Faculty of Aerospace Engineering

Quadrotor Upset Recovery after Rotor Failure

Master of Science Thesis

Matthias Baert

23 June 2019

TU Delft Delft University of Technology

Challenge the future

Quadrotor Upset Recovery after Rotor Failure

Master of Science Thesis

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering at Delft University of Technology

Matthias Baert

23 June 2019

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

Delft University Of Technology
Department Of
Control and Simulation

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"Quadrotor Upset Recovery after Rotor Failure"** by **Matthias Baert** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 23 June 2019

Readers:

dr. ir. C. de Visser

dr. ir Q.P. Chu

External Member

Acronyms

AFTC	Active Fault Tolerant Control
AINDI	Adaptive Incremental Non-linear Dynamic Inversion
CCW	Counterclockwise
CFD	Computational Fluid Dynamics
CW	Clockwise
EKF	Extended Kalman Filter
EOM	Equations of Motion
FDI	Fault Detection and Isolation
FTC	Fault Tolerant Control
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INDI	Incremental Non-linear Dynamic Inversion
LMS	Least Mean Squares
LPF	Low Pass Filter
NDI	Non-linear Dynamic Inversion
PFTC	Passive Fault Tolerant Control
PID	Proportional Integral Derivative
UAV	Unmanned Aerial Vehicle

Glossary

G Control effectiveness matrix

I_v Inertia matrix of quadcopter

R_{IB} Rotation matrix

X_b Body X-axis

Y_b Body Y-axis

Z_b Body Z-axis

α_{LOC} Loss of control angle for altitude control

α_{peak} Peak angle for altitude control

α_{total} Angle between primary axis and primary axis target

$\triangle \boldsymbol{\Omega}_{des}$ Desired change in rotational rate

$\boldsymbol{\Omega}$ Rotational rates

$\boldsymbol{\omega}$ Rotor speeds

η_{red} Reduction factor

ϕ Roll angle

ψ Yaw angle

τ Total moment produced by all rotors

τ_{aero} Total moment produced by other aerodynamic effects

A Linear constraint matrix of the QP problem

F Jacobian of state function

H Hessian of the QP cost function

K Kalman Gain

M Jacobian of measurement function

$\mathbf{P}$ Covariance matrix

$\mathbf{S}$ Residual covariance

$\mathbf{a}^B_{ref,z}$ desired body acceleration

$\mathbf{b}$ Constraint matrix of the QP problem

$\mathbf{c}$ Linear part of the QP cost function

$\mathbf{c}_0$ Constant part of the QP cost function

$\mathbf{d}$ Position vector

$\mathbf{g}$ Gravity vector

$\mathbf{q}$ Quaternion describing attitude

$\mathbf{v}$ Velocity vector

$\mathbf{y}$ Measurement residual

θ Pitch angle

a_{des} Desired acceleration vector

a_z measured acceleration in z-direction

b Half width of quadrotor

c_{ld} drag/lift ratio

f State function

f_i Lift force rotor i produces

h Measurement function

l Half length of quadrotor

n_{des} Desired primary axis

$n_{rotaxis}$ Rotation axis

p Roll rate

q Pitch rate

r Yaw rate

u Input to the system

x State of system

x_{des} Desired x position

y_{des} Desired y position

z Measurements of the system

z_{des} Desired z position

Contents

List of Figures

General Introduction

This report summarises the thesis work done between September 2018 and July 2019. The report consists of three main parts, a scientific paper presented in Part I, the preliminary report in Part II and appendices to the paper in Part III. All essential information can be found in the paper which is a stand-alone document. However, more elaborate or extra information is presented in the appendices. The thesis aims to give an answer to the following research question:

How can a quadrotor with rotor failure recover from an upset condition and how does it compare to upset recovery in the nominal case?

Layout Scientific Paper

The paper is written following the IEEE standards. The introduction in Chapter **I** shows the importance of the research as well as a literature review is presented. The quadrotor model is presented in Chapter **II** where quaternions are used in order to model the dynamics. The methodology in Chapter **III** explains all parts of the proposed controller. This is the core of the paper. The experimental setup is briefly explained in Chapter **IV** and the results are presented in Chapter **V**. The report concludes with Chapter **VI**.

Layout Preliminary Report

The preliminary thesis report was first handed in at the end of November 2018. The introduction is presented in Chapter 1. The research motivation and research questions can be found in Chapter 2. A literature review is conducted in Chapter 3. The methodology in Chapter 4 presents the thesis focus and goals as well as the expected planning from that point in time. Some preliminary results can be seen in Chapter 6 where upset recovery of a nominal (not damaged) quadrotor is tested.

Layout Appendices

In Appendix A the quaternion based Extended Kalman Filter (EKF) will be explained more elaborate. The EKF was shortly mentioned in the paper but not worked out in detail as it was not the essence of the research. As the measurements of the EKF are not always available, this requires an interesting solution. The control allocator is a big part in the paper, however there are more practical details that were not mentioned in the paper. Therefore more information about the control allocation is given in Appendix B. The attitude control has become a complex system. An attempt has been made to explain it as short as possible in the paper but

in Appendix C some extra details about the estimated time to rotate will be explained. In order to develop all the methods presented in the paper, a simulator was built. This work is presented in Appendix D. The developed control system has to be deployed on the quadrotor. For this the PX4 framework was used. This is further explained in Appendix E. Additional visualisation of the results presented in the paper are presented in Appendix F.

Acknowledgement

I would like to thank my supervisors Coen de Visser and Sihao Sun for their guidance and help throughout the process of writing this thesis. I would also like to thank Bram Strack van Schijndel for his work on the implementation of the developed control systems into PX4 and the implementation of PX4 on the Bebop 2. Lastly I would like to thank Roy Vorster and Suzanne van den Boogaart for their help during the tests.

Conclusion and recommendations

The conclusion is already presented in the scientific paper. In this chapter the answers to the research questions will be given as well as some recommendations for future research.

Answer to research questions

Although most of the research questions can be answered by the information presented in the paper and the appendices, this section will specifically address the answers to the research questions established in the preliminary report.

The main research question: **How can a quadrotor with rotor failure recover from an upset condition and how does it compare to upset recovery in the nominal case?**

is what the paper is all about. Recovery of the nominal case and the damaged case is explained in detail. Although the recovery in both cases is quite different, the basics are the same. In the damaged case there are some extra steps required, but once the quadrotor is controllable the same controller method is used as in the nominal case. In the damaged case the quadrotor is less powerful as it basically loses half of its thrust, which causes the recovery to take longer.

Next to the main research question some sub-questions were also presented. An important question that is not answered in the paper is: **How can an upset condition be defined for a quadrotor?**

The definition of an upset condition depends on the context. One could state that an upset condition exists when some states are uncontrollable and/or unstable. One could also state that an upset condition is a condition in which the vehicle does not normally operate or a condition which is far away from the desired condition. For this paper the first definition is most applicable. One could even expand the definition and say that the state has to be uncontrollable for a certain time horizon for it to be an upset condition. Otherwise the hover condition would already be an upset condition as lateral position can not be controlled directly. One could also expand the definition by stating that one has to give up control or find a balance between states in order to recover from the upset condition. For example, when the quadrotor is upside down, altitude can not be controlled and has to be given up initially. Some steps have to be taken in order to make altitude controllable again.

How does a fully functional quadrotor recover from an upset condition?

This question is clearly answered in the paper. The key is to find the right balance between the different control demands. Prioritising pitch and roll demands seemed to be essential for recovery.

Can the same upset recovery methods be used for an undamaged and a damaged quadrotor?

Recovery for both undamaged and damaged condition are presented in the paper. Some parts of the control system can be used in both cases (eg. the control allocator, position control, altitude control, ..), however some extra controllers are needed in the damaged case.

Is there a specific set of upset conditions for which the damaged quadrotor can recover?

The quadrotor is able to recover from any condition, the real question is how much altitude loss is permitted. It was proven that for any initial position and attitude the quadrotor was able to recover, assuming no rotational rates.

Is there a specific set of upset conditions for which the damaged quadrotor cannot recover?

Initial conditions with specific initial rotational rates are harder to recover from. If initial yaw rate is given in the opposite direction of the natural yaw rate the quadrotor will generate, recovery takes significantly longer. Initial rotational rate around the uncontrollable axis also has a big impact on the recovery performance. From this research it does not seem that there are initial conditions that are impossible to recover from, assuming enough altitude available. If limited altitude is taken into account, then there are definitely conditions for which the quadrotor will not be able to recover.

Can a passive Fault Tolerant Control (FTC) be developed?

Although not clearly mentioned in the paper, the proposed control system is passive FTC. The control allocator will prioritise pitch and roll commands due to which yaw rate control will automatically be given up. The recovery performance of the passive FTC is significantly worse though.

How quick should the fault detection be?

This depends on the available altitude. As the quadrotor is passive FTC fault detection is not even needed. This however goes at serious cost in performance. During indoor tests fault detection delays were emulated up to 0.1s which the control system was still able to handle without losing more than 3m of altitude.

Can INDI be used in all cases?

The INDI principle can definitely be used for recovery, however the usual implementation which is proposed in several literature sources is not ideal [1, 7, 8]. As not all control requirements can always be met, a balance has to be made by the control allocator. This is solved by creating a QP-INDI as explained in the paper and Appendix B.

Recommendations

The state of the art control methods before this research were not able to recover a damaged quadrotor from an upset condition. During this research a lot of new strategies and methods were developed and validated, some more successful than others. When solving certain problems, new problems always arise. The final product presented in this thesis project still has plenty of aspects to be improved on which will be discussed in this section.

The current recovery strategy is very successful in recovering the damaged quadrotor from any initial attitude. However, in the case of big rotational rates the recovery is less successful, especially when there are rotational rates that either create an opposite yaw rate or rotational rates around the uncontrollable axis. Currently, the first two recovery steps consist of flipping the quadrotor to point up and then generate yaw rate. Looking at this from a different perspective one might say that the goal is to generate a certain angular momentum in the inertial frame. With this concept more abstract solutions might come out that could potentially improve the recovery performance significantly.

In order to determine how the quadrotor will rotate, the estimated time of rotation is calculated in both directions around a certain rotation axis. This assumption is valid as long as there are only rotational rates around that rotation axis (during a flip manoeuvre for example). However, in the case of rotational rates around the other axes, this time estimation will not be correct anymore and a better rotation trajectory might exist. In future research this should be improved on, but one should realise that this will add a lot of complexity to the problem.

For this thesis project all the tests were conducted in an indoor facility with limited space. Due to the limited space the quadrotor was not always able to recover in time. From simulation one can show that the quadrotor can recover in limited altitude, this however is not validated. If the quadrotor would be able to fly outdoors one can validate if the quadrotor is really able to recover within a certain altitude. Flying outdoors can be quite challenging with respect to state estimation. Indoors, position and attitude are given quite accurately, outdoors this is not the case. In the damaged case the quadrotor will spin very fast which will make state estimation even more challenging. It will be interesting to see if future research can manage to move the research to an outdoors environment.

Part I

Scientific paper

Quadrotor Upset Recovery after Rotor Failure

Matthias Baert
Supervisors: Sihao Sun, Coen de Visser

***Abstract*—A fault-tolerant controller is presented that is able to recover a quadrotor from an upset condition in the case of a single rotor failure. With only 3 rotors the quadrotor is not fully controllable anymore, however by using precession the quadrotor can be made controllable again in roll and pitch. A step-by-step recovery strategy is proposed in which controllability is actively recovered before recovery of attitude and altitude. Control allocation is based on the Incremental Non-Linear Dynamic Inversion (INDI) principle and takes rotor saturation into account by solving a constrained quadratic optimisation problem. The controller is validated in a real life test environment where the quadrotor is thrown into the air with only 3 propellers from which it has to recover using the techniques presented in this paper.**

***Index Terms*—Robot Safety, fault-tolerant, recovery, sensor-based control.**

I. Introduction

In recent years, multi-rotor aerial vehicles have received a lot of attention. These aerial vehicles are usually unmanned robots that can perform various tasks, in some cases without human intervention. Multi-rotors are mainly used outdoors for agricultural purposes, architecture and construction, delivery, emergency services, media purposes or to monitor and conserve the environment [1]. As these vehicles will become more involved in daily life, safety can not be overlooked.

One of the most common multi-rotors is the quadrotor due to its simplicity and energy efficiency [2]. As the name implies, a quadrotor has four rotors positioned in a rectangular profile on the vehicle. Quadrotors are extremely powerful which makes them very agile and able to fly under many different circumstances. However, because this vehicle is not over-actuated, this type of multi-rotor suffers most from an actuator failure and might not be able to continue its mission or worse, might not be able to land safely.

Fault-tolerant control for quadrotors has been the subject of various literature sources. Some research is focused on the partial damage of a rotor [3, 4], while other research considers the complete loss of one or multiple rotors [5, 6, 7].

A solution to the case of a complete loss of a rotor is presented in Lanzon et al. where the author proposes to give up on yaw control in order to maintain control over the other states [6]. Mueller and D'Andrea present an analytical solution when one, two or three propellers are completely lost [5]. Lippiello et al. present both a PID and backstepping approach focusing on an emergency landing in case of failure [8, 9]. A fault tolerant controller using INDI is presented by Lu and Van Kampen where fault detection is also implemented [10]. An interesting view on hover conditions is presented in Mueller and D'Andrea where the hover conditions are relaxed such that hover solutions can be found for the under actuated damaged quadrotor [11].

In Sun et al. the theory is put into practise by showing that a quadrotor with loss of single rotor is able to fly in high speed conditions [7]. This research also uses the INDI method with reduced attitude control. The research does not take rotor saturation into account. This means that at high speeds (9 m/s) the quadcopter loses control as the rotors start saturating. In Höppener, D an improved INDI control allocation method is presented which takes rotor saturation into account [12]. This method is validated with partial damage on one of the propellers. In Zhang et al. fault detection in combination with fault-tolerant control is presented and validated in a real life environment [13] .

Recovery or more specific upset recovery is rarely considered in quadrotor research. In Faessler et al. the recovery of a quadrotor is discussed, but the focus is on the recovery of the vision-based state estimation [14]. The quadrotor is thrown into the air by hand in order to create an upset condition. For airplanes upset recovery is a more common field of study. Many literature sources propose a step-by-step recovery process when an airplane is in an upset condition [15, 16]. Plenty of literature sources cover trajectory generation and control for quadrotors, which is also related to recovery [17, 18, 19]. Hehn and Andrea presents a trajectory generation solution, that should be able to solve the recovery problem in an abstract way [20].

