remain motionless during this period. This is not difficult as this period is only about 0.15 seconds, and occurs while the vehicle is still on the ground. However, if the vehicle moves, the vectors will be skewed, resulting in inaccurate initial attitude and magnetic field estimates.

5.1.1 Magnetometer Correction and Sensor Filtering

During the characterization of the sensor noise, it was found that the magnetometer onboard Navi was significantly affected by the quadcopter's electronics. It is well known that the magnetometer measurements can be affected by the magnetic fields generated by the propeller motors, the power distribution board and the battery onboard a UAV. The only methods to circumvent interference from these components is to move the magnetometer further away from them or to use magnetically permeable alloys to reroute the magnetic fields generated by these devices. The design of Navi does not allow for significant changes to be made to the position of the electronics, so the magnetometer could not be moved far enough from the power distribution board and motors to completely avoid interference. Attempts were also made to use magnetically permeable alloys to reduce the interference caused by the electronics but again, the design of Navi did not allow for the proper placement of these alloys, and their effects were minimal. To properly shield the electronics (without blocking out Earth's magnetic field) or to move the magnetometer far enough away to discount the effects would require significant re-design of the platform, beyond the scope of this work.

Another method to counteract the magnetic effects of the power electronics is to compensate the magnetometer measurements based on the current flow out of the battery. As it was found that the power distribution board was a big contributor to the interference, this method yielded a significant increase in accuracy for the estimation algorithms. The magnetic field generated from the electronics is linearly proportional to the current flow out of the battery, and thus the magnetometer measurement can be compensated using a linear correction factor. The top graph in Figure 5.1 shows the extent to which the magnetometer reading is corrupted by the currents through the battery and power distribution board. The figure shows the magnetometer measurement as the throttle is alternated between minimum and maximum thrusts three times, while the vehicle is held stationary. Linear coefficients were found which map the current draw from the battery to the change

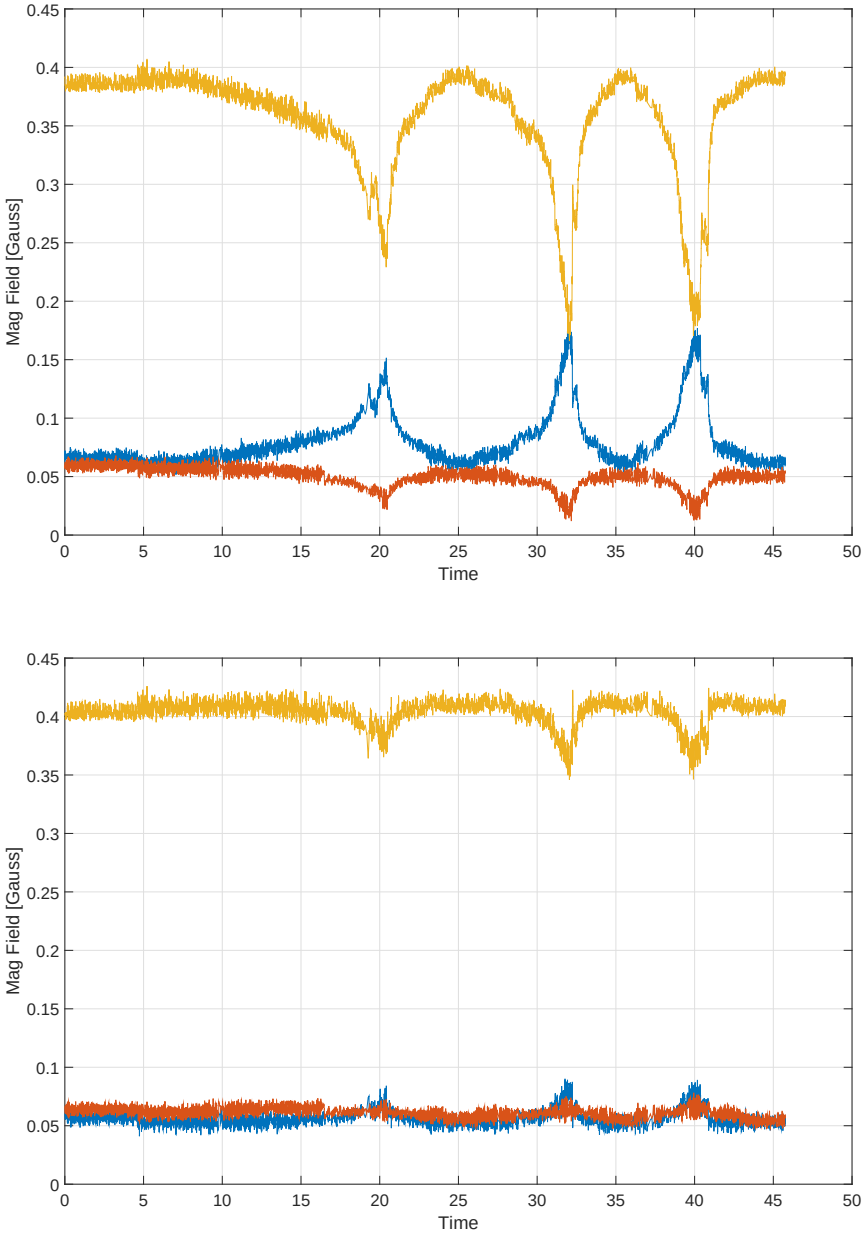

FIGURE 5.1: Uncompensated (top) and battery current compensated (bottom) magnetometer measurements when propellers alternate between max and min thrust

in magnetometer measurement by taking the average difference between the expected and measured magnetometer measurement as a ratio to the change in the battery current at that instant. Here, the battery current is available as a measurement and the expected magnetometer measurement is based on the magnetometer measurement before the motors are throttled. These coefficients are then multiplied by the measured battery current and subtracted from the magnetometer reading to give its corrected value. The lower graph in Figure 5.1 shows the corrected magnetometer measurement. While this method does correct the magnetometer measurement a significant amount, there is still some error in the magnetometer. This correction also only accounts for disturbances from the battery and power distribution board, which were found to generate the largest magnetic field changes, but disturbances from other components may still interfere with the magnetometer reading.

The PX4 flight stack also passes the accelerometer and gyroscope data through a digital lowpass filter. The code documentation explains that this is done because the signals are insufficiently filtered in the electronics. Therefore, consistent with the PX4 implementation, the accelerometer and gyroscope data were passed through the same digital lowpass filter implemented in the PX4 attitude estimator.

5.2 Post-Processing Collisions with Estimation Algorithms

A series of quadcopter collision data sets have been collected during experiments in the AML carried out over the time frame of April-May 2017 by Gareth Dicker. These experiments were done in order to validate the collision recovery controller. These data sets include all of the sensor data as well as an attitude estimate from the Vicon motion capture system. By tracking an asymmetric constellation of tracking balls, the Vicon system can provide very accurate estimates of a vehicles position and orientation. This is very useful for validating state estimation algorithms in experiments as the Vicon's measurements are accurate enough that they can be considered the "true" orientation and position of the vehicle. The issue with using the Vicon system for an attitude estimate during a collision is that the Vicon tracking balls must be visible by the Vicon cameras, and the large rotations caused by aggressive collisions sometimes prevent that. As this evaluation is focussed on the estimation algorithms' performance during a collision, loss of a Vicon attitude estimate can be detrimental. Another requirement for accurate Vicon tracking is that the positioning of the Vicon

balls must be constant relative to each other. This means that the Vicon balls must be mounted in a rigid formation, which is again an issue during collisions. During a collision, the quadcopter experiences deformation and vibrations which can result in a less accurate Vicon pose estimate. The standard deviation in the attitude estimate of the Vicon while the quadcopter is stationary and its propellers are rotating has been found to be 0.133°. Any change in the standard deviation of the attitude estimate caused by errors due to deformation or vibrations of the vehicle as a result of collision is difficult to calculate and is not computed in this work.