The combination of a fault-tolerant controller and upset recovery however, is not considered yet. When a failure occurs, it is likely that the quadrotor will be in an upset condition. In this paper a fault-tolerant solution is presented which is able to recover from upset conditions. A quaternion based Extended Kalman Filter is used for state estimation while some modifications to existing position control methods are applied. A new look on attitude control is presented in which controllability is considered. This combined with an INDI control allocator that takes rotor saturation into account and is able to prioritise between different commands. The methods are validated in a real-life environment where the quadrotor is thrown into the air to simulate upset conditions. This paper uses the work presented in Sun et al. as a starting point [7].

In Section II the model of the quadrotor used is explained. Section III presents the proposed fault tolerant control system that is able to recover from an upset condition. In Section IV the experimental setup is explained while in Section V the results are presented. The paper concludes with Section VI.

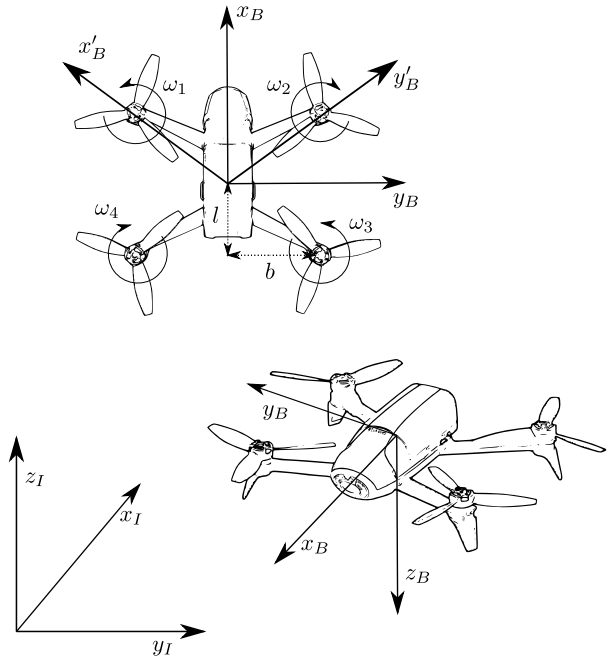

Fig. 1. Quadrotor with axis definitions

II. Quadrotor Model

The Parrot Bebop 2 quadrotor as well as the axis definitions are shown in Figure 1. The inertial frame is defined by the vectors x_I, y_I, z_I and will be used as the reference coordinate system. The body frame is defined by the vectors x_B, y_B, z_B. Alternatively one can also use the non-orthogonal body frame defined by x'_B, y'_B, z_B which has interesting properties. The quadrotor has 4 rotors (ω_{1-4}) which are located at a distance of b along the y_B axis and l along the x_B axis. The spinning direction is also indicated in Figure 1.

The equations of motion are based on [21] and [19] and are given as

$$\dot{\boldsymbol{d}} = \boldsymbol{v} \tag{1}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{g} + \boldsymbol{q} \otimes \frac{1}{m} \boldsymbol{F}^B \otimes \boldsymbol{q}^* \tag{2}$$

$$\boldsymbol{I}_v \dot{\boldsymbol{\Omega}} = -\boldsymbol{\Omega} \times \boldsymbol{I}_v \boldsymbol{\Omega} + \boldsymbol{M}^B \tag{3}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2} \boldsymbol{\Omega} \otimes \boldsymbol{q} \tag{4}$$

where position and velocity are denoted by $\boldsymbol{d}$ and $\boldsymbol{v}$ respectively. $\boldsymbol{q}$ is the quaternion expressing the attitude of the quadrotor, $\boldsymbol{g}$ is the gravitational acceleration, m and $\boldsymbol{I}_v$ are the mass and inertia of the vehicle, $\boldsymbol{F}^B$ and $\boldsymbol{M}^B$ are the forces and moments acting on the quadrotor. The $\otimes$ operator represents the Kronecker product and is used when quaternions are involved [21]. The rotational rate is denoted by $\boldsymbol{\Omega}$.

The aerodynamic forces and moments acting on the quadrotor come from two sources: rotors ($_r$) and body ($_b$) as shown in

$$\boldsymbol{F}^B = \boldsymbol{F}_r + \boldsymbol{F}_b \tag{5}$$

$$\boldsymbol{M}^B = \boldsymbol{M}_r + \boldsymbol{M}_b \tag{6}$$

The aerodynamic force produced by the rotors ($\boldsymbol{F}_r$) can be described by

$$f_i \sim c_f \omega_i^2, \quad \boldsymbol{F}_r = \begin{bmatrix} 0 \\ 0 \\ -\sum f_i \end{bmatrix} \tag{7}$$

where c_f is a lift coefficient. Note that this force acts in the z_B axis. The moments created by the rotors ($\boldsymbol{M}_r$) is described as follows

$$\boldsymbol{M}_r = \begin{bmatrix} b & -b & -b & b \\ l & l & -l & -l \\ \frac{1}{c_{ld}} & -\frac{1}{c_{ld}} & \frac{1}{c_{ld}} & -\frac{1}{c_{ld}} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{8}$$

where the coefficient c_{ld} is the lift-over-drag ratio of the rotors. The aerodynamic moments coming from the body ($\boldsymbol{M}_b$) are approximated by

$$\boldsymbol{M}_b = - \parallel \boldsymbol{\Omega} \parallel \boldsymbol{K}_d \boldsymbol{\Omega}, \quad \boldsymbol{F}_b = 0 \tag{9}$$

where $\boldsymbol{K}_d$ is a drag coefficient matrix. The aerodynamic forces coming from the body ($\boldsymbol{F}_b$) are neglected.

III. Methodology

A controller will be presented that is able to recover from any condition in both nominal (no damage) and single rotor failure condition. In order to do this the controller is split up in several subsystems. The controller is presented in Figure 2 and consists of a state estimation part, a P + PI position controller, a complex attitude controller and a control allocator based on Incremental Nonlinear Dynamic Inversion (INDI). It will be assumed that the failure is known by the system, as estimation of the failure is beyond the scope of this paper.

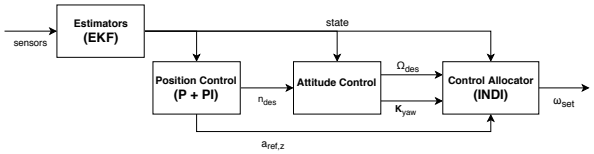

Fig. 2. Controller overview

A. State Estimation

Attitude and position information is available via a motion capture system but in some cases the motion capture system might lose tracking. In order to cope with these situations as well as enhancing the attitude, position and velocity estimation in general, a quaternion based Extended Kalman Filter (EKF) was used based on the model presented before. In the case where the motion capture system does not update anymore, only the state ahead prediction of the EKF will be used.

Additionally a free-fall detection has been implemented to allow a person to throw the quadrotor into the air from which it has to recover. Free-fall is detected if $rms(\boldsymbol{a}_{meas}) \leq a_{lim}$ for a certain amount of time. As the quadrotor has some aerodynamic drag during free-fall, the accelerometer will not measure 0 m/s^2. From empirical data a_{lim} was chosen to be 4 m/s^2.

B. Position control

For position control a cascaded P + PI controller is used as follows

$$\begin{aligned} \boldsymbol{v}_{ref} &= k_p^{pos}(\boldsymbol{d}_{des} - \boldsymbol{d}), \\ \boldsymbol{a}_{ref} &= k_p^{vel}(\boldsymbol{v}_{ref} - \boldsymbol{v}) + k_i^{vel} \int (\boldsymbol{v}_{ref} - \boldsymbol{v})dt - \boldsymbol{g} \end{aligned} \tag{10}$$

The output is a desired normalised acceleration vector. ($\boldsymbol{n}_{des}^I$). $k_p^{pos}, k_p^{vel}, k_i^{vel}$ are the gains of the position controller. An integrator has been added in the velocity loop to counteract any disturbances like wind. Note that the position controller acts in the inertial frame. Limits on the desired acceleration vector are necessary to make sure the quadrotor would not push itself into an upset condition. A maximum angle from the normal (θ_{max}) is applied as follows

$$\lambda = \frac{\sqrt{\boldsymbol{a}_{ref,x}^2 + \boldsymbol{a}_{ref,y}^2}}{\boldsymbol{a}_{ref,z} \cdot tan(\theta_{max})} \tag{11}$$

$$\boldsymbol{a}_{ref,xy} = \frac{\boldsymbol{a}_{ref,xy}}{max(\lambda, 1)} \tag{12}$$

$$\boldsymbol{n}_{des}^I = \frac{\boldsymbol{a}_{ref}}{\parallel \boldsymbol{a}_{ref} \parallel} \tag{13}$$

If the initial reference is outside of the bounds, λ will be greater than 1 and will scale $\boldsymbol{a}_{ref}$ down. This vector is then normalised, indicating the desired thrust direction ($\boldsymbol{n}_{des}^I$) of the quadrotor.

Altitude control requires an extra step as it is related to the amount of thrust the rotors have to produce, which is in the body frame. The desired z-axis acceleration is converted from the inertial frame to the body frame as follows

$$\boldsymbol{a}_{ref,z}^B = \frac{\boldsymbol{a}_{ref,z}^I}{cos(\theta)} \tag{14}$$

where θ indicates the total angle from the normal described as

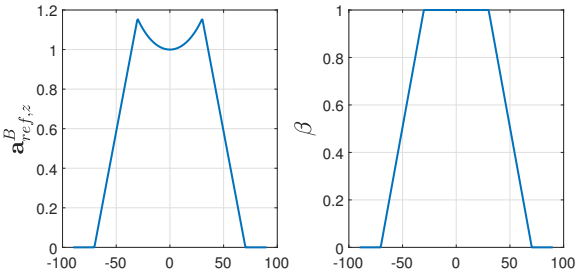

Fig. 3. Altitude control scaling visually presented. At extreme angles the altitude action will be reduced.

$$cos(\theta) = \frac{-\boldsymbol{g} \cdot \boldsymbol{n}^I}{\parallel \boldsymbol{g} \parallel \cdot \parallel \boldsymbol{n}^I \parallel} \tag{15}$$

where $\boldsymbol{n}^I$ is the thrust vector in the inertial frame. However, in extreme conditions ($\theta = \pm\frac{\pi}{2}$) Equation (14) becomes singular and can not be used. Also, at extreme angles this method would create a lot of lateral displacement. A balance has to be made where altitude control is given up at some point. An extra scaling factor β is introduced which is capped between 0 and 1 as shown in

$$\beta = 1 - \frac{\theta - \theta_1}{\theta_2 - \theta_1}, \quad 0 \leq \beta \leq 1 \tag{16}$$

The scaling factor β will linearly reduce the altitude control demand from θ_1, at θ_2 altitude control is given up completely. The revised calculation of $\boldsymbol{a}_{ref,z}^B$ is

$$\boldsymbol{a}_{ref,z}^B = \frac{\boldsymbol{a}_{ref,z}^I}{cos(min(\theta, \theta_1))} \cdot \beta \tag{17}$$

where θ_1 and θ_2 are chosen to be 30 deg and 70 deg respectively. Additionally the total angle used is also capped in the denominator. The effect of this revised calculation is visualised in Figure 3 where $\boldsymbol{a}_{ref,z}^I = 1$ is assumed.

C. Attitude Control

The goal is to come up with rotational rate setpoints in order to control the attitude. Attitude control plays a major role in recovery of the quadrotor. First the nominal case (without damage) will be considered.

1) Nominal Case: From position control a desired thrust vector ($\boldsymbol{n}_{des}^I$) is given in the inertial frame and has to be converted to the body frame ($\boldsymbol{n}_{des}^B$) as follows

$$\boldsymbol{n}_{des}^B = \boldsymbol{q} \otimes \boldsymbol{n}_{des}^I \otimes \boldsymbol{q}^* \tag{18}$$

where $\boldsymbol{q}$ represents the quaternion describing the quadrotor attitude. A normalised rotation vector ($\boldsymbol{\rho}^B$) is calculated as

$$\boldsymbol{\rho}^B = \frac{\boldsymbol{n}_{des}^B \times \boldsymbol{n}^B}{\parallel \boldsymbol{n}_{des}^B \times \boldsymbol{n}^B \parallel} \tag{19}$$

with the thrust vector $\boldsymbol{n}^B = [0, 0, -1]$. The desired pitch and roll rate ($\boldsymbol{\Omega}_{xy,des}$) can be calculated using

$$\boldsymbol{\Omega}_{xy,des} = k_p^{rot}(\boldsymbol{\rho}^B \cdot \alpha) \tag{20}$$

where the total angle (α) is calculated as

$$cos(\alpha) = \frac{\boldsymbol{n}_{des}^B \cdot \boldsymbol{n}^B}{\parallel \boldsymbol{n}_{des}^B \parallel \cdot \parallel \boldsymbol{n}^B \parallel} \tag{21}$$

Using this method, the desired pitch and roll rates are determined, the desired yaw rate (r_{des}) comes from a separate yaw controller

$$r_{des} = k_p^{yaw}(\psi_{des} - \psi) \tag{22}$$

where ψ and ψ_{des} indicates the yaw angle and desired yaw angle of the vehicle respectively. If the quadrotor has some rotational velocity already, the aforementioned solution might not be the optimal solution. The shortest route might not be the quickest route, therefore an estimate of the time it takes to rotate is calculated

$$\begin{aligned} 0 = &-\alpha_0 + \Omega t + \Omega_0^2 \left(\frac{t \cdot a_{min} + \Omega_0}{a_{min} - a_{max}}\right) \frac{a_{max}}{a_{min}} + \\ &\left(\frac{t \cdot a_{min} + \Omega_0}{a_{min} - a_{max}}\right)^2 \left(1 + \frac{a_{max}}{a_{min}}\right) \frac{a_{max}}{2} \end{aligned} \tag{23}$$

where t is the time it takes to rotate α_0 given an initial Ω_0 and minimum and maximum rotational acceleration a_{min}, a_{max}. Ω_0 is the rotation speed around the desired rotation vector $\boldsymbol{\rho}^B$ and is calculated as follows

$$\Omega_0 = \boldsymbol{\Omega} \cdot \boldsymbol{\rho}^B \tag{24}$$

One is now able to determine the time to rotate around $\boldsymbol{\rho}^B$ and $-\boldsymbol{\rho}^B$ and decide which direction is quickest. This rotation vector will then be chosen.