Another source of error with using the Vicon attitude estimate as the true vehicle state is due to a communication delay between the Vicon data and other logged data. The communication delay is due to the fact that the Vicon system computes the attitude estimate on an offboard computer, sends it over wifi to an ODROID microcomputer onboard Navi which then sends the estimate to the Pixhawk where it is logged with the sensor data. A correction is applied to synchronize the Vicon data and sensor data in time but any error in this delay correction would cause a systematic error. The delay in the communication pipeline was determined through experiments using the Vicon system. These tests involved alternating between keeping the vehicle motionless and then manually rotating it very aggressively. An estimate of the delay was then found as the difference in time between when the Vicon estimate and when the onboard sensors register motion. The average delay was found to be 0.07 seconds; thus, the Vicon data is shifted by this much when used as the true state. The use of the constant delay value introduces some error into the evaluation as the communication delay is not necessarily constant, it being dependent on the traffic in the communication pipeline.

The Vicon system outputs the orientation of the vehicle relative to an inertial frame which is defined in terms of the laboratory walls. Therefore, its attitude output must be rotated to utilize the NED inertial frame which is used for attitude estimation in our algorithms. As the rotation between NED and the Vicon reference frames depends on a definition of the NED frame, which in turn depends on accelerometer and magnetometer data, this gives rise to another possible source of error in the evaluation. This rotation is found by again using an estimate of the rotation from the quadcopter body frame to the NED inertial frame based on equations (5.1)–(5.6). As this rotation can cause systematic errors if inaccurate, and computational complexity is not an issue as these computations are done after the flight is over, more complex methods of averaging the measurements

to compute an estimate of the rotation from the body to inertial frame were examined. The main alternative method that was investigated was one in which each set of measurements is used to compute an attitude estimate for that instant, and then these attitude estimates are combined using a modified solution to Whabba's problem [52]. Whabba's problem is a classic problem of computing the most likely orientation of a body based on a series of vector attitude measurements. However, this was found to give a less accurate result than simply averaging the measurements and using the averaged values as the estimates. Thus, the simpler and more accurate method was used.

The Vicon attitude estimate, $\mathbf{q}_{V_iV_b}$, which relates the Vicon inertial frame, $\boldsymbol{\mathcal{F}}_{V_i}$, to the Vicon body frame, $\boldsymbol{\mathcal{F}}_{V_b}$, has the z-axis pointing upwards in both body and inertial frames, so it first must be rotated so that the body frame has the z-axis pointing down. This is achieved with the following quaternion multiplication:

$$\mathbf{q}_{V_iN_b} = \mathbf{q}_{V_iV_b} \otimes \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{5.8}$$

where $_{N_b}$ corresponds to Navi's body frame, $\boldsymbol{\mathcal{F}}_b$. Then an estimate for the attitude at time step $\hat{\mathbf{q}}_k$ is computed using equations (5.1)–(5.6). Since we are post-processing the data, the initial attitude estimate can be improved further by using all the measurements for which the vehicle is not moving. For initializing the estimation algorithms, only 20 timesteps are used, since it is a relatively short amount of time and it is reasonable to assume that the quadcopter will not be moving during this initialization period. However, sometimes the quadcopter remains motionless much longer at the start of a flight; this can be observed in post processing and more measurements can be averaged in computing the attitude estimate used to rotate the Vicon inertial frame. The rotation from the NED frame to Navi's body frame, can then be used to estimate the quaternion from the Vicon inertial frame to the NED inertial frame, as

$$\hat{\mathbf{q}}_{V_iN_i} = \mathbf{q}_{V_iN_b} \otimes \hat{\mathbf{q}}_k^{-1} \tag{5.9}$$

The rotation from the Vicon inertial frame to the NED inertial frame can then be used to transform the Vicon attitude estimate at time step k to use the NED inertial frame as

$$\hat{\mathbf{q}}_{N_i N_b|k} = \hat{\mathbf{q}}_{V_i N_i}^{-1} \otimes \mathbf{q}_{V_i N_b|k} \tag{5.10}$$

The values of $\hat{\mathbf{q}}_{N_i N_b|k}$ then give an accurate estimate of the true quaternion relative to the NED frame and these can be used to evaluate the accuracy of the estimation algorithms.

It can be seen from the methodology above, that the processes involved in using Vicon as the true attitude of the vehicle has several potential sources of error. Therefore, the estimation algorithms also need to be validated through actual collision recovery experiments which use the algorithms in the control loop, as will be done in Section 5.3. Nevertheless although there may be some systematic error when using the method of post-processing sensor data, it is still useful as it is closer to the actual implementation than just simulating the collisions.

5.2.1 Experimental Set Up and Data Collection

The data set used for post processing the sensor data consists of 72 collisions, of which only 49 have little or no loss of the Vicon attitude estimate during the collisions. The data sets which lose Vicon data during the collision tend to be collisions which are more aggressive, usually when the vehicle experiences high yaw rates after impact. While these data sets are not useful for comparing estimates during the collision recovery stage, they can still be useful to see how accurate the attitude estimate is before and after more aggressive collisions.

A single data set contains data from the moment the quadcopter is armed, prior to take off, until it is disarmed, when it lands. Therefore, each data set contains the take off, approach, collision, recovery and landing of the vehicle. For the majority of a data set, the quadcopter is flown manually by a pilot. The pilot has full control of the quadcopter, except during collision recovery which is done autonomously, handing control back to the pilot after the attitude and height have been stabilized. During a flight, the pilot attempts to keep the yaw of the vehicle such that the quadcopter impacts the wall with two propeller bumpers parallel to the wall. This is done to reduce the yaw rotation of the vehicle after an impact. When the quadcopter impacts the wall with only a single propeller bumper, the vehicle can experience large moments and yaw rates. While the collision recovery

controller can stabilize the quadcopter when there are large yaw rates, it usually takes longer to stabilize and thus the quadcopter can travel far from the wall. This is avoided as the AML does not have enough space in which to safely allow the quadcopter to stabilize, and the motors have to be shut off using a safety override, resulting in an unsuccessful collision recovery.

The pitch and roll angles of the vehicle are limited by the attitude controller so that they do not surpass angles above which the recovery controller cannot stabilize the vehicle. This limit is also used to ensure that the impact angle across a series of collisions is varied uniformly. For this data set, the max pitch and roll angles were varied between 15-18 degrees, such that an even number of collisions were performed with a max angle of 15, 16, 17 and 18 degrees. For each data set the pilot applies the maximum possible pitch command, ensuring that these maximum impact angles are obtained. The pilot also starts the approach trajectory at different distances from the wall in order to vary the impact velocity. From examining the Vicon position data, the collisions vary in impact velocity from 1-4 m/s where the pilot tried to have a similar set of impact velocities for each maximum angle value. This variety of angles of attack and impact velocities allows for a Monte Carlo style analysis of the estimation algorithms.

In post-processing the data sets, the estimation algorithms were retuned to find the optimal gains and initial conditions to get the best results. The algorithms were tuned by varying each tunable variable through a set of values, testing the estimation algorithms performance using a small randomly selected subset of the data (18 data sets), and then comparing the results for different values of the tuning parameters. The values for the tuning parameters which gave the best results are given in Tables 5.1 and 5.2. For parameters not included in these two tables, the values used in simulation (Tables 4.3 and 4.4) also gave the best results when post-processing the experimental data sets. The significant changes in some of the estimator parameters are discussed in the following section.

5.2.2 Post-Processing Results

A comparison of the results of the estimation algorithms is given in Figures 5.2–5.3 and Tables 5.3–5.4. Figure 5.2 and Table 5.3 are similar to the earlier figures and tables and compares the average and standard deviation of the RMS attitude estimate error for the total flight, and just during the collision. Only the 49 data sets which have a Vicon attitude estimate available during

TABLE 5.1: ICs for all scenarios

CEKF and H_∞ filters	
$\mathbf{Q}$ =diag$[\tilde{\sigma}_{g0}\mathbf{1}_{1\times4}, \tilde{\sigma}_{b_g}\mathbf{1}_{1\times3}]$	$\mathbf{R}$ =diag$[500\tilde{\sigma}_{a0}\mathbf{1}_{1\times4}, 100\sigma_m\mathbf{1}_{1\times3}]$
UKFs	
$\mathbf{Q}$ =diag$[\tilde{\sigma}_{g0}\mathbf{1}_{1\times3}, \tilde{\sigma}_{b_g}\mathbf{1}_{1\times3}]$	$\mathbf{R}$ =diag$[500\tilde{\sigma}_{a0}\mathbf{1}_{1\times3}, 100\sigma_m\mathbf{1}_{1\times3}]$
All filters	
$\tilde{\sigma}_{b_a} = 0.05[m/s^2]$	$\tilde{\sigma}_{b_m} = 0.005[G]$
$a_{bnd} = 5$	

TABLE 5.2: Estimator specific parameters

PX4 Filter	$k_P = 1$	$k_a = 0.25$
	$k_m = 0.05$	$k_b = 0.15$
Complementary Filter	$k_P = 1$	$k_a = 0.25$
	$k_m = 0.05$	$k_b = 0.15$
H_∞ filter	$\frac{1}{\xi} = 0.1$	
High Gain AUKF	$\beta = 0.6$	

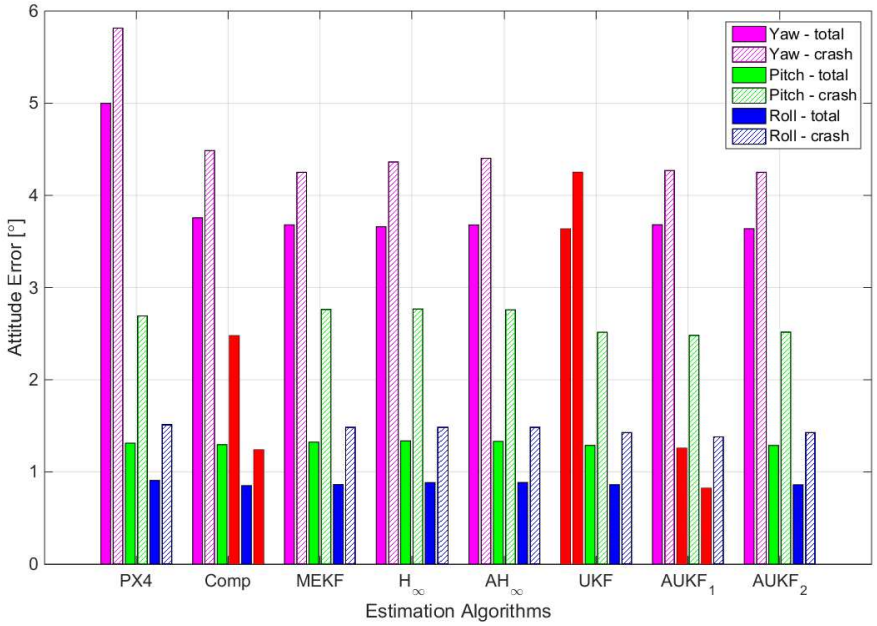

FIGURE 5.2: Average RMS Euler angles when post processing sensor data. Contains the average RMS error for the whole run, and during the crash only; based on data sets which have Vicon data during the collision recovery stages (49 flights). Red bars correspond to the lowest error

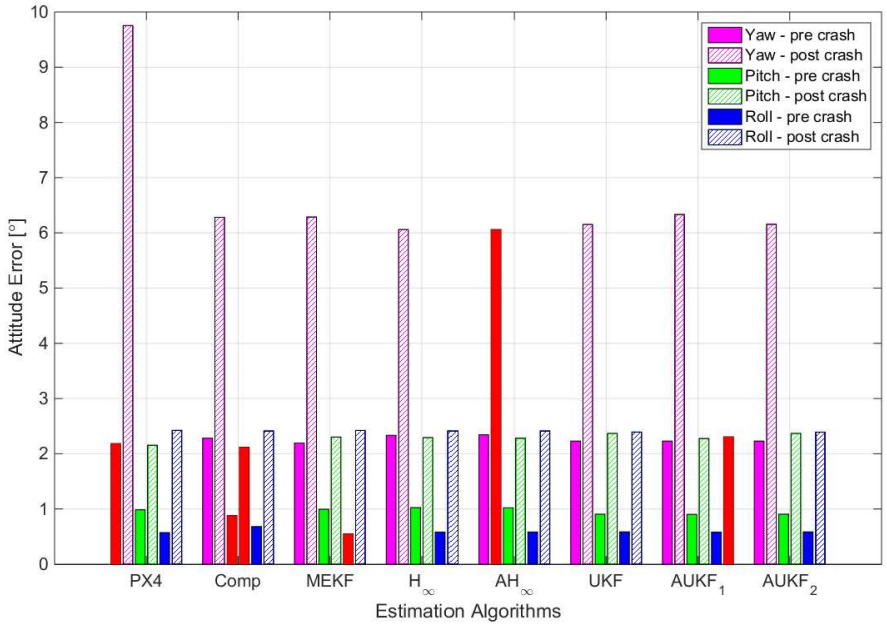

FIGURE 5.3: Average RMS Euler angles when post processing sensor data. Contains the average RMS error for before and after the collision. Shows error for all data sets (72 flights). Red bars correspond to the lowest error

TABLE 5.3: Standard deviation of experimental RMS angle error for 49 flights which have Vicon data during recovery stage. Bold values signify the lowest values.

Algorithm	Standard deviation [°]					
	Yaw		Pitch		Roll	
	Total	Crash	Total	Crash	Total	Crash
PX4	2.47	3.08	0.552	1.87	0.390	1.12
Complementary	2.50	2.98	0.492	**1.53**	0.354	**0.885**
MEKF	**2.38**	**2.74**	0.502	1.99	0.358	1.15
H_∞	2.60	3.03	0.493	1.98	0.359	1.15
AH_∞	2.59	3.09	0.489	1.98	0.360	1.15
UKF	2.48	2.86	0.521	1.84	0.363	1.03
$AUKF_1$	2.41	2.76	**0.481**	1.82	**0.338**	1.01
$AUKF_2$	2.48	2.86	0.521	1.849	0.362	1.03

TABLE 5.4: Standard deviation of experimental pre and post collision RMS angle error for 72 flights which do not have Vicon data during recovery stage. Bold values signify the lowest values.