2) Failure Case: In case of failure, attitude control becomes more challenging. Without loss of generality, failure on rotor 1 will be assumed. Looking at the achievable moments in Figure 4, which gives an indication on controllability, it can be seen that no positive moment around the y'_B axis can be produced (red area), meaning this axis is uncontrollable. Note that the non-orthogonal body frame (x'_B, y'_B) is now being used as this simplifies calculations. In order to make the vehicle controllable again, precession can be added. Precession can be generated as shown in

$$My'_{prec} = \Omega_z \Omega_{x'}(I_{y'} - I_z) \tag{25}$$

meaning rotational rates in the z_B (Ω_z) and x'_B (Ω_x) axis are needed, which are controllable. In Figure 4 one can see the shift in the achievable moments (green area), making the quadrotor controllable again. Note that in the case of failure on rotor 1, the quadrotor will end up having negative yaw rate.

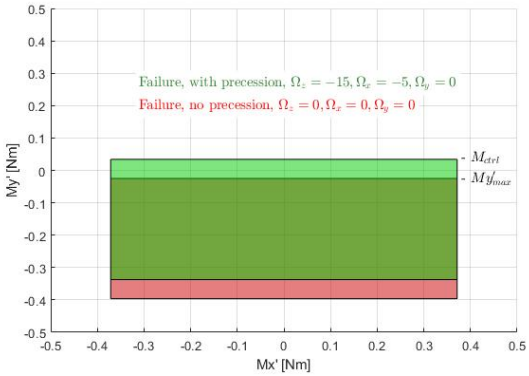

Fig. 4. Achievable moments in the x'_B, y'_B axis frame following failure on rotor 1. Red area is no precession while green area does have some precession added. The green area is controllable while the red area is not.

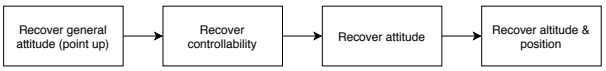

Fig. 5. Recovery strategy overview

In order to recover the quadrotor from an upset condition, a set of recovery steps have to be taken as shown in Figure 5. The first step is to rotate the quadrotor such that the thrust vector is pointing 'up', hence in the negative z direction. Secondly controllability has to be recovered by generating yaw rate (Ω_z) and some roll rate ($\Omega_{x'}$). Once the quadrotor is controllable again, attitude, altitude and position can be recovered.

First step is to recover general attitude, meaning the thrust vector of the quadrotor should point up. Yaw rate will be generated eventually, so it is important to have the quadrotor point upwards first, otherwise it will require a lot of energy to flip the quadrotor. With failure on rotor 1, rotation around the x'_B axis is still fully controllable, so this axis will be used to flip the quadrotor. The same procedure is followed as in the nominal case, but $\boldsymbol{n}_{des}^B$ and $\boldsymbol{n}^B$ are projected on the y'_B axis, meaning the x component of the vector is zero. However, cases exists in which the x'_B axis is parallel with the normal meaning the quadrotor can not be flipped around that axis. In this case first some yaw rate will be generated, causing the x'_B axis to move and eventually the quadrotor can flip around this axis again.

Once the general attitude is recovered and the quadrotor is pointing upwards, controllability will be recovered. This is done by generating precession. A high yaw rate demand is set in order to generate precession as quickly as possible. Because this is so important, the weight of the yaw action in the control allocator is increased as shown in

$$\begin{aligned} &dW_{yaw} = k_p^{prio}(\Omega_{z,des} - \Omega_z), \quad 0 \leq dW_{yaw} \\ &W_{yaw} = W_{yaw}^0 + dW_{yaw} \end{aligned} \tag{26}$$

where k_p^{prio} is the gain, W_{yaw} is the yaw action weight

and dW_{yaw} is the additional weight in order to increase the priority. The weight priority is only increased in case there is not enough yaw rate, once the yaw rate is sufficient the priority will be at the original level again. The magnitude of $\Omega_{z,des}$ is a tuning parameter and is chosen to be $15rad/s$. The sign of $\Omega_{z,des}$ in this case is negative but depends on which rotor failed as there will only be one feasible yaw rate direction. The amount of precession needed ($My'_{prec,des}$) can be calculated as shown in

$$My'_{prec,des} = M_{ctrl} - My'_{max} \tag{27}$$

where M_{ctrl} is a tuning parameter indicating the minimum controllability margin and My'_{max} is the current maximum moment that can be produced around the y'_B axis. If $My'_{max} \geq M_{ctrl}$ no precession will be added because the quadrotor is controllable enough. Using Equation (25) the required $\Omega_{x'}$ can be determined.

This results in a conflicting requirement with the attitude controller as $\Omega_{x'}$ is used to control the roll angle of the quadrotor. The requirement from the precession module can be regarded as a bias added to the attitude control demand but another method is by tilting the thrust vector ($\boldsymbol{n}^B$) slightly, causing the quadrotor to 'wobble' and have a steady-state roll/pitch rate [5]. Previously $\boldsymbol{n}^B$ was chosen to be $[0, 0, -1]$, however from

$$\begin{bmatrix} \dot{n}_1 \\ \dot{n}_2 \\ \dot{n}_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix} \boldsymbol{\Omega} \tag{28}$$

where $\boldsymbol{n}^B = [n_1, n_2, n_3]$, one can see that in case $\Omega_z \neq 0$ there exist stable solutions where $n_1 \neq 0$ and/or $n_2 \neq 0$. In this example $\Omega_{x'} \neq 0$ and $\Omega_{y'} = 0$, this means that $n_1 = \frac{\Omega_{x'} n_3}{\Omega_z}$ and $n_2 = 0$. In this way precession can be generated without influencing the attitude control. This is an elegant solution and does not require any change in the other parts of the controller.

Once the controllability has been recovered, the attitude control as presented in Section III-C1 is able to control the attitude again. To optimise the overall performance of the controller even further the rate setpoints are hedged such that only achievable targets are requested to the control allocator, also known as pseudo control hedging [22]. The hedging also protects the quadrotor from pushing itself into an uncontrollable situation. Desired pitch and roll rate limits are calculated in

$$\begin{aligned} \Omega_{x'y',des,min} &= \Omega_{x'y'} - \dot{\Omega}_{x'y',min} \cdot t_{horiz} \\ \Omega_{x'y',des,max} &= \Omega_{x'y'} + \dot{\Omega}_{x'y',max} \cdot t_{horiz} \end{aligned} \tag{29}$$

where the time horizon (t_{horiz}) is a tuning parameter and chosen to be 0.05s. Assuming the rotor failure is known, an estimate can be made on $\dot{\Omega}_{x'y',min}$ and $\dot{\Omega}_{x'y',max}$. An example calculation is shown in

$$\dot{\Omega}_{x',max} = \frac{(c_f \omega^2_{4,max} - c_f \omega^2_{2,min}) s}{I_{x'}} \tag{30}$$

where $s = \sqrt{b^2 + l^2}$ is the distance from the rotor to the center of gravity. This equation can be used to calculate the other limits as well. In order to make sure the quadrotor does not push itself into an uncontrollable condition, an extra limit is applied as shown in

$$\begin{aligned} \dot{\Omega}_{x'y',min} &= max(\dot{\Omega}_{x'y',min}, -\dot{\Omega}_{x'y',max}) \\ \dot{\Omega}_{x'y',max} &= min(\dot{\Omega}_{x'y',max}, -\dot{\Omega}_{x'y',min}) \end{aligned} \tag{31}$$

which is a symmetric limit that is applied such that the quadrotor will not generate rotation rates that can not be dissipated.

D. Control allocation

The goal of the control allocator is to match the rotational rate setpoints ($\boldsymbol{\Omega}_{des}$) and the altitude control demand ($\boldsymbol{a}^B_{ref,z}$). It is not always feasible to match all the demands, certainly not in the failure case. This means there has to be some balance between the different demands, which also gives the ability to prioritise certain demands. The control method that will be used is based on INDI, which has proven to be very powerful in the field of quadrotors [23, 7]. In order to balance out the demands, a quadratic cost function has been set up that will be minimised. This makes it a quadratic programming problem that can be solved. This method will be called Quadratic Programing INDI (QP-INDI).

A P-controller is used to determine the required change in rotational rate and as can be seen in

$$\triangle \boldsymbol{\Omega}_{des} = k_p^{prio} (\boldsymbol{\Omega}_{des} - \boldsymbol{\Omega}) \tag{32}$$

where k_p^{prio} is the gain. Assuming the control effectiveness matrix $\boldsymbol{G}$ is known, or an estimate is available, the required change in rotor speeds can be calculated using [7]

$$\boldsymbol{E} = \begin{bmatrix} \triangle \boldsymbol{\Omega}_{des} \\ \boldsymbol{a}^B_{ref,z} \end{bmatrix} - \begin{bmatrix} \dot{\boldsymbol{\Omega}} \\ a_z \end{bmatrix} \tag{33}$$

$$\boldsymbol{E} = \boldsymbol{G} \triangle \boldsymbol{\omega} \quad \rightarrow \quad \triangle \boldsymbol{\omega} = \boldsymbol{G}^{-1} \boldsymbol{E} \tag{34}$$

In the basic form, all demands have to be followed and there is no balance between the demands. In order to solve this a new cost function is presented in

$$\min \quad (\boldsymbol{G} \triangle \boldsymbol{\omega} - \boldsymbol{E})^T \boldsymbol{K} (\boldsymbol{G} \triangle \boldsymbol{\omega} - \boldsymbol{E}) + \triangle \boldsymbol{\omega}^T \boldsymbol{K}_2 \triangle \boldsymbol{\omega} \tag{35}$$

where $\boldsymbol{K}$ indicates the weights for each demand. In case that all demands can be met, the small penalty $\boldsymbol{K}_2$ will make sure the solution of least change in rotor speeds will be chosen. Next to the cost function there are constraints as well as shown in

$$\triangle \boldsymbol{\omega}_{min} \leq \triangle \boldsymbol{\omega} \leq \triangle \boldsymbol{\omega}_{max} \tag{36}$$

This problem can be solved by using Quadratic Programming (QP). If the problem can be rewritten to the form shown in

$$\begin{aligned} &\text{minimise} \quad \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{c}_0 \\ &\text{subject to} \quad \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} \end{aligned} \tag{37}$$

then the optimum can be found by solving the KKT conditions [24] shown in

$$\begin{aligned} &(1) \quad \boldsymbol{H}\boldsymbol{x} + \boldsymbol{c} = \boldsymbol{A}^T\boldsymbol{y}, \\ &(2) \quad \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \\ &(3) \quad \boldsymbol{y} \geq 0, \\ &(4) \quad \boldsymbol{y}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) = 0 \end{aligned} \tag{38}$$

where the solution (x) is the incremental change in rotor speed ($\triangle\boldsymbol{\omega}$). Solving this problem comes down to finding the correct Lagrange multipliers ($\boldsymbol{y}$), which is similar to an active-set method. Given a choice of Lagrange multipliers, the solution is given by

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{A}_{eq}^T \\ \boldsymbol{A}_{eq} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{c} \\ \boldsymbol{b}_{eq} \end{bmatrix} \tag{39}$$

where $\boldsymbol{A}_{eq}$ and $\boldsymbol{b}_{eq}$ are the $\boldsymbol{A}$ and $\boldsymbol{b}$ matrices with only the rows of the active Lagrange multipliers. If the solution satisfies the requirements from Equation (38) then the optimum is found. The QP problem is solved on board in real time. The Lagrange multipliers from the previous timestep can always be used as the first guess, which greatly reduces the computation time. Rewriting Equation (35) into the standard QP form, results in the values presented in

$$\begin{aligned} &\boldsymbol{H} = \boldsymbol{G}^T\boldsymbol{K}\boldsymbol{G} + \boldsymbol{K}_2, \\ &\boldsymbol{c} = \boldsymbol{G}^T\boldsymbol{K}\boldsymbol{E} + (\boldsymbol{E}^T\boldsymbol{K}\boldsymbol{G})^T), \\ &\boldsymbol{c}_0 = 0 \end{aligned} \tag{40}$$

The choice of $\boldsymbol{K}$ and $\boldsymbol{K}_2$ can be found in Equations (41) and (42). These weights determine which demands are prioritised. The first two elements on the diagonal represent the pitch and roll weight, which are assigned a high weight as they are essential for overall control. The third element is the weight for yaw action, which is assigned a low gain. The last element on the diagonal is the weight for altitude control. With this method, the quadrotor is passive fault tolerant to a certain degree as it will give up on yaw control as a consequence of the weights given.

$$\boldsymbol{K} = \text{diag}(1 \times 10^4, 1 \times 10^4, 1, 1 \times 10^2) \tag{41}$$

$$\boldsymbol{K}_2 = \boldsymbol{I} \times 10^{-4} \tag{42}$$

IV. Experimental Setup

To verify and validate the methods developed in Section III a variety of experiments are set up. The quadrotor that is used in all the experiments is the Parrot Bebop 2. For some experiments the nominal configuration with 4 rotors is used, while in other experiments the 3 rotor configuration is used as shown in Figure 6. The quadrotor is modified by removing the camera and changing the battery in order to reduce overall mass. The mass, inertia, geometry and rotor parameters can be found in Table I. The vehicle is equipped with a Parrot P7 dual-core CPU Cortex 9 while the onboard IMU is a MPU6050 with 512 Hz sampling rate. On the top side of the quadrotors there are at least 6 reflecting markers which are used by a motion capture system from Optitrack in order to provide position and attitude information to the vehicle. The information is sent via WiFi to the quadrotor. PX4 is used as framework in which modules are added that contain the developed control system. The control systems are developed in MATLAB Simulink. By creating a simulation environment in Simulink, the first tests can already be done to verify the control logic. The same tests can then be performed in a real-life environment to validate. The real-life environment that is used is the Cyberzoo at the TU Delft, The Netherlands. The Cyberzoo is a safe area of about 6x6x6 meters surrounded by nets such that aerial vehicles can never escape in case something goes wrong.

Fig. 6. Parrot Bebop 2 in 3 rotor configuration.

TABLE I
Parrot Bebop 2 parameters

m [kg]	0.374	l [m]	0.088
I_{xx} [kgm^2]	1.67×10^{-3}	b [m]	0.115
I_{yy} [kgm^2]	1.38×10^{-3}	ω_{max} [rpm]	12000
I_{zz} [kgm^2]	2.82×10^{-3}	ω_{min} [rpm]	3000

The tests that will be performed and analysed can be categorised as follows: A) upset recovery without rotor failure, B) recovery from sudden rotor failure in hover C) upset recovery with rotor failure.

A. Upset recovery without rotor failure

The quadrotor has to be able to recover from upset conditions in the nominal (no damage, 4 rotor configuration) case first, before making it more challenging with rotor failure. Upset recovery will be tested by throwing the quadrotor by hand into a random condition, from which it has to recover. If free-fall is detected, the control systems will take over and recover from the condition it is in. Various throws will be performed in which the person throwing will be given a

certain intention (throw gently, give a lot of rotation, etc..). The quadrotor will always aim to fly towards a given position setpoint. An analysis on the chance of recovery can be performed depending on the initial state.

B. Recovery from sudden rotor failure in hover

In this test the quadrotor will take-off with 4 rotors and fly to a position setpoint. Once in stable hover, a failure will be simulated in which one of the rotors will be set to minimum speed. The quadrotor then has to recover from this failure and fly back to the position setpoint.

C. Upset recovery with rotor failure

In this test one of the propellers will be removed from one of the rotors in order to simulate a full rotor failure. Similarly as before, the quadrotor can be thrown by hand into a random condition from which it has to recover. The effect of the initial condition will be analysed as well as the effect of the precession controller and the effect of the yaw rate priority controller. The precession controller can be disabled as well as the yaw rate priority controller in order to see the impact.

V. Results

In this section the results of the tests described in section IV will be elaborated upon. First upset recovery of a fully functional quadrotor is tested, secondly recovery from a sudden rotor failure is tested and finally upset recovery of a quadrotor with single rotor failure is tested. A video summary of the results can be seen in https://youtu.be/5UUsu7OlGd0.

A. Upset recovery without rotor failure

In Figure 7 The response of the quadrotor following a throw can be seen. Initially the rotor speeds are set to idle speed. Once free-fall is detected, the quadrotor recovers from the condition it is in. Important to note that the quadrotor is thrown with upwards velocity, giving some extra room in terms of altitude. One can see that pitch and roll are prioritised over yaw angle. A 3D plot of the position and attitude can be seen in Figure 8. Here one can clearly see that the quadrotor is thrown into the air, recovers and returns to the position setpoint. The control allocation is visualised in Figure 9 where one can see the priority to the roll and pitch commands over the yaw and acceleration demands.

This test is performed 10 times with the same person throwing the quadrotor. In all of the cases the quadrotor manages to recover.

B. Recovery from sudden rotor failure in hover

In this test the quadrotor takes off in the 4-rotor configuration and while in hover, one of the rotors will be disabled by setting it to idle speed. One of the test results can be seen in Figure 10. One can see that the quadrotor immediately starts to generate yaw rate as well as some roll and pitch rate in order to create precession. At a certain point the quadrotor is fully controllable again and attitude and position can be recovered. Note the constant pitch and roll rate in the hover condition at the end.

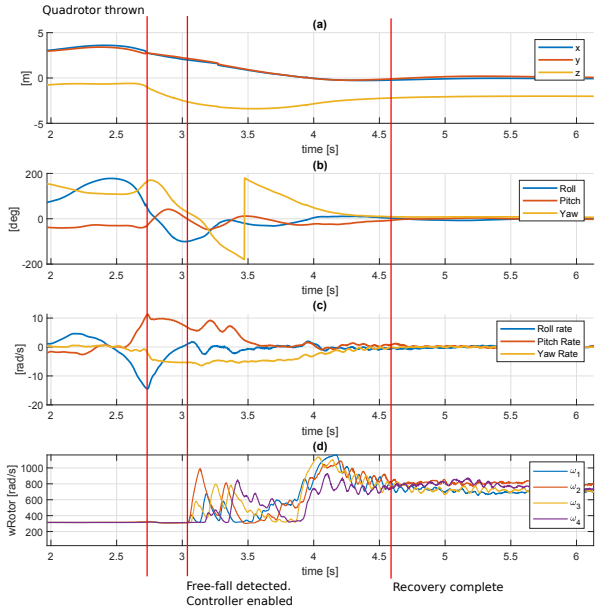

Fig. 7. Quadrotor thrown into the air. (a) Position. (b) Attitude. (c) Rotational velocity. (d) Rotor speeds. The controller detects free-fall, re-enables and manages to recover.

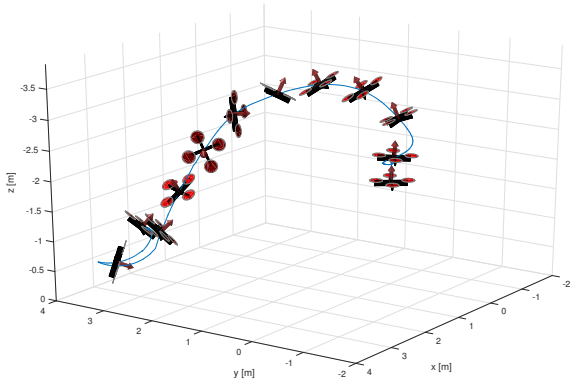

Fig. 8. Quadrotor thrown into the air, recovers and lands - position 3D plot.

This test is performed 10 times. In all of the cases the quadrotor manages to recover. The quadrotor loses a maximum of 0.2m in altitude and has a maximum lateral displacement of 0.5m in all the tests. Due to the yaw rate priority when rotor failure occurs, the quadrotor is able to generate up to 9 rad/s of yaw rate in 0.5s. Failure on other rotors were also tested and give similar results.

C. Upset recovery with rotor failure

In this test the quadrotor with one propeller removed is thrown into the air by hand. The result can be seen in Figure 11 where the quadrotor is thrown with high pitch rate and some pitch angle from which it has to recover. The vehicle first makes sure it is 'pointing up' before generating precession. Once there is enough yaw rate the quadrotor is controllable

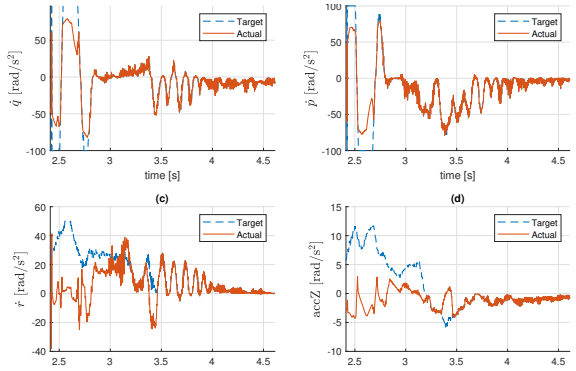

Fig. 9. QP-INDI allocation during recovery from throw. (a) roll command. (b) pitch command. (c) yaw command. (d) acceleration demand. Roll and pitch command are prioritised.

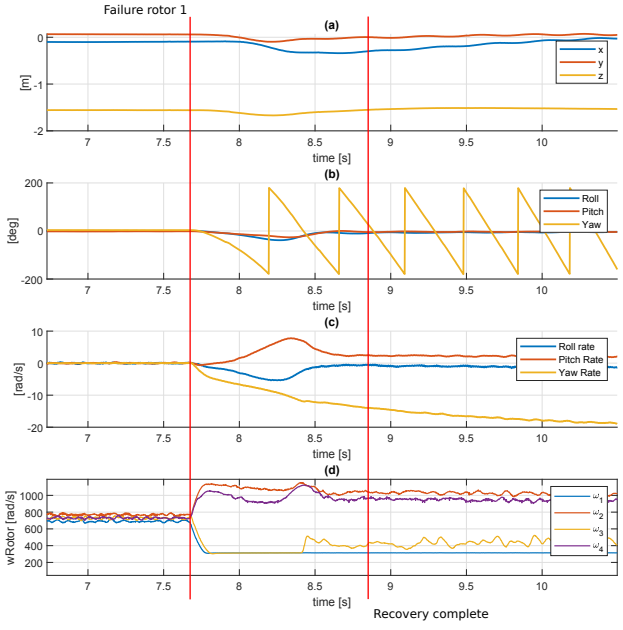

Fig. 10. Quadrotor sudden failure on rotor 1 from hover condition. (a) Position. (b) Attitude. (c) Rotational velocity. (d) Rotor speeds.

again and attitude and position can be recovered easily. A 3D plot of the quadrotor during this manoeuvre can be found in Figure 12.

This test is performed 65 times with different people throwing the quadrotor. In 46 of the cases the quadrotor manages to recover, while in 19 cases it did not fully recover as can be seen in Figure 13. In the case the quadrotor did not recover, 13 times did the quadrotor already recover attitude and was in progress of recovering altitude resulting in a rough landing ($<$ 3m/s) causing no damage to the vehicle. In 6 cases the quadrotor was still recovering attitude which resulted in a hard crash. From this data it can be concluded that there is a 71% chance of full recovery, 20% chance of partial recovery with rough landing and a 9% chance of a crash. This conclusion however is biased, as recovery rate highly depends on the

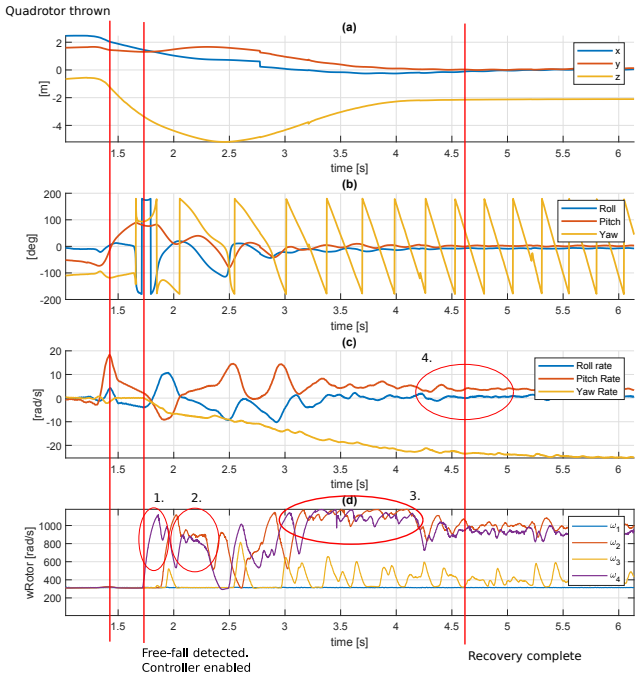

Fig. 11. Quadrotor thrown into the air with 3 rotors. (a) Position. (b) Attitude. (c) Rotational velocity. (d) Rotor speeds. The controller detects free-fall, re-enables and manages to recover. First the quadrotor is flipped using the controllable axis (1.), secondly yaw-rate is being generated (2.). In (3.) altitude is being recovered. In (4.) one can see the constant pitch rate that is generated to create precession.

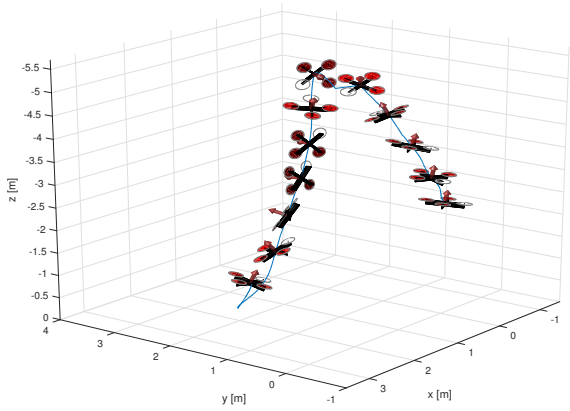

Fig. 12. Quadrotor thrown into the air with 3 rotors - position 3D plot

way the quadrotor is thrown into the air. This test was also performed in simulation. 500 throws were simulated and it could be seen that in the worst case the quadrotor needed less than 30m of altitude in order to recover completely. The simulations had no limit on initial attitude, the rotational rates were limited to ± 10 rad/s for Ω_x and Ω_y and ± 5 for Ω_z. No initial velocity or position was given in order to be able to analyse the required altitude in a fair way.

In Figure 14 a scatter plot is made in which the state of

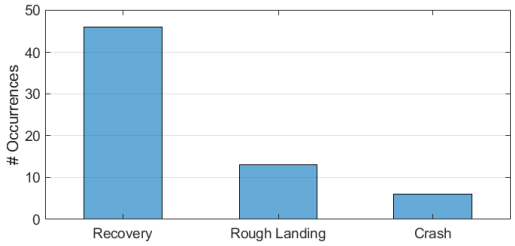

Fig. 13. Recovery rate of 65 throws in damaged (3 rotor configuration) condition

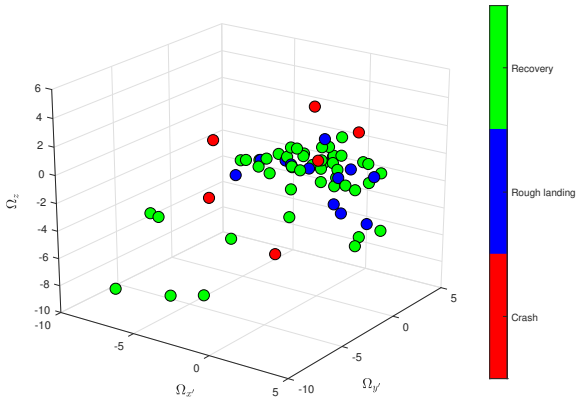

Fig. 14. Single rotor failure recovery depending on initial state after being thrown into the air - 3D scatter.

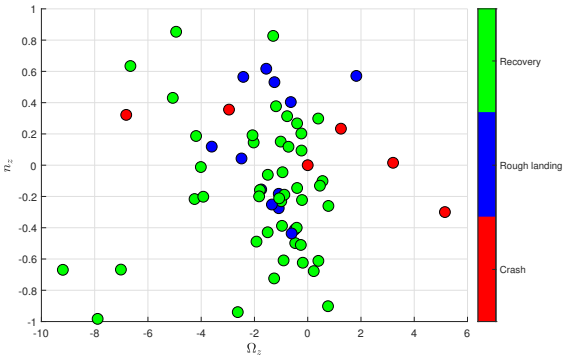

Fig. 15. Single rotor failure recovery depending on initial state after being thrown into the air - 2D scatter.