Algorithm	Standard deviation [°]					
	Yaw		Pitch		Roll	
	Pre-Crash	Post-Crash	Pre-Crash	Post-Crash	Pre-Crash	Post-Crash
PX4	**1.59**	5.25	0.750	1.99	0.345	1.97
Complementary	1.82	3.88	0.725	**1.84**	0.320	**1.95**
MEKF	1.76	**3.87**	0.721	1.91	**0.305**	2.05
H_∞	1.89	4.01	0.726	1.90	0.329	2.04
AH_∞	1.91	3.96	0.725	1.90	0.331	2.04
UKF	1.81	3.94	0.702	1.98	0.340	2.11
$AUKF_1$	1.81	3.91	**0.700**	1.89	0.339	2.08
$AUKF_2$	1.81	3.94	0.702	1.98	0.340	2.11

the collision were used in computing the values in Figure 5.2 and Table 5.3. Figure 5.3 shows the attitude estimate error up until slightly before the collision, and the attitude estimate error from slightly after completing collision recovery until the quadcopter lands. All 72 data sets are used in computing the values shown in Figure 5.3. Table 5.4 presents the standard deviation of the data used in Figure 5.3. The figures and experimental data analysis now also include the results of the PX4 algorithm. It can be seen from the figures that the trends are somewhat different than those found in simulation, with the main difference being that the complementary filter performs significantly better. Here, the complementary filter performed similarly to all the Kalman based filters and the PX4 filter performs similarly to the other algorithms in estimating pitch and roll, but has a worse yaw estimate, by an average amount of 1.32° over the entire flight test, compared to all of the other algorithms. From Figure 5.3, it can be seen that the worse overall yaw estimate for the PX4 algorithm is due to the collision, as its error is similar to that of the other algorithms before the crash but worse by an average of 3.56° compared to all other algorithms after the collision. Table 5.3 shows that all the algorithms have similar standard deviations, with a larger standard deviation during the collision than during the rest of the run. Another difference from the simulation trends is that the state estimate during and after the collision is worse than the estimate before the collision for all algorithms. The results in Figure 5.3 also show that after collision recovery is complete, the

pitch and roll estimate errors are more similar, with the average difference between pitch and roll errors for all algorithms before and after the collision being 0.36° and -0.12°, respectively. Also, as seen in Table 5.4, all algorithms see a large increase in the standard deviation of their estimate when comparing the standard deviation before and after the collision. Table 5.4 also shows that the PX4 filter has the lowest yaw standard deviation before the collision, but has the largest standard deviation after the collision. Besides the standard deviation of the PX4 algorithm along the yaw direction, all algorithms have similar standard deviation values.

Some of the trends from simulation are still observed here as all of the filters produce a poor yaw estimate, with the pitch estimate being slightly worse than the roll estimate (except after a collision when roll error is higher). The trends between the Kalman based filters are also similar to those found in simulation. The H_∞ algorithms perform slightly worse than the MEKF, with an average error increase of 0.08° across all Euler angles for the entire flight. The adaptive algorithms again do not show significantly different performance from their conventional counterparts. There is little difference between the EKF and UKF based algorithms, and the complementary filter actually provides the best roll and pitch estimates during the collision, with average total flight errors of 2.47° and 1.24°, respectively. However the differences between the Kalman based algorithms and the complementary filter are small, and given that the standard deviation of the Vicon system is similar in magnitude to the differences in the averages and standard deviations of the attitude errors between the algorithms, it is not possible to identify the algorithm with superior performance.

Figure 5.4 shows how the attitude estimate error for the UKF develops over a single flight. The data is a flight from the set of 49 flights used in Figure 5.2 and Table 5.4. It can be seen that the error in yaw increases and then decreases both before and after the collision period. The yaw error also increases prior to the vehicle take off, but is fairly accurate at initialization, and becomes more accurate after take off. The error in the yaw of the vehicle is likely due to magnetometer disturbances. Figure 5.5 shows the magnetometer measurements for the same flight rotated into the inertial frame using Vicon attitude measurements. In theory, the magnetometer measurements in the inertial frame should be constant as the Earth's magnetic field is theoretically constant in the inertial frame. It can be seen that when the magnetometer reading is rotated into the inertial frame the y component (which directly affects the yaw estimate of the vehicle) varies in a pattern somewhat similar to the yaw error seen in Figure 5.4. It should be noted that Figure 5.4 shows the

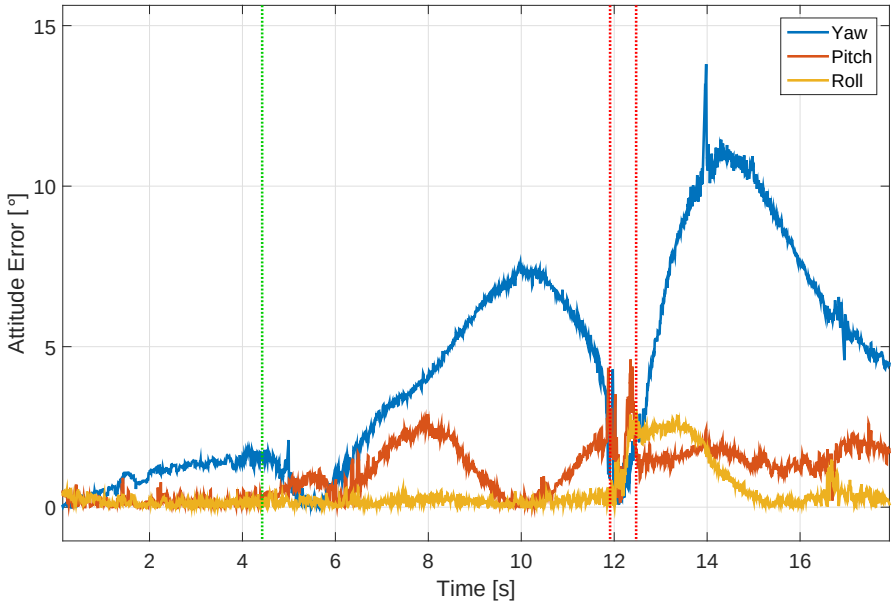

FIGURE 5.4: Absolute Euler angle error for UKF during a single experimental flight. Dashed green line corresponds to take off. First dashed red line marks the impact with the wall and the second marks when attitude is stabilized