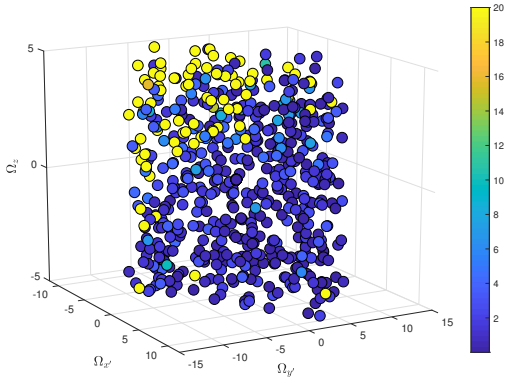

Fig. 16. Single rotor failure recovery altitude needed in meter - Simulation data - 3D scatter.

recovery is plotted versus some initial conditions. The initial conditions are defined as the conditions at which the quadrotor detects free-fall and is re-enabled. The initial conditions selected are the rotational rates $\Omega_{x'}$, $\Omega_{y'}$ and Ω_z. The test data does not cover the whole range of possibilities, so it is hard to identify a clear tend.

A 2D scatter plot where Ω_z versus the z-component of the thrust vector (n_z) is compared can be seen in Figure 15. Although there is little data, one can see that in case of positive yaw rate, the quadrotor has more difficulties recovering. This makes sense as it will take the quadrotor longer to generate the correct yaw rate, which is negative for failure on rotor 1.

Upwards velocity would also be a good candidate to consider as initial condition, however during all the tests performed the upwards velocity was more or less the same.

As the 3D figure of real data was inconclusive, 500 additional simulations were performed in order to see if a more clear trend could be found. The altitude needed to recover is presented in Figure 16 where $\Omega_{x'}$, $\Omega_{y'}$ and Ω_z are considered as initial conditions. It can be confirmed that having initial positive Ω_z is bad for recovery, but one can also see that having negative $\Omega_{y'}$ is also bad for recovery. This can be explained by the fact that around this axis, the quadrotor is uncontrollable. So negative $\Omega_{y'}$ can not be dissipated as there is a failure on rotor 1.

In Figure 17 a scatter of recovery altitude is made in similar fashion but with n_x, n_y and n_z as initial conditions (initial attitude) and only small initial rotation rates are considered ($\text{rms}(\mathbf{\Omega}) < 2$ rad/s). One can see that in all these cases the quadrotor manages to recover in less than 10m. This result shows that the controller is robust to initial attitude, but has a harder time recovering from initial rotation rates.

In Section III a yaw rate priority controller was proposed which would push the quadrotor to generate yaw rate in case of failure. Some tests were done in which this controller was switched off, the results can be seen in Figure 18. Clearly the recovery rate is much lower without this controller. When analysing the data in Figure 19, one can clearly see that yaw rate was generated too slow, causing a slower recovery. In case there is more altitude available, the quadrotor would also be able to recover eventually, as yaw rate is generated passively anyways.

The importance of tilting the thrust vector to actively generate precession was tested as well. The results can be seen in Figure 20. One can see that these results are comparable to the results presented in Figure 13. Looking at Figure 21 one can see that there is still some roll rate creating precession,

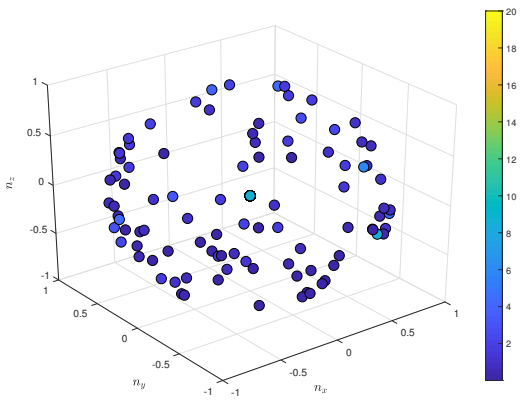

Fig. 17. Single rotor failure recovery altitude needed without initial rotation rates - Simulation data - 3D scatter.

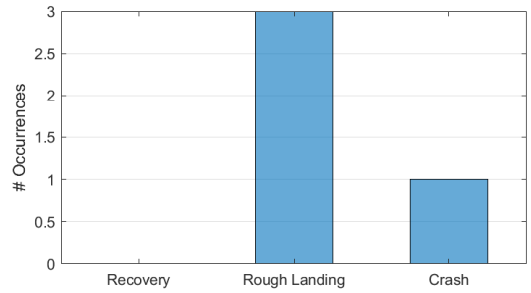

Fig. 18. Recovery rate of 4 throws in damaged condition without yaw rate priority

although one would expect this to be zero in this mode. Comparing with Figure 11 one can see that less precession is generated as the rotor speeds ω_3 are quite different. Although with this mode the quadrotor is closer to neutral stability, it is still sufficient to recover and hover.

VI. Conclusion

A fault-tolerant controller is presented that is able to recover from an upset condition in both nominal (without damage) and single rotor failure configuration. The control system is validated in a real-life test environment by throwing the quadrotor by hand in order to bring the quadrotor in a random state from which it has to recover. The research shows that even with a rotor failure, the quadrotor is able to recover. Given the limited altitude in the test facility, the quadrotor was able to recover in 71% of the cases, 20% of the cases resulted in a rough landing and 9% resulted in a crash. Analysis of the data, supported with extra simulation data, shows that the quadrotor can easily recover from any initial attitude. However, initial rotational rates have more impact on the recovery performance. From simulation it can be shown that the quadrotor can recover from any condition in less than 30m assuming no initial downwards velocity.

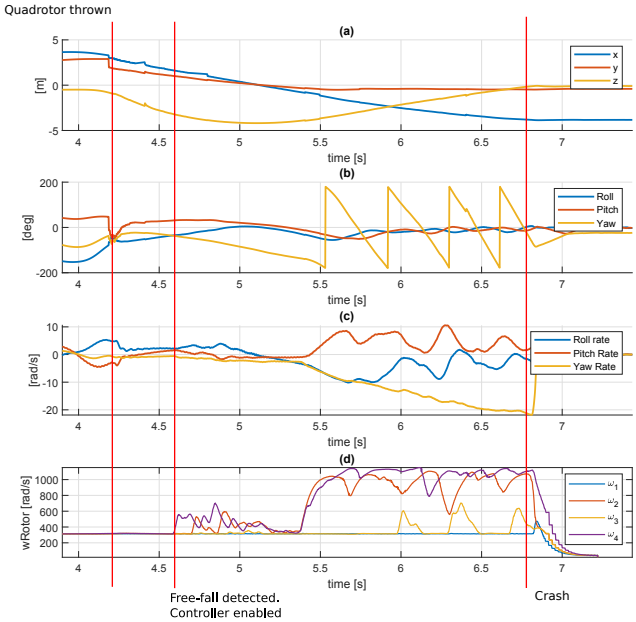

Fig. 19. Quadrotor thrown into the air with 3 propellers without yaw rate priority. (a) Position. (b) Attitude. (c) Rotational velocity. (d) Rotor speeds. The controller detects free-fall, re-enables but does not manage to recover.

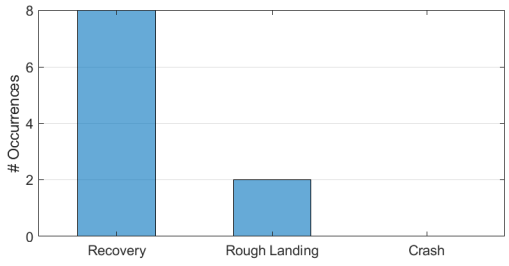

Fig. 20. Recovery rate of 10 throws in damaged condition without actively generating precession

A key element of this research is to actively recover pitch/roll controllability by adding precession to the system. Instead of completely ignoring yaw rate control, it is used to make the quadrotor controllable again. Another key element is the introduction of QP-INDI, which is an INDI based control allocator that is able to take rotor saturation into account. A weight is given to every demand such that the allocator can prioritise the most important commands. This makes the control system fault-tolerant as automatically the control systems will be able to maintain control over certain states while others are given up.

References

[1] Adam C. Uzialko. 10 cool commercial drone uses coming to a sky near you, May 2018.

[2] Joshuah K. Stolaroff, Constantine Samaras, Emma R. O'Neill, Alia Lubers, Alexandra S. Mitchell, and Daniel Ceperley. Author Correction: Energy use and life cycle greenhouse gas emissions of drones for commercial package delivery (Nature Communications (2018) DOI: 10.1038/s41467-017-02411-5). *Nature Communications*, 9(1):1–13,

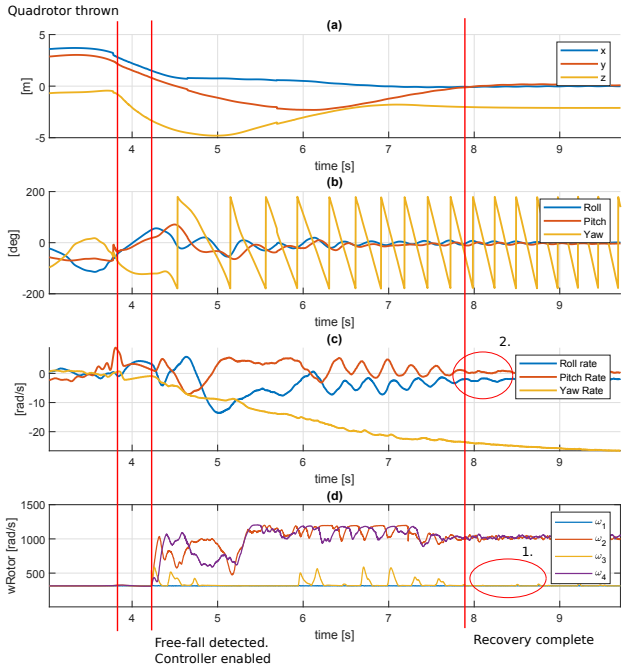

Fig. 21. Quadrotor thrown into the air with 3 propellers without actively generating precession. (a) Position. (b) Attitude. (c) Rotational velocity. (d) Rotor speeds. The controller detects free-fall, re-enables and manages to recover. This method results in a different steady-state solution where rotor 3 is not used anymore.

2018. ISSN 20411723. doi: 10.1038/s41467-018-03457-9. URL http://dx.doi.org/10.1038/s41467-017-02411-5.

[3] Liu Xulin and Guo Yuying. Fault tolerant control of a quadrotor UAV using control allocation. In *Proceedings of the 30th Chinese Control and Decision Conference, CCDC 2018*, pages 1818–1824, 2018. ISBN 9781538612439. doi: 10.1109/CCDC.2018.8407422.

[4] Tong Li, Youmin Zhang, and Brandon W Gordon. Passive and active nonlinear fault- tolerant control of a quadrotor unmanned aerial vehicle based on the sliding mode control technique. 227(1):12–23, 2012. doi: 10.1177/0959651812455293.

[5] Mark W. Mueller and Raffaello D'Andrea. Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 45–52, 2014. ISSN 10504729. doi: 10.1109/ICRA.2014.6906588.

[6] Alexander Lanzon, Alessandro Freddi, and Sauro Longhi. Flight Control of a Quadrotor Vehicle Subsequent to a Rotor Failure. *Journal of Guidance, Control, and Dynamics*, 37(2):580–591, 2014. ISSN 0731-5090. doi: 10.2514/1.59869. URL http://arc.aiaa.org/doi/10.2514/1.59869.

[7] Sihao Sun, Leon Marinus Christiaan Sijbers, Xuerui Wang, and Coen de Visser. High-Speed Flight of Quadrotor despite Loss of Single Rotor. *IEEE Robotics and Automation Letters*, 3(4):1–1, 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2851028. URL https://ieeexplore.ieee.org/document/8398406/.

[8] Vincenzo Lippiello, Fabio Ruggiero, and Diana Serra. Emergency landing for a quadrotor in case of a propeller failure: A PID approach. *IEEE International Conference on Intelligent Robots and Systems*, pages 4782–4788, 2014. ISSN 21530866. doi: 10.1109/IROS.2014.6943242.

[9] Vincenzo Lippiello, Fabio Ruggiero, and Diana Serra. Emergency landing for a quadrotor in case of a propeller failure: A backstepping approach. *IEEE International Conference on Intelligent Robots and Systems*, (Iros):4782–4788, 2014. ISSN 21530866. doi: 10.1109/IROS.2014.6943242.

[10] Peng Lu and Erik Jan Van Kampen. Active fault-tolerant control for quadrotors subjected to a complete rotor failure. *IEEE International Conference on Intelligent Robots and Systems*, 2015-Decem:4698–4703, 2015. ISSN 21530866. doi: 10.1109/IROS.2015.7354046.

[11] Mark W. Mueller and Raffaello D'Andrea. Relaxed hover solutions for multicopters: Application to algorithmic redundancy and novel vehicles. *International Journal of Robotics Research*, 35(8):873–889, 2016. ISSN 17413176. doi: 10.1177/0278364915596233.

[12] Höppener, D. Actuator Saturation Handling using Weighted Optimal Con- trol Allocation Applied to an INDI Controlled Quadcopter.

[13] Y. M. Zhang, A. Chamseddine, C. A. Rabbath, B. W. Gordon, C. Y. Su, S. Rakheja, C. Fulford, J. Apkarian, and P. Gosselin. Development of advanced FDD and FTC techniques with application to an unmanned quadrotor helicopter testbed. *Journal of the Franklin Institute*, 350(9): 2396–2422, 2013. ISSN 00160032. doi: 10.1016/j.jfranklin.2013.01.009.

[14] Matthias Faessler, Flavio Fontana, Christian Forster, and Davide Scaramuzza. Automatic Re-Initialization and Failure Recovery for Aggressive Flight with a Monocular Vision-Based Quadrotor. 2016.

[15] Jacobus A. Engelbrecht, Simon J. Pauck, and Iain K. Peddle. A Multi-Mode Upset Recovery Flight Control System for Large Transport Aircraft. *AIAA Guidance, Navigation, and Control (GNC) Conference*, pages 1–18, 2013. doi: 10.2514/6.2013-5172. URL http://arc.aiaa.org/doi/10.2514/6.2013-5172.