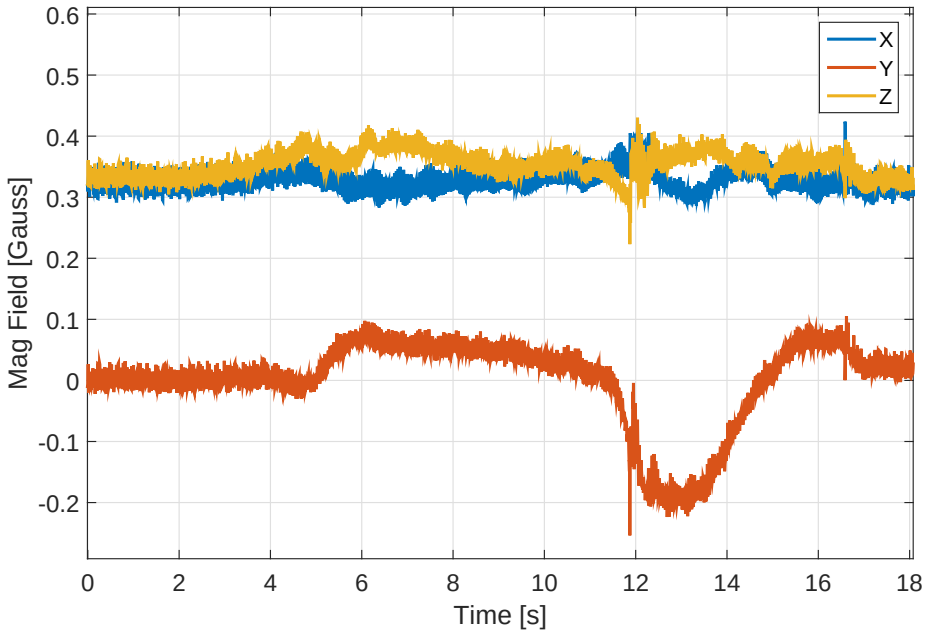

FIGURE 5.5: Magnetometer reading, rotated into the inertial frame using Vicon attitude measurements

absolute Euler angle error, and that if the error after the collision was shown to be negative, its general shape would match that of the y component of the inertial magnetometer measurement as it increases then decreases from 4-12 seconds, crosses 0 at around 12 seconds, and then increases and decreases (in magnitude) again. Some error in the attitude measurement can also be attributed to the state estimation algorithm developing error in the gyroscope bias estimate as it adapts the state estimate to try to account for the disturbances in the magnetometer measurement. An inaccurate magnetometer measurement, and matching error in the yaw of the vehicle occurs in the majority of flight data. Besides the large drifting yaw error, it can be seen that immediately when the impact of the collision occurs there is a spike in the pitch and yaw error, but then both errors decrease. After the impact, the error in roll, pitch and yaw increase until the attitude is stabilized, at which point the yaw error continues to increase, while the roll and pitch error plateau and then decrease. The spike in the estimation error immediately when a crash occurs is common to all flights, however the trend in data between the impact and when the attitude is stabilized varies, likely dependant on the collision characteristics, and magnetometer disturbances. In most flights, after the attitude is stabilized, the roll and pitch error tend to stabilize as well. This analysis of the error over time shows that even with the magnetometer correction used, there are still significant disturbances experienced by the magnetometer.

There are some notable changes in performance compared to the simulation, besides magnetometer issues already discussed, the causes are thought to be due to the following reasons. One main factor is that the sensors were calibrated before each set of flights, using a standard sensor calibration toolbox available with the PX4 software. The sensor calibration compensates the IMU sensor measurements to remove biases and misalignment errors. In simulation, it was found that when the initial gyroscope bias was set to be very small or zero, the MEKF, UKF, and complementary filter performed very similarly (see Appendix A), which is what appears to be the case with the experimental data.

Another likely cause of these discrepancies is due to large changes in the accelerometer and magnetometer measurements' magnitudes caused by vehicle accelerations and magnetic disturbances, respectively. It should be noted that large changes had to be made to the Kalman based filters noise covariances and accelerometer bounds (Table 5.1) compared to values used in simulation (Table 4.3)

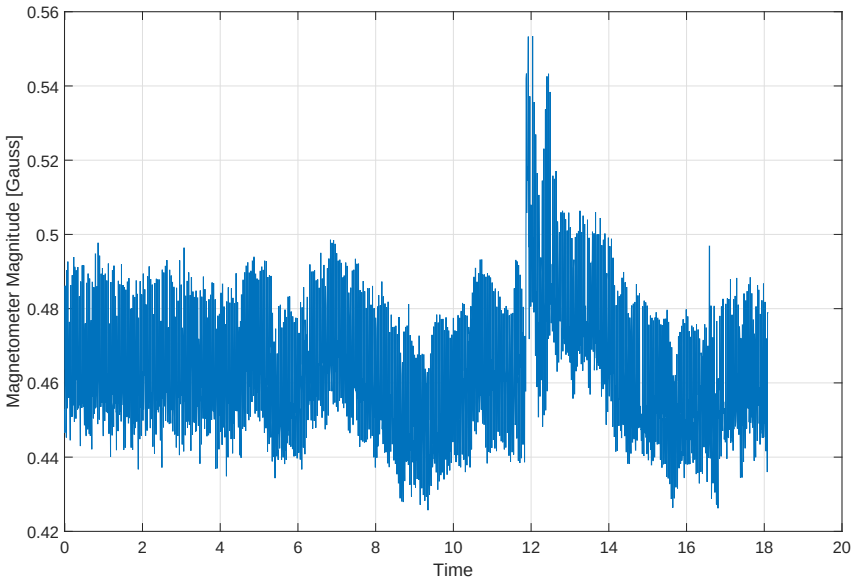

FIGURE 5.6: Magnitude of magnetometer measurement for a single flight

in order to get the Kalman based filters to provide performance comparable to the non-linear observers. The PX4 and complementary filters both normalize the accelerometer and magnetometer measurements for use, whereas the Kalman based algorithms do not. Figure 5.6 shows how the magnitude of the magnetometer reading varies over a single data set.

Since the Kalman based algorithms do not normalize the measurements, then any consistent deviation from the base magnetometer reading (as is seen after 8 seconds in Figure 5.6) would result in an error in the correction even if the predicted and actual measurement vectors are collinear. The same would occur to the accelerometer reading when the vehicle is undergoing any linear accelerations. Increasing the bound for which the accelerometer is used, as well as increasing the noise covariance of the accelerometer reading by a factor of 500, and the noise covariance of the magnetometer by a factor of 100, led to a significant improvement in performance for the Kalman based algorithms. This may show that since the magnetometer measurement is less accurate, using the accelerometer measurement more frequently, even though it is also less accurate due to linear accelerations, is better than only using it occasionally with a more accurate measurement. The improved performance when the magnetometer measurement covariance is increased is likely due to the significant error in the magnetometer caused by the vehicle's electronics. Similarly, increasing the accelerometer noise covariance likely improves performance when using an increased

accelerometer bound as the accelerometer will be distorted by linear accelerations.

These results point towards investigating algorithms which normalize the measurements, or methods which properly account for the changes in the measurement noise probability density function when the measurements are normalized. In this case it could also be worth investigating an adaptive algorithm which always uses the accelerometer reading, but which increases the accelerometer noise covariance as the magnitude of the accelerometer measurement increases.

5.3 Estimator in the Loop Results

The ultimate test of the performance of the estimation algorithms is how they affect the control system performance in a closed-loop state feedback controller. Therefore, some algorithms were chosen to be implemented in the PX4 flight stack and compared in flight. The time it takes the system to stabilize after a collision is used as a metric to compare estimation algorithms. The recovery controller operates in two stages: the first stage of the recovery controller stabilizes the attitude of the vehicle and then the second stage stabilizes the height of the vehicle. The collision recovery controller enters the first stage when the quadcopter impacts the wall, triggered when the accelerometer reading is over some threshold. The recovery controller switches between stage one (attitude stabilization) and stage two (height stabilization) when the roll, pitch, roll rate, and pitch rate of the vehicle are within specified bounds and exits stage two when the vertical velocity of the quadcopter is within a certain bound. It is thought that the length of time spent in stage one of the recovery controller is a good measure of the accuracy of the attitude estimate. This is because if the estimate is more accurate the vehicle should be able to stabilize more quickly and consistently, whereas an inaccurate estimate should result in a longer and more erratic stabilization.

Given the large number of estimation algorithms that were investigated in this thesis, it was not possible to implement and perform experiments with all the methods. Hence, only a small subset of the algorithms were selected for implementation in the PX4 flight stack. Based on both the simulation and experimental data it was decided to attempt to implement the UKF and High Gain AUKF because the UKF based algorithms showed a benefit when faced with sensor biases, whereas the rest of the simulation and post processing results did not show any distinct improvements between algorithms. With this in mind, an attempt was made to implement the UKF but it was found that the algorithm overloaded the flight controller's CPU. This likely occurred because the

algorithm was implemented such that it used every set of IMU measurements that were available, which is at a rate slightly higher than 200 Hz. It is possible that the UKF based algorithms would not overload the CPU if they were run at a slower rate, using an average of IMU measurements for a series of time steps instead of using every one. As discussed in Section 4.2.3, the evaluation in simulation shows that executing the algorithms at 200 Hz provided an improvement over 100 Hz, however, a more detailed investigation would be needed to determine the significance of the execution rates. Since implementation of UKF based algorithms was not viable at the update rate of the IMU, and a definitive improvement was seen in simulation between the 200 Hz results and 100 Hz results, it was decided to implement the MEKF and AH_∞ filters running at 200 Hz. The MEKF based algorithms were chosen as they also showed improved performance in simulation when dealing with initial gyroscope biases over the complementary filter. The AH_∞ filter was chosen to implement over the H_∞ filter as it showed some slight improvements in performance in simulation as well.

The implementation of the AH_∞ filter was slightly different from the post-processed and simulation versions in that the AH_∞ bound needed to be raised in order for the closed-loop system to be stable. It was found that the value which worked best in the post-processing comparison and in simulation, $\left(\frac{1}{\xi}\right)_{max} = 0.2$, resulted in the vehicle oscillating aggressively after exiting stage one of the collision recovery process. While using the original bound, $\left(\frac{1}{\xi}\right)_{max} = 0.2$, the pilot commented that the vehicle sometimes seemed less stable than usual during normal flight and that after a collision with the wall, the vehicle sometimes did not recover properly. Upon examining the data post-flight, it was found that after a collision with the wall the recovery controller exited stage one regularly, but the vehicle would continue to oscillate afterwards. This indicated that there was significant error in the state estimate, as the vehicle would think it had stabilized when it had not. It was deemed too dangerous to perform the collisions using the original AH_∞ bound, due to the unpredictability of the recovery, and therefore the AH_∞ bound was raised by setting $\left(\frac{1}{\xi}\right)_{max} = 0.01$. This required that the step size at which the filter adapts, e, also be changed in order to have a smooth transition, thus it was set as $e = 0.001$. The difference in results when using the AH_∞ filter's attitude estimate in the control loop shows how although an algorithm's estimate appears to converge when using post-processed data, the instantaneous error in the estimate can result in the instability of the closed-loop system.

TABLE 5.5: Average and standard deviation of the time spent in each recovery stage during collision recovery when algorithms are used in the control loop, averaged over 53 flights. Bold values signify the lowest values

Estimation algorithm	Average time in stage one [sec]	Average time in stage two [sec]	Average total time for collision recovery [sec]
PX4	0.607 ± 0.224	0.424 ± 0.322	1.03 ± 0.35
MEKF	0.522 ± 0.085	**0.301 ± 0.172**	**0.823 ± 0.175**
AHINF	**0.511 ± 0.076**	0.421 ± 0.322	0.932 ± 0.349

The MEKF and AH_∞ filters were successfully implemented in the PX4 flight stack and 53 data sets of successful collision recoveries were gathered for each algorithm. The results for the time spent in each recovery stage for the sets of successful collision recoveries are given in Table 5.5. The table gives the average time spent in each stage, as well as an error term which corresponds to the standard deviation of the average times. The results are given separately for stage one and stage two, and a third value of the total time spent in the collision recovery mode is included as well. It can be seen that, compared to the PX4 estimator, the MEKF and AH_∞ filters improve both the average time spent in stage one by about 15%, and have a lower standard deviation in their results. The lower mean and significantly lower standard deviation for the time spent in stage one indicates that it generally took longer to recover with the PX4 state estimate feedback, and the EKF and AH_∞ filters gave a much more consistent collision recovery. In a scenario where the safety of the vehicle and the surrounding environment might be in danger because of the collision, having a consistent collision recovery is very valuable.

Evaluating the implications of the time spent in stage 2 of the recovery process is a little more speculative, and more data is needed to be certain of any hypotheses. The height estimate which is used in the control loop is provided by a separate estimation algorithm in the PX4 flight stack. While this estimate is computed separately from the attitude estimate, it still uses the attitude estimate in order to rotate the accelerometer reading into the inertial frame, and thus is somewhat dependant on the accuracy of the attitude estimate. It can be seen from Table 5.5, that when using the MEKF in the control loop, the vehicle spent less time in stage two. The average time spent in stage two also had a lower standard deviation, meaning the controller was more consistent in stabilizing the vehicle's vertical velocity. The PX4 algorithm and AH_∞ filter both spent similar amounts of time in stage two, with the same standard deviation. As was seen when the AH_∞ bound

was set to be too low, the collision recovery controller was sometimes able to exit stage one even though the attitude was not fully stabilized. This means that an inaccurate attitude estimate could possibly lead to the controller spending more time in stage two trying to stabilize the height, but this is difficult to corroborate. The issues encountered when the AH_∞ bound was set too low could explain why the AH_∞ filter performs very similarly to the MEKF for stage one of the recovery process but worse for stage two. While the bound of the AH_∞ filter was changed, it may still be too low, and cause errors in the attitude estimate, resulting in a longer time spent in stage two of the recovery. The PX4 algorithm showed poor performance in estimating yaw during a collision when post-processing real sensor data so an inaccurate yaw estimate could be a factor in why the PX4 algorithm spent longer in stage two.

However, we do not want to spend too much effort interpreting the results of stage two as there were also anomalies noticed during some flights regarding height control of the vehicle that bring up issues with this speculation. It was found that after some collisions, the quadcopter would not respond properly to height commands from the RC controller used to fly the vehicle. It was found that the vehicle would either slowly rise or descend. Further investigation needs to be done to definitively attribute the different times spent in stage two of the recovery process to a specific cause.

The time spent in stage one of the collision recovery confirm post-processing results showing that the MEKF and AH_∞ filters can provide a better attitude estimate during collisions than the PX4 filter. Putting any speculations on the stage two results aside, the stage one results also confirm our findings in simulation that the AH_∞ filter provides a slight improvement in attitude estimation during a collision. However, as collisions are highly variable, further Monte Carlo style experiments could be carried to further support these results, as well as to test the hypotheses regarding the time spent in stage two.

Chapter 6

Conclusions

6.1 Summary of Work

This research has helped to advance the realization of a complete collision recovery control system to aide a quadcopter after a collision with a wall. An investigation into the effects of collisions on sensor data and on ways to mitigate these effects through the use of robust and adaptive estimation algorithms was successfully completed. It was shown that the inertial IMU sensors, specifically gyroscopes and accelerometers, are significantly affected by vibrations after a quadcopter impacts a wall. Based on these observations, a model was derived in order to imitate the effect of a collision on inertial sensors for the purpose of validating the state estimation algorithms through simulation.