[16] Luis Crespo, Sean Kenny, David Cox, and Daniel Murri. Analysis of Control Strategies for Aircraft Flight Upset Recovery. *AIAA Guidance, Navigation, and Control Conference*, pages 1–31, 2012. ISSN 02555476. doi: 10.2514/6.2012-5026. URL http://arc.aiaa.org/doi/10.2514/6.2012-5026.

[17] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors. 2014. doi: 10.1007/978-3-642-28572-1.

[18] Mark W Mueller, Markus Hehn, and Raffaello D Andrea. A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation. *IEEE Transactions on Robotics*, 31(6):1294–1310, 2015. doi: 10.1109/TRO.2015.2479878.

[19] Ayman A. El-Badawy and Mohamed A. Bakr. Quadcopter Aggressive Maneuvers along Singular Configurations: An Energy-Quaternion Based Approach. *Journal of Control Science and Engineering*, 2016, 2016. ISSN 16875257. doi: 10.1155/2016/7324540.

[20] Markus Hehn and Raffaello D Andrea. Real-Time Trajectory Generation for Quadrocopters. *IEEE Transactions on Robotics*, 31:877–892, 2015. doi: 10.1109/TRO.2015.2432611.

[21] Emil Fresk and George Nikolakopoulos. Experimental evaluation of a full quaternion based attitude quadrotor controller. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2015-Octob:3864–3869, 2015. ISSN 19460759. doi: 10.1109/ETFA.2015.7301555.

[22] Q P Chu and J A Mulder. Flight Envelope Protection. *Control*, (August), 2010.

[23] Ewoud J. J. Smeur, Qiping Chu, and Guido C. H. E. de Croon. Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Air Vehicles. *Journal of Guidance, Control, and Dynamics*, 39(3):450–461, 2016. ISSN 0731-5090. doi: 10.2514/1.G001490. URL http://arc.aiaa.org/doi/10.2514/1.G001490.

[24] Amol Sasane and Krister Svanberg. *Optimization (SF1811 / SF1831 / SF1841) Amol Sasane and Krister Svanberg*. 2014.

Part II

Preliminary report

Chapter 1

Introduction

In recent years, multi-rotor aerial vehicles have received a lot of attention. These aerial vehicles are usually unmanned robots that can perform various tasks, in some cases without human intervention. Multi-rotors are mainly used outdoors for agricultural purposes, architecture and construction, delivery, emergency services, media purposes or to monitor and conserve the environment [9]. As this technology is still in its early days, the amount of applications is growing every day and existing applications can still be improved. Multi-rotors will get more and more involved in the daily life. It is therefore important that research continues to improve the quality, reliability and safety of these aerial vehicles [10].

One of the most common multi-rotors is the quadrotor due to its simplicity and energy efficiency [11]. As the name implies, a quadrotor has four rotors positioned in a rectangular profile on the vehicle. Quadrotors are extremely powerful which makes them very agile and able to fly under many different circumstances. The agility of the quadrotor makes it suitable to solve certain problems that otherwise require a complex structure or human intervention. As quadrotors will become even more common in daily life, safety cannot be neglected and should be taken into account when designing the vehicle and its mission.

When thinking about safety one should consider what happens when a quadrotor is damaged. Is the quadrotor then still able to continue its mission, or is it able to land in a safe manner? Say one of the rotors fails during flight, the quadrotor will become under actuated which might have critical consequences for the quadrotor and the mission it is performing. The partial or complete failure of a rotor has been described by various authors and solutions do exist in which the quadrotor is capable of controlling position [12]. This means the quadrotor can at least fly to a certain position and land safely using FTC.

Research in fault tolerant control for quadrotors is young but promising as shown. Most literature assumes to detect failure or damage immediately and the quadrotor to be in a favourable initial position, however in reality this is most likely not the case. The transition from nominal flight condition to damaged flight condition is very abrupt and can cause the vehicle to end up in an abnormal attitude and position, also called upset condition. The longer the failure detection takes, the more the quadrotor will have deviated from the hover condition. Although a quadrotor is very agile, recovering from an upset position might not always be that straight-forward, certainly if the quadrotor is damaged.

This thesis project will therefore focus on the upset recovery for damaged quadrotors. As little research has been conducted on this subject but promising developments are being made in FTC, the subject is suitable for a thesis assignment.

The outline of the report is as follows: in Chapter 2 the motivation for this research will be elaborated on as well as the research question. In Chapter 3 the literature review will be presented. The methodology can be found in Chapter 4 while the thesis plan can be found in Chapter 5. In Chapter 6 some preliminary tests are performed and the results are presented.

Chapter 2

Thesis Project

This thesis project will conduct research in the field of FTC and more specifically in the recovery of damaged quadrotors. The research motivation will be presented in Section 2-1 followed by the research question and sub-questions in Section 2-2.

2-1 Research Motivation

Unmanned Aerial Vehicles (UAVs) are becoming increasingly popular, not only in military use but also in commercial use. Especially the multi-rotors are popular due to the extreme power and agility. The industry is starting to focus more on the usage of these aerial vehicles for various tasks [10]. However, the current regulations [13] concerning the usage of UAVs are very strict and do not allow the full potential of the UAVs to be used. There are various reasons why the rules are so strict, but the main reason is safety. An UAV that loses control can cause significant harm to people in the neighbourhood, especially heavy UAVs. The current commercial UAVs do not have fault tolerant control capabilities yet, so more research needs to be done to make UAVs safer. Once it can be proven that UAVs are safe, the regulations can be relaxed so that the full potential can be exploited.

FTC for multi-rotors is a young but promising field of development. Specifically when taking quadrotors into consideration, damage or a complete rotor failure causes the vehicle to be under actuated. This means that not all states can be controlled anymore at the same time, therefore an alternative control strategy is required. As the literature review will show, various FTC solutions for quadrotors are being developed and some have already proved to be very effective. However, all these solutions assume a favourable initial condition and do not consider initial conditions that are way outside the normal hover conditions, which will be called upset condition. When a quadrotor gets damaged, this can either be due to an internal failure or due to a collision. In the case of a collision, the vehicle will most likely be in a weird state or upset condition. In the case of another failure, take for example a sudden rotor failure, the control systems need some time to detect the fault and act accordingly. During this time the quadrotor will deviate from the hover condition and could enter an upset condition. To be useful, the FTC needs to be able to recover from these conditions.

Little literature about recovery of quadrotors is available, certainly in case of damage. Therefore more research into this subject has to be done so that the flight envelope of the quadrotor with FTC can be expanded.

2-2 Research Questions

The research question defines the thesis project. It is therefore important to come up with a good research question. In order to make the research question specific enough, the research will focus on rotor failures for a quadrotor and how they compare to the nominal case. In sub-questions one can then expand to see how other forms of damage would affect the result. The thesis project is particularly interested in the recovery from a specific state, which will be called an upset condition. Therefore the research question for this thesis project will be:

- How can a quadrotor with rotor failure recover from an upset condition and how does it compare to upset recovery in the nominal case?

In order to answer these questions some sub-questions need to be answered and can be seen as intermediate goals for the thesis project.

As there is little literature about upset conditions with quadrotors, the author needs to define the upset condition of a quadrotor.

- How can an upset condition be defined for a quadrotor?

Assuming upset conditions exist for both nominal and the damaged case, the author needs to figure out how a quadrotor can recover. This is closely related to the main research question as if there is no solution there is no way to recover.

- How does a fully functional quadrotor recover from an upset condition?
- Can the same upset recovery methods be used for an undamaged and a damaged quadrotor?
- Is there a specific set of upset conditions for which the damaged quadrotor can recover?
- Is there a specific set of upset conditions for which the damaged quadrotor cannot recover?

In the case that recovery is not possible, the main research question can be answered negatively, one can again expand the thesis project and wonder why this is the case.

- Why does a quadrotor fail to recover from an upset condition?
- Why does a quadrotor with rotor failure fail to recover from an upset condition?

The research question asks how the quadrotor can recover. This means some questions regarding the control methods used need to be answered.

- How quick should the fault detection be?
- Can INDI be used in all cases?
- Can a passive FTC be developed?

Clearly, many questions arise when performing this research and one could possibly think of many more. The research questions above form a solid basis to start the thesis project.

Chapter 3

Literature Research

A literature research is performed to get an overview of the state-of-the-art technology concerning FTC for quadrotors. In order to truly understand the problem, the author will familiarise himself with the modelling of quadrotors in Section 3-1. Before one can design a fault tolerant controller one needs to have sufficient knowledge about the control of a quadrotor in nominal condition, which will be presented in Section 3-2. The FTC literature research is presented in Section 3-3 and finally upset recovery will be treated in Section 3-4.

3-1 Quadrotor Model

A quadrotor is a type of multi-rotor that uses, as the name suggests, four propellers to generate lift. The propellers are fixed pitch blades all pointing upwards. Control of the vehicle is therefore completely done by altering the lift each propeller produces. A quadrotor does not depend on big lifting surfaces (eg. wings) that require the vehicle to have some airspeed. This makes a quadrotor very agile and able to operate in many circumstances. In Figure 3-1 one can see an example of a quadrotor as well as the definition of the body axis system and the numbering of the rotors used in this thesis project. Remark that rotors 1 and 3 rotate counterclockwise while rotors 2 and 4 rotate clockwise. This direction difference enables the quadrotor to control its rotation around the Z_b axis.

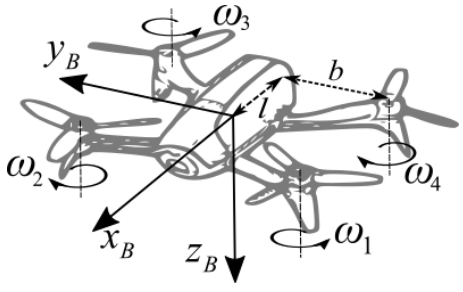

Figure 3-1: Bebop axis definitions [1]

In this section the modelling of a quadrotor will be discussed. It is important to have a good representation of the system and to know what the limits of the model are. A problem in the

model will likely propagate into the final system. The quadrotor model will be the basis from where the development can start. In most papers about control of quadrotors the quadrotor model used is presented. This is done by deriving the Equations of Motion (EOM) and simplifying where necessary. The EOM are generally split up in two parts being the translational part and the rotational part. One might argue that the aerodynamic model should also be part of the EOM, but most literature consider it as a separate subject. Attitude representation is usually done in two ways; either using Euler angles or using quaternions [14]. Euler angles are used far more often than the quaternion approach as it is more intuitive but it has its limitations. Two forms of EOM will therefore be presented in Section 3-1-1 and Section 3-1-2.

Before going into depth into the EOM, some variables will be defined. Every rotor i produces a force f_i which is aligned with the Z_b axis. The total moment produced by the rotors around the centre of gravity is denoted by τ. The calculation can be seen in Equation (3-1) where dimensions b and l are used as well as a drag/lift coefficient c_{ld}. Note that the Z_b points downwards. How the forces themselves are generated will be shown in Section 3-1-3.

$$\begin{bmatrix}\tau_x\\ \tau_y\\ \tau_z\end{bmatrix} = \begin{bmatrix}-f_1 b + f_2 b + f_3 b - f_4 b\\ -f_1 l - f_2 l + f_3 l + f_4 l\\ -f_1 c_{ld} + f_2 c_{ld} - f_3 c_{ld} + f_4 c_{ld}\end{bmatrix} + \tau_{aero} \tag{3-1}$$

3-1-1 EOM using Euler angles

The translational EOM of a quadrotor are quite straight forward as the quadrotor is not a complex model. They can be seen in Equation (3-2) and Equation (3-3) and are represented in the inertial frame [15]. The rotation matrix used is presented in Equation (3-4) [15] where sine and cosine are abbreviated by s and c respectively. The rotation matrix is used to convert the forces from the body frame to the inertial frame. From the EOM one can already see that the quadrotor can only control its lateral position by using roll (θ) and/or pitch (ϕ).

$$\dot{d} = v \tag{3-2}$$

$$\dot{v} = \begin{bmatrix}0\\0\\g\end{bmatrix} + R_{IB} \cdot \frac{1}{m}\begin{bmatrix}0\\0\\\sum f_i\end{bmatrix} \tag{3-3}$$

$$R_{IB} = \begin{bmatrix}c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi\\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi\\ -c\phi s\theta & s\phi & c\phi c\theta\end{bmatrix} \tag{3-4}$$

The rotational EOM are slightly more complex. The change in rotation matrix can be seen in Equation (3-5) while the rotational acceleration is calculated as shown in Equation (3-6) [15] [1]. The rotational velocity of the quadrotor is denoted by $\mathbf{\Omega}$ ($= [p, q, r]$) while the inertia of the quadrotor is denoted by I_v. Using this method one does not know what the actual Euler angles are, which can be essential to know. Equation (3-7) [16] offers a solution to this problem and can be used either next to Equation (3-5) to only calculate the Euler angles or can also be used to calculate the rotation matrix R_{IB}. In order to calculate the Euler angles one has to perform a matrix inversion, which can be dangerous and lead to singularities. This method is therefore only valid if $-\frac{\pi}{2} < \phi < \frac{\pi}{2}$ and $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$ [17].

$$\dot{R}_{IB} = R_{IB}\Omega \tag{3-5}$$

$$I_v\dot{\Omega} = -\Omega \times I_v\Omega + \tau \tag{3-6}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\phi & c\phi c\theta \end{bmatrix}^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3-7}$$

3-1-2 EOM using Quaternions

Less popular is the quaternion approach to model the quadrotor due to the increased complexity. As all info about the attitude is condensed into one 4 dimensional vector, it can be hard to see what is going on, making it less intuitive to use. Quaternions are 4 dimensional vectors that are an extension to the complex numbers, or also called the hyper complex numbers [18]. Quaternions are used in pure mathematics but come of good use for practical purposes such as modelling the rotation dynamics of a quadrotor. Quaternions can be expressed in various ways, but the two most common are shown in Equation (3-8) and Equation (3-9). Equation (3-10) shows how one can convert Euler angles to a quaternion where α is the total rotation angle around the vector. In Figure 3-2 a multiplication table is shown where one can already see the unique properties of quaternions.

$$\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k \tag{3-8}$$

$$\mathbf{q} = [q_0 + q_1 + q_2 + q_3] \tag{3-9}$$

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} cos(\alpha/2) \\ sin(\alpha/2)cos(\psi/2) \\ sin(\alpha/2)cos(\theta/2) \\ sin(\alpha/2)cos(\phi/2) \end{bmatrix} \tag{3-10}$$

×	**1**	***i***	***j***	***k***
1	1	*i*	*j*	*k*
i	*i*	−1	*k*	−*j*
j	*j*	−*k*	−1	*i*
k	*k*	*j*	−*i*	−1

Figure 3-2: Quaternion multiplication table [2]

The translational EOM can be found in Equation (3-11) and Equation (3-12) [19] [20]. They are very similar to the ones presented in the Euler angle approach but now the rotation matrix is replaced by quaternions. The $\otimes$ symbol represents the Kronecker product [19]. In MATLAB one can use *quatmultiply* in order to perform the Kronecker product on two quaternions. The conjugate of a quaternion is denoted by $\mathbf{q}^*$.

$$\dot{d} = v \tag{3-11}$$

$$\dot{v} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \mathbf{q} \otimes \frac{1}{m} \begin{bmatrix} 0 \\ 0 \\ \sum f_i \end{bmatrix} \otimes \mathbf{q}^* \tag{3-12}$$

The rotational EOM using quaternions is different, although similarities can be seen. Equation (3-6) can still be used but Equation (3-13) replaces Equation (3-5). At any time, the Euler angles can be calculated from the quaternion using Equation (3-14) or by using the MATLAB function *quat2eul*. Although quaternions are not as intuitive to use as the Euler angle approach, quaternions do not have any singularities and can be used for any attitude. As this thesis project will focus on recovery of quadrotors, it is very likely that more extreme angles will occur. Therefore the quaternion approach is the preferred one for this thesis project.

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega \otimes \mathbf{q} \tag{3-13}$$

$$\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix} = \begin{bmatrix}atan2(2(q_0q_1 + q_2q_3), q_0^2 - q_1^2 - q_2^2 + q_3^2)\\ asin(2(q_0q_2 - q_3q_1))\\ atan2(2(q_0q_3 + q_1q_2), q_0^2 + q_1^2 - q_2^2 - q_3^2)\end{bmatrix} \tag{3-14}$$

3-1-3 Aerodynamics

So far aerodynamics were left out in the EOM. As one can imagine, aerodynamics play a big role in an aerial vehicle like a quadrotor. There are two main parts that will be discussed: the aerodynamics concerning the rotors and the body aerodynamics. Rotor aerodynamics have been extensively studied in the past century with the development of the helicopter [15]. The lift a rotor produces can be modelled in a simple way as shown in Equation (3-15). For most applications this equation is accurate enough, but one should be aware that more complex aerodynamic effects are taking place, certainly in highly dynamic manoeuvres. If the vehicle has some airspeed, the rotors will behave differently compared to the hover condition. The extra airspeed can create a different angle of attack on the rotors, causing them to behave differently. Rotors are not infinitely stiff either, so a difference in thrust on one side compared to the other can create an effect called rotor flapping [15] or thrust variation [21]. The assumption is made that the air is in the same state everywhere, but in reality this is not the case. A rotor leaves turbulent air behind and if it flies through that again, the resultant lift will change. This is probably hard to model as it does not only depend on the current state, but also on the time-history of the states. One could try and analytically determine all these effects, but alternatively one can perform tests and map the c_f coefficient used in Equation (3-15) over a range of states. In order to do so one needs sufficient test data, which might not always be available.

$$f_i = c_f\omega^2 \tag{3-15}$$

Body aerodynamics are mostly just drag components. If the quadrotor moves through the air, some drag will be present. In most literature only yaw damping is considered. In M. Mueller, R. D'Andrea the body drag is simply modelled as shown in Equation (3-16) where γ is a constant that is determined from test data [12]. In this paper the yaw drag was added to make sure the quadrotor would find a steady-state yaw rate. One can add drag in the same fashion for pitch rate and roll rate. In [22] pitch and roll drag is added to the equation and the drag is not linear but quadratic with respect to rotational velocity. The resulting equation can be seen in Equation (3-17).

$$\tau_{aeroSimple} = (0, 0, -\gamma r) \tag{3-16}$$

$$\tau_{aeroQuadratic} = - \parallel \omega \parallel K_d\omega \tag{3-17}$$

For some applications a more detailed aerodynamic model is required. There are several options to model this. One could use a live Computational Fluid Dynamics (CFD) simulation of the quadrotor in order to find al aerodynamic forces and moments on the vehicle. This would be very

computationally intensive, and would therefore be extremely slow. Another option is to perform some CFD simulations and make a mapping of the aerodynamic forces and moments depending on a wide range of states. This might be a viable option, depending on the amount of CFD simulations that will be needed to cover all relevant states. Instead of performing CFD simulations, one could also perform the tests in real life and map the forces and moments. The estimation of the forces and moments might be hard to perform as well as getting the quadrotor in the required states.

In order to use the gathered data from either CFD simulations or real life testing, one has to use some sort of an interpolation algorithm such that the aerodynamic model is continuous. A lot of research has been performed in this field. In [23] a solution using multivariate splines is proposed while in [24] a neural network is used to create the aerodynamic model.

3-2 Nominal quadrotor control

In order to fully understand the dynamics and controls of a damaged quadrotor one first has to understand how a fully functional quadrotor works. Therefore a literature review on nominal quadrotor control will be conducted in this section. Although many literature sources are available concerning quadrotor control, most of them are based on the same concepts. First a general overview of how a quadrotor can be controlled will be presented in Section 3-2-1. Secondly the different concepts in position control will be dealt with in Section 3-2-2. Next the attitude control will be presented in Section 3-2-3. To finish this section some info about state estimation for the quadrotor will be given in Section 3-2-4.

3-2-1 Basic Control

In this section the basic principles in quadrotor control will be explained. The quadrotor has 4 rotors which can be controlled individually. Each rotor is set up to rotate only one direction, either Clockwise (CW) or Counterclockwise (CCW), which allows for yaw control. In order to control the attitude of the vehicle, moments can be created by distributing the force in a specific way. From Equation (3-1) one can see how moments can be created around all axes. For example; if one increases the lift force on rotors 1 and 4 the roll angle will be controlled. The sum of the forces ($\sum f_i$) will cause an acceleration of the vehicle in the direction the rotors are pointing. If the quadrotor is level, this will result in an altitude acceleration. If the quadrotor is at a certain angle, it will also cause some acceleration laterally. Important to note that rotors can saturate, meaning either they cannot spin faster or the maximum lift force has been reached. This can become relevant in extreme manoeuvres.

3-2-2 Position Control

Position control is an essential part in the control systems of a quadrotor. It is usually the first controller in the whole system as it will come up with targets for the following subsystems. In most literature the position controller will output a pitch and roll target, either directly or by first calculating the required accelerations in each direction [25] [26] [27]. Altitude control is done in a different way, as the total force upwards (inertial) is important. A more complex option is the use of predictive control in which a trajectory is generated [28] and followed. Altitude control is then incorporated in the trajectory. The first case where the position controller comes up with pitch and roll targets will be elaborated on. A PID controller compares the desired lateral position with the actual lateral position. The output of this controller is a desired lateral acceleration. The desired acceleration is then converted to a pitch and roll angle using the yaw angle as well, which can be seen in Equation (3-18) and Equation (3-19) [25]. Some literature calls this the backstepping method. The position controller is visualised in Figure 3-3.

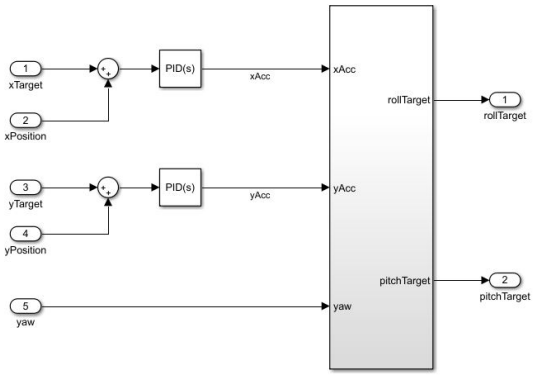

Figure 3-3: Position Control

$$\phi_{des} = \frac{1}{g}(\ddot{x}_{des} \cdot sin(\psi) - \ddot{y}_{des} \cdot cos(\psi)) \tag{3-18}$$

$$\theta_{des} = \frac{1}{g}(\ddot{x}_{des} \cdot cos(\psi) + \ddot{y}_{des} \cdot sin(\psi)) \tag{3-19}$$

Altitude control is done separately as mentioned before. Usually a PID controller is used similar to the one in the lateral position control but instead of converting it to a pitch or roll angle, it is converted into a total force. This is done using a simple Non-linear Dynamic Inversion (NDI) described in Equation (3-20). In this way the PID is still controlling a linear system. This altitude controller only works in situations where pitch and roll angle are not extreme. If one of the two angles becomes too big, the resultant required total force will cause the rotors to saturate. This equation also has singularities when roll and pitch are $-\frac{\pi}{2}$ or $\frac{\pi}{2}$. Note that the inertial z-axis is pointing downwards, so the total force in hover will be negative.

$$\sum f_i = \frac{m(\ddot{z}_{des} - g)}{cos(\theta)cos(\phi)} \tag{3-20}$$

3-2-3 Attitude Control

While the position controller is considered as a slower outer loop, the attitude controller is the fast inner loop. The goal of the attitude controller is to track the desired roll, pitch and yaw angle. In literature various different concepts are proposed with varying complexity.

The simplest method is having a single PID or cascaded PID that compares the current attitude with the desired attitude [27]. The outputs are the moments required around each axis. By using Equation (3-1) and the requirement that the sum of all forces should be the target value coming from the altitude controller, one can calculate the required force on each rotor. Using the aerodynamic model, the rotor speed can be found. The control scheme is visualised in Figure 3-4. In extreme conditions where rotational velocities are high, this method will likely fail, or perform badly. This due to the non-linear nature of the rotational EOM. In Equation (3-21) the rotational EOM has been written out. Clearly the control of the pitch, roll and yaw attitude is not only affected by the moment applied but also by the gyroscopic precession. For example, in Mellinger [25] and Dong et al. [26] this is taken into account so that the system becomes an NDI controller.

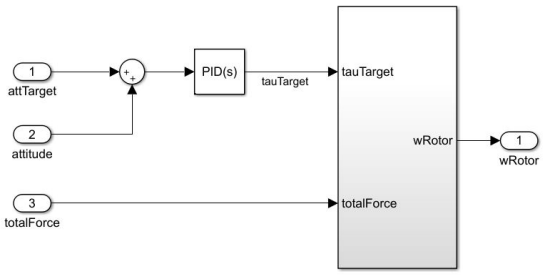

Figure 3-4: PID Attitude Control

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{qr(I_y - I_z) + \tau_x}{I_x} \\ \frac{pr(I_z - I_x) + \tau_y}{I_y} \\ \frac{pq(I_x - I_y) + \tau_z}{I_z} \end{bmatrix} \tag{3-21}$$

The simple PID controller method with or without NDI part does assume that the aerodynamic properties of the rotors are relatively constant. If this is not the case, due to disturbances or an inaccurate model in the NDI, then the overall performance will decrease as the system will seem more non-linear for the controller.

A solution to this problem is proposed in Smeur et al. [7] where initially an Incremental Non-linear Dynamic Inversion (INDI) controller is used. An INDI controller is less model dependent and is thus more robust to model inaccuracies. Instead of directly using the model to determine the angular acceleration in a certain state to get the rotor output, an incremental change in output is made based on the relative change in state and desired state. In this way disturbances and other unmodeled effects are compensated [7]. A drawback of the INDI approach is that it requires derivatives of certain states, which can lead to noisy signals. Appropriate filtering is therefore needed and is one of the challenges when implementing an INDI.

The core of an INDI controller is the control effectiveness matrix as it is used to determine the incremental change in output. Although the INDI method is very robust, it requires a correct control effectiveness matrix. For the quadrotor this would mean that one needs to map the behaviour of the rotors as well as possible, which goes against the reason why an INDI controller is used in the first place. Instead one can adaptively determine the control effectiveness matrix during flight. This is called an Adaptive Incremental Non-linear Dynamic Inversion (AINDI) controller [7].

In an AINDI controller the expected accelerations are compared to the measured accelerations and used to adapt the control effectiveness matrix. One could use a Least Mean Squares (LMS) filter to adapt the matrix. An example of this is shown in Equation (3-22) [7] where G is the control effectiveness matrix and μ_1 and μ_2 are diagonal matrices that are used to set the adaptation constant for specific inputs or dimensions. One can see that the expected rotational acceleration ($G(k-1)\begin{bmatrix} \triangle w \\ \triangle \dot{w} \end{bmatrix}$) is compared to the measured change in rotational acceleration. Other filters can be used as well in order to adapt the control effectiveness matrix. When designing such a filter it is important to decide whether a finite or infinite horizon will be used. Both have advantages and disadvantages. When using a finite horizon filter, the filter might 'forget' essential data during hover, as the system is not being excited. But when using a infinite horizon filter, the adaptation to sudden changes might be slower. Imagine a sudden rotor failure, past data will not help and quick changes to the matrix have to be made.

$$G(k) = G(k-1) - \mu_2(G(k-1)\begin{bmatrix} \triangle w \\ \triangle \dot{w} \end{bmatrix} - \triangle\dot{\Omega})\begin{bmatrix} \triangle w \\ \triangle \dot{w} \end{bmatrix}^T)\mu_1 \tag{3-22}$$

3-2-4 State Estimation

So far many control problems have been posed and different solutions have been found. In the previous sections the assumption was made that the states are accurate and precise. In reality however, this is not necessarily the case. Depending on the environment, certain states will be hard to measure well and need some sort of estimator to get decent values. Indoor test facilities are usually equipped with an optical tracking system that is able to determine position, velocity, attitude and rotational velocity of the quadrotor very accurately. The acceleration and angular acceleration can still be obtained from an Inertial Measurement Unit (IMU) onboard but could in theory also be derived from the optical tracking system.