Novel adaptive attitude estimation algorithms were designed to mitigate the effects of the post-collision vibrations. A survey of attitude estimation algorithms was completed to compare the capabilities of the novel algorithms to conventional ones. More specifically a comparison in performance was completed for a complementary filter, MEKF, H_∞ filter, novel adaptive H_∞ filter, UKF and two novel adaptive UKFs. The comparison of these algorithms was conducted through simulations as well as experiments.

Simulations showed that when dealing with large initial estimate errors, the complementary filter performed significantly worse than the Kalman based filters and that the UKF provided the best performance. The adaptive algorithms provided some minor improvements in attitude estimation during a collision, and their use for the given scenarios was difficult to endorse given the amount of tuning that was required to obtain these small improvements. It was found that the H_∞ filter showed a decrease in performance compared to the conventional MEKF, as the detrimental effects brought about by the collision were not significant enough to warrant its robust design. Simulations

also showed that solely using an IMU for attitude estimation results in a poor yaw estimate due to observability issues stemming from the alignment of the gravitational vector with the yaw axis of the vehicle. It was found that the aggressive motion following a collision aided in observability of a quadcopter, causing an overall improvement in attitude estimation during a simulated collision.

Before any algorithms were implemented in the control loop of the quadcopter they were evaluated by post processing sensor data. The attitude estimates of the algorithms were compared to the very accurate orientation estimates provided by a Vicon motion capture system. The data showed that the algorithms detailed in this thesis improved performance when compared to the estimator included in the PX4 software used onboard the quadcopter. The trends were different than those found in simulation, in that the complementary filter matched the performance of the Kalman based estimation algorithms, and the attitude estimates were less accurate during the collision. This was attributed to large error in the magnetometer measurements due to interference from the electronics as well as pre-flight sensor calibration causing the experimental data to have lower sensor biases than were used in simulation. No algorithm performed definitively better than the others when comparing them through post-processing because the standard deviation in the Vicon measurements was of the same order of magnitude as the differences in performance between the estimation algorithms.

In implementing the algorithms onboard the quadcopter in a closed-loop control system it was found that the UKF algorithms were too computationally intensive, and overloaded the CPU. The MEKF and AH_∞ filter were successfully implemented onboard and compared to the PX4 estimation algorithm. The experiments showed that the time spent in the attitude recovery stage by the MEKF and AH_∞ filter was less than the PX4 estimator and was much more consistent. The MEKF also spent much less time in the second stage of the recovery process compared to the other two algorithms, but the reasons why are speculative. Overall this thesis clarified the attitude estimation issues experienced by the quadcopter during collision recovery and improved the attitude estimation algorithm performance for a more consistent collision recovery.

6.2 Recommendations for future work

There are many extensions and experiments that can be done to further validate and improve state estimation for collision recovery. The method to corrupt and model sensor data during a

collision could be improved through a vibrational analysis of the quadcopter after impacts. The effect of using a dynamics model, including the collision contact dynamics model, in the attitude estimation algorithms is worth investigation to see if it will improve the algorithms' performance. Future algorithms utilizing a GPS and estimating the position need to be implemented to enhance the collision recovery system for use in real environments. An investigation of the effects of a collision on the height estimate would be useful to determine the implications of the time spent in the height stabilization stage of the recovery process, as well as determine the effect of a collision on other quadcopter sensors.

Other experiments that would be of interest are to induce a small bias on the sensors, by calibrating the sensors indoors, and completing the experiments outdoors (or vice-versa). This could be used to validate the simulation results when using biased sensor data. It could also be of interest to implement the complementary filter in the PX4 flight stack in order to validate the use of Kalman based algorithms when faced with sensor biases. While implementation of the AH_∞ filter required the AH_∞ bound to be changed, no other tuning was performed on the filters and the values found from post-processing were used. It is likely that slightly better results could be obtained by tuning the filter parameters while using them in the control loop, as was seen through changing the AH_∞ bound. Performing the experiments on a vehicle designed to protect the magnetometer from the magnetic fields generated by the electronics would likely also improve the attitude estimates. Since it was not possible to implement the UKF based algorithms onboard the Pixhawk flight controller, the adaptive algorithms developed for the UKF could easily be used to modify the noise covariance matrices in the MEKF. However, it is difficult to say whether this would be worthwhile; the adaptive algorithms were found to be quite difficult to tune and sensitive to a small change in parameter values. As there are many other robust state estimation algorithms, the investigation of different designs of H_∞ filters could also lead to a better method to robustly estimate the state of the vehicle.

Appendix A

Extraneous Simulation Results

Table A.1 presents the average total Root Mean Square (RMS) attitude estimate errors for scenario 3 (the collision scenario) when the initial attitude error and initial gyroscope bias are set to be low, $\tilde{\sigma}_{\phi} = 0.1$ and $\tilde{\sigma}_{b_g} = 0.001$, which are ten times smaller than those expected under normal operating conditions. The complementary filter uses the value of k_b found in Table 5.2. The results show that the performance of the complementary filter is much more similar to the KF based algorithms when faced with lower than normal gyroscope bias and initial attitude error. The difference in the performance of the EKF and UKF algorithms is similar to their performance when faced with normal initial error and gyroscope bias values and the covariance matching AUKF performs the best overall. It is likely that the high gain AUKF and AHINF filter show little difference when compared to the UKF and EKF as the adaptive gain may not have changed during the collision effectively. The adaptive gain may not have changed due to lower innovation values caused by the lower gyroscope bias and initial attitude error.

TABLE A.1: Average RMS attitude error for scenario 3 (collision trajectory) with lower than normal intial attitude error and gyroscope bias ($\tilde{\sigma}_{\phi} = 0.1$ and $\tilde{\sigma}_{b_g} = 0.001$). Bold values correspond to the lowest error.

	Attitude Error [°]			
Estimator	Total	Roll	Pitch	Yaw
Complementary	1.63	0.373	0.463	1.49
MEKF	1.55	0.255	0.416	1.47
H_∞	1.79	0.288	0.479	1.70
AH_∞	1.55	0.255	0.416	1.47
UKF	1.47	0.235	0.400	1.39
$AUKF_1$	1.47	0.238	0.401	1.39
$AUKF_2$	**1.44**	**0.233**	**0.392**	**1.36**

Bibliography

[1] S. Gupte *et al.*, "A survey of quadrotor unmanned aerial vehicles", in *Proc. of IEEE SoutheastCon 2012*, 2012, pp. 1–6.

[2] T. M. Ravich, "The integration of unamnned aerial vehicles into the national airspace", *North Dakota Law Review*, vol. 85, p. 597, 2009.

[3] R. Weibel and R. J. Hansman, "An integrated approach to evaluating risk mitigation measures for UAV operational concepts in the NAS", in *Infotech@Aerospace*, 2005, p. 6957.

[4] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems", *Journal of Field Robotics*, vol. 29, no. 2, pp. 315–378, 2012.

[5] F. Chui *et al.*, "Dynamics of a quadrotor undergoing impact with a wall", in *International Conf. on Unmanned Aircraft Systems (ICUAS)*, 2016, pp. 717–726.

[6] G. Dicker *et al.*, "Quadrotor collision characterization and recovery control", in *IEEE International Conf. on Robotics and Automation (ICRA)*, 2017, pp. 717–726.

[7] M. D. Shuster, "A survey of attitude representations", *Navigation*, vol. 8, no. 9, pp. 439–517, 1993.

[8] J. L. Crassidis *et al.*, "Survey of nonlinear attitude estimation methods", *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 1, pp. 12–28, 2007.

[9] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. John Wiley & Sons, 2006.

[10] J. L. Crassidis and J. L. Junkins, *Optimal Estimation of Dynamic Systems*. CRC press, 2011.

[11] R. Zanetti *et al.*, "Norm-constrained Kalman filtering", *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 5, pp. 1458–1465, 2009.

[12] M. S. Arulampalam *et al.*, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking", *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[13] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation", in *IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, 2000, pp. 153–158.

[14] J. L. Crassidis and F. L. Markley, "Unscented filtering for spacecraft attitude estimation", *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 4, pp. 536–542, 2003.

[15] S. A. Chee and J. R. Forbes, "Norm-constrained unscented Kalman filter with application to high area-to-mass ratio space-debris tracking", in *AIAA Guidance, Navigation, and Control Conf.*, 2016, p. 1856.

[16] R. Van Der Merwe and E. A. Wan, "The square-root unscented Kalman filter for state and parameter-estimation", in *IEEE International Conf. on Acoustics, Speech, and Signal Processing*, 2001, pp. 3461–3464.

[17] F. Daum and J. Huang, "Curse of dimensionality and particle filters", in *Proc. of IEEE Aerospace Conf.*, vol. 4, 2003, 4_1979–4_1993.

[18] S. Qi and H. Jian-Da, "An adaptive UKF algorithm for the state and parameter estimations of a mobile robot", *Acta Automatica Sinica*, vol. 34, no. 1, pp. 72–79, 2008.

[19] R. Mahony *et al.*, "Nonlinear complementary filters on the special orthogonal group", *IEEE Transactions on Automatic Control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[20] J. F. Vasconcelos *et al.*, "A nonlinear observer for rigid body attitude estimation using vector observations", *IFAC Proc. Volumes*, vol. 41, no. 2, pp. 8599–8604, 2008.

[21] J. F. Guerrero-Castellanos *et al.*, "A robust nonlinear observer for real-time attitude estimation using low-cost mems inertial sensors", *Sensors*, vol. 13, no. 11, pp. 15 138–15 158, 2013.

[22] P.-J. Bristeau *et al.*, "The navigation and control technology inside the AR.Drone micro UAV", *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 1477–1484, 2011.

[23] K. D. Sebesta and N. Boizot, "A real-time adaptive high-gain EKF, applied to a quadcopter inertial navigation system", *IEEE Transactions on Industrial Electronics*, vol. 61, no. 1, pp. 495–503, 2014.

[24] C. Hajiyev and H. E. Soken, "Robust adaptive Kalman filter for estimation of UAV dynamics in the presence of sensor/actuator faults", *Aerospace Science and Technology*, vol. 28, no. 1, pp. 376–383, 2013.

[25] L. Xie *et al.*, "H infinity estimation for discrete-time linear uncertain systems", *International Journal of Robust and Nonlinear Control*, vol. 1, no. 2, pp. 111–123, 1991.

[26] H. Li and M. Fu, "A linear matrix inequality approach to robust H infinity filtering", *IEEE Transactions on Signal Processing*, vol. 45, no. 9, pp. 2338–2350, 1997.

[27] S. A. Chee and J. R. Forbes, "Discrete-time minmax filtering subject to a norm-constrained state estimate", *Automatica*, Under Review, 2016.

[28] A. Cavallo *et al.*, "Experimental comparison of sensor fusion algorithms for attitude estimation", *IFAC Proc. Volumes*, vol. 47, no. 3, pp. 7585–7591, 2014.

[29] G. M. Hoffmann *et al.*, "Quadrotor helicopter flight dynamics and control: Theory and experiment", in *Proc. of the AIAA Guidance, Navigation, and Control Conf.*, vol. 2, 2007.

[30] F. Chui, "Quadrotor collision dynamics and fuzzy logic characterization", Master's thesis, McGill University, Montreal, Canada, 2016.

[31] G. Dicker, "Quadrotor reorientation control for collision recovery", Master's thesis, McGill University, Montreal, Canada, 2016.

[32] J. Sola, "Quaternion kinematics for the error-state KF", *Laboratoire d'Analyse et d'Architecture des Systemes-Centre national de la recherche scientifique (LAAS-CNRS), Toulouse, France, Tech. Rep*, 2015.

[33] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors", *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[34] W. Flenniken, "Modeling inertial measurement units and anlyzing the effect of their errors in navigation applications", PhD thesis, Auburn University, Auburn, Alabama, 2005.

[35] R. Van Der Merwe and E. A. Wan, "Sigma-point Kalman filters for integrated navigation", in *Proceedings of the 60th Annual Meeting of the Institute of Navigation*, 2004, pp. 641–654.

[36] E. Thébault *et al.*, "International geomagnetic reference field: The 12th generation", *Earth, Planets and Space*, vol. 67, no. 1, p. 79, 2015.

[37] L. Meier, *PX4 Firmware*, https://github.com/PX4/Firmware, 2017.

[38] T. Karvonen, "Stability of linear and non-linear Kalman filters", Master's thesis, University of Helsinki, Helsinki, 2014.

[39] U. Shaked and Y. Theodor, "H infinity optimal estimation: A tutorial", in *Proc. of the 31st IEEE Conf. on Decision and Control*, 1992, pp. 2278–2286.

[40] I Yaesh and U Shaked, "Game theory approach to optimal linear estimation in the minimum H infinity-norm sense", in *Proc. of the 28th IEEE Conf. on Decision and Control*, 1989, pp. 421–425.

[41] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems", in *AeroSense'97*, 1997, pp. 182–193.

[42] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation", *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[43] L.-G. Zhang *et al.*, "Square-root unscented Kalman filter for vehicle integrated navigation", in *IEEE International Conf. on Machine Learning and Cybernetics*, vol. 1, 2007, pp. 556–561.

[44] M. Tailanián *et al.*, "Design and implementation of sensor data fusion for an autonomous quadrotor", in *Proc. of IEEE International Instrumentation and Measurement Technology Conf. (I2MTC)*, 2014, pp. 1431–1436.

[45] W. Li and J. Wang, "Effective adaptive Kalman filter for MEMS-IMU/magnetometers integrated attitude and heading reference systems", *Journal of Navigation*, vol. 66, no. 01, pp. 99–113, 2013.

[46] W. Premerlani, *Fast Rotations*, https://www.dropbox.com/s/zae23dr9ky8agiv/fastRotations.pdf?dl=0, 2011.

[47] M. Zhang *et al.*, "Autonomous flight of a quadrotor using multi-camera visual slam", in *Proc. of International Conf. on Intelligent Unmanned Systems*, vol. 10, 2014.

[48] S. Hong *et al.*, "Observability of error states in GPS/INS integration", *IEEE Transactions on Vehicular Technology*, vol. 54, no. 2, pp. 731–743, 2005.

[49] T. Ainscough *et al.*, "Q-method extended Kalman filter", *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 4, pp. 752–760, 2014.

[50] M. D. Shuster, "Kalman filtering of spacecraft attitude and the QUEST model", *Journal of the Astronautical Sciences*, vol. 38, pp. 377–393, 1990.

[51] F. L. Markley, "Unit quaternion from rotation matrix", *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 2, p. 440, 2008.

[52] F. L. Markley *et al.*, "Averaging quaternions", *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193–1197, 2007.