In an outdoor test environment, an optical tracking system is usually not available as the test area is not bounded. This means that the quadrotor needs a different way of estimating the aforementioned states. Most important is the estimation of the attitude and rotational velocity, as these are very important for the basic functionality of the quadrotor. Assuming the quadrotor has an IMU and a magnetometer onboard which can measure the earth magnetic field, a Kalman filter can be made that uses these sensors to estimate these states [15]. The gyro's in the IMU are used for the state-ahead prediction while the magnetometer and accelerometers are used as measurements of the attitude. In this way the rotational velocity can be estimated and the attitude estimation will be more robust and possibly quicker. Note that the attitude estimation from the accelerometers need some extra processing. The same principle is used for the estimation of position. The gyro's and accelerometers from the IMU will be used as well as the estimated attitude. However, there is still need for a measurement of position somewhere. Most commonly Global Positioning System (GPS) is used, but this only gives position updates every 10 Hz [15]. So a Kalman filter is definitely needed to speed up the frequency at which the position is determined. With the Kalman filter one can estimate the velocity as well. More complex systems do exist which use vision in order to determine position and/or velocity. The use of optical flow is a good example of that. Altitude can theoretically be obtained from GPS but most likely another system will be needed for this. Depending on the application, one can choose for laser-ranging, infrared or acoustic sensors for low altitude, and barometers for higher altitudes.

3-3 Fault Tolerant Control

In this section a literature review on FTC is performed. As the name suggests, this field of study looks into control systems that are able to control the quadrotor in case of a failure. FTC for quadrotors is quite new and still in its early days. Currently a lot of research is being done, but not all issues are solved yet. In the introduction and research motivation it was stated that there is a high demand to make quadrotors safer. One way to make quadrotors safer is to have a working FTC in which the quadrotor can at least land in a safe manner if a failure occurs. Faults or damage can occur due to various reasons. An electrical problem with one of the rotor motors can cause the motor not to work anymore. Overheating of a motor is a common failure as well. As the rotors are spinning at high speed, a collision with a wall or some object can cause the rotors to take damage. This may mean that the rotor is completely broken, but it might also be partially broken (eg. one side of the propeller breaks off). A system that can cope with these issues would enhance the safety of quadrotors significantly.

This thesis will focus on a single rotor failure, meaning one rotor does not provide any lift force anymore. In this case the solution is not trivial. Without loss of generality it is assumed that rotor 1 failed. Looking at Equation (3-1) one can see that if $f_1 = 0$ the solution to hover ($\tau_{xyz} = 0$) is the null solution, meaning all rotors have to stop generating lift. This is not a useful solution as this would mean the quadrotor would fall out of the sky. The sum of the forces should still meet the altitude controller demands. Note that $f_i >= 0$ as the lift cannot be inverted. Clearly a quadrotor needs all rotors in order to have full control over its attitude. However, this does not mean a quadrotor is completely uncontrollable if a rotor failure occurs.

Mueller and D'Andrea [12] conducted research if there are analytical solutions for a damaged quadrotor in hover. They consider one, two and three propeller failures and show that analytical solutions exist, although the definition of hover had to be revised. The solution is to give up on yaw control. In this way both roll and pitch can still be controlled. This also means the position can be controlled. Using this solution a quadrotor can stay on a certain position (hover) and still go to a certain position despite the quadrotor spinning in yaw.

Two types of FTC exist; Active Fault Tolerant Control (AFTC) and Passive Fault Tolerant Control (PFTC) which are presented in Section 3-3-1 and Section 3-3-2 respectively.

3-3-1 Active Fault Tolerant Control

In AFTC the control laws are reconfigured when a fault or problem is detected [29]. The system actively has to check if there are faults and to act accordingly. Currently, most literature in FTC for quadrotors use a AFTC system. The field of study is still young and most papers even assume the fault is a given. In this section the AFTC solution from literature will be presented. Mainly the differences compared to the nominal control presented in Section 3-2 will be highlighted.

As yaw control will be given up and the quadrotor will start spinning around its Z_b, some concepts from the nominal control systems will have to be revised or be made more abstract. In Sun et al. [1] the concept of a primary axis is introduced. The primary axis of a quadrotor is a vector with coordinates $[0, 0, -1]$ in the body frame, so it is 'pointing up'. Using this vector, position control can be done independently of yaw angle. This also makes the attitude control a bit more abstract, as no pitch or roll angle targets are given directly, but instead a vector. Position control works in a similar way as in the nominal quadrotor. PID controllers come up with acceleration targets which are used to calculate the desired primary axis. The calculation can be seen in Equation (3-23). Note that g is the gravitation vector. n_{des} is defined in the inertial frame and is normalised. This method is used in various papers [12] [8] [27].

$$n_{des} = \frac{a_{des} - g}{\parallel a_{des} - g \parallel} \tag{3-23}$$

The primary axis is the output of the position controller. As the desired primary axis is in the inertial frame, it has to be converted to the body frame. This can be done using the rotation matrix described in Equation (3-4). The goal of the attitude controller is now to follow this primary axis target as well as possible. This is done in various ways across all literature. In Lippiello et al. [27] the desired primary axis is converted to a pitch or roll angle target depending on which rotor failed. The body axis system is different in this paper due to which roll and pitch are defined differently as well. Using the pitch or roll target a PID controller is used to determine the required torque. The opposing rotor will thus not be used either in this system.

A different concept is to come up with a desired pitch and roll rate from the desired primary axis. This is done in [1] and [8]. Using an NDI, the desired pitch and roll rate can be determined. The outcome can be seen in Equation (3-24) where the virtual input ν_{out} is defined in Equation (3-25). $n_{des} = [h_1, h_2, h_3]$ is used to simplify some notations. $\hat{n}^B_{des}$ denotes the desired primary axis x and y components in the body frame. The virtual input is basically a proportional controller that compares the current primary axis and the primary axis target. Usually the x and y components of the current primary axis are zero. Once the target pitch and roll rates are determined a simple PID controller can be used to determine the desired pitch and roll accelerations. These can then be fed into an INDI or AINDI can be used to determine the target rotor speeds.

$$\begin{bmatrix} p_{des} \\ q_{des} \end{bmatrix} = \begin{bmatrix} 0 & 1/h_3 \\ -1/h_3 & 0 \end{bmatrix} \left(\nu_{out} - \begin{bmatrix} h_2 \\ -h_1 \end{bmatrix} r - \hat{n}^B_{des} \right) \tag{3-24}$$

$$\nu_{out} = \begin{bmatrix} k_x(n_x^B - h_1) \\ k_y(n_y^B - h_2) \end{bmatrix} \tag{3-25}$$

As mentioned before, AFTC needs to know if there is a failure in order to adapt the control strategy. The controller will then go into primary axis control mode and the INDI or AINDI will not use the failed rotor anymore. This principle is called Fault Detection and Isolation (FDI). Little literature treats the fault detection, but in Lu and Van Kampen [8] and Zhang et al. [30] some research has been conducted into fault detection. Similarly as the AINDI the control effectiveness of each rotor is estimated. If the control effectiveness deviates too much from the expected aerodynamic model, it can be assumed that the rotor has failed. The biggest challenge in doing this is the trade-off between how quickly one can detect a failure and not detect failures when there are none. Therefore it is important to select a suitable filter.

3-3-2 Passive Fault Tolerant Control

In PFTC the control systems are robust to faults or problems. This means the same control laws will be used for nominal control as when a fault occurs. Little literature about PFTC for quadrotors can be found as currently AFTC is already challenging enough. The advantage of having a PFTC is that there is no need to detect the fault, as the system itself is robust enough that it can handle faults. Ideally this is the preferred method, however in practice this might not be easy to achieve. A PFTC has to work in all situations, the increased flexibility usually means that the peak performance in a certain situation will decrease. In this section possible PFTC concepts will be presented.

In Section 3-3-1 the primary axis was introduced for position control. This concept can also be used in nominal quadrotor control. This would mean that position control is a PFTC and thus the control law does not have to be changed if a fault occurs. The NDI part of the attitude controller used in AFTC can also be used in nominal quadrotor control. This gives the pitch and roll rate command. For nominal control, a yaw controller is also needed that comes up with the desired yaw rate. The next step of the control system is where the problems start to occur. An AINDI will try to match pitch, roll and yaw rate commands, however when a rotor fails, this is not possible to do anymore. In order to make this part PFTC a new strategy will be needed. If one can find a way to prioritise roll and pitch and regard yaw control as a secondary tier target, a method could be developed that is PFTC. This could be a system that hedges the pitch, roll and yaw rate inputs into the final force distributor. Pseudo control hedging in FTC is described in [31]. A perfect force distributor for PFTC is the AINDI. The AINDI will estimate the control effectiveness of the rotors, so if a failure occurs, the control effectiveness should go to zero and the rotor will not be used anymore. With this configuration a PFTC for a quadrotor could possibly exist if one does not prioritise yaw control. However no example of this is present in literature.

3-4 Upset Recovery

Upset recovery is rarely considered in quadrotor research. This is due to various reasons. First of all, upset conditions are not well defined for quadrotors. As quadrotors are very agile and have a high power-to-weight ratio upset conditions do not seem to exist. Secondly, quadrotors do not rely on lifting surfaces that need vehicle airspeed and a certain angle of attack making them less vulnerable to complete loss of control. In Section 3-4-1 it will be shown that upset conditions for a quadrotor do exist. It will also be shown that it is likely that the quadrotor will enter an upset condition when damage takes place. A literature review concerning upset conditions for multirotors and trajectory generation will be conducted as well. In Section 3-4-2 a short review of the literature about upset recovery in the airplane industry will covered.

3-4-1 Upset recovery for quadrotors

As mentioned, upset condition for a quadrotor is not well defined. An upset condition does not mean a condition from which recovery is not possible, it indicates that a specific set of manoeuvres is required to get the vehicle back in full control. Using this knowledge, one can review again what an upset condition could be for a quadrotor. When the quadrotor is not damaged and in normal hover condition the attitude and altitude can be controlled all together. Clearly this is not an upset condition. Now consider a quadrotor in an upside down situation. In this case the altitude cannot be controlled. In fact some force will be needed to rotate the quadrotor meaning the quadrotor will be accelerated down even more. This can be seen as an upset condition. First the attitude has to be recovered before the altitude can be controlled again. A counter intuitive move has to be made (eg. accelerating down) in order to recover. Even when the quadrotor is at high pitch or roll angles, it might be beneficial to give up on altitude control for a moment and give more authority to the attitude controller. Otherwise a too big lateral displacement might occur. This also depends on the space the quadrotor can fly. If the quadrotor is very close to the ground, giving up altitude might not be an option. On the other hand, if the quadrotor is close to a wall there is no room for any lateral displacement in that direction. This is only considering a healthy quadrotor. If the quadrotor suffers from a rotor failure the flight envelope becomes smaller and more upset conditions will exist. Clearly upset conditions for a quadrotor do exist.

From the literature study one can conclude that little research into upset recovery for quadrotors or multirotors has been conducted so far. A very relevant paper for this subject is from Beffa et al. [32]. In this paper recovery for both state estimation and control is considered and is tested for extreme manoeuvres. As the state estimation is based on vision one can imagine this being very challenging. Regarding control this paper proposes a step-by-step recovery method in which attitude is recovered first, followed by altitude and position.

Literature with regards to trajectory determination and following is relevant for upset recovery as well. Plenty of research is being conducted with regards to this subject. From [33], [34] and [35] one can conclude that a big challenge in trajectory generation is to make it computationally efficient enough such that it can run on a real-time system. In Hehn and Andrea [35] a method is proposed in which a trajectory is generated to bring the quadrotor from any initial state to a target position in rest. This method can thus be used to recover the quadrotor from an upset condition as well. Generation of the trajectory is done in less than a tenth of a second. In Mueller et al. [33], [34] a similar method is proposed but with a different focus, for example catching a ball mid-air. All methods use some pre-defined model properties and limits, which do not necessarily correspond to the real limits of the vehicle. In this case they are probably on the safe side. However in case of damage the vehicle properties change significantly in which case these trajectory generation methods will fail. In order to use these methods, a modification will be needed in order to use it for damaged quadrotors. Ideally the limits can be set dynamically, although this will probably go at cost of computation time. External disturbances are also not considered in these methods.

Looking back to the current control systems of quadrotors presented in Section 3-2 and Section 3-3, they are likely not to recover from an upset condition. Looking at Equation (3-20) one can already see that the required total force at 90°pitch or roll will become infinite. Also at higher pitch or roll angles the required total force will grow and might saturate the propellers, causing the vehicle to lose attitude control. Looking at Equation (3-24) one can see that if the primary axis and desired primary axis are at 90°of each other, h_3 becomes zero and no meaningful solution will be found. Clearly, some of the systems will have to be made more robust and a strategy has to be developed in order to recover from an upset condition even in the undamaged case. With a damaged quadrotor the flight envelope is even smaller, so recovery will be even harder.

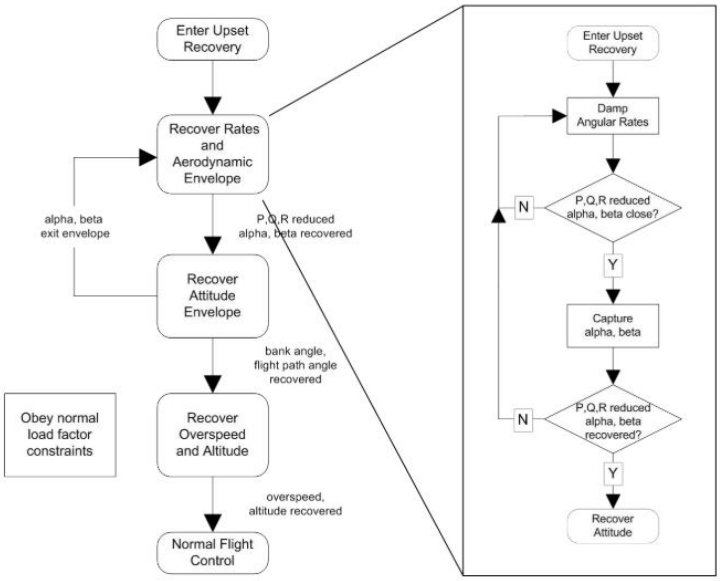

Figure 3-5: Upset recovery strategy [3]

3-4-2 Upset recovery for airplanes

For airplanes a lot of research has been conducted in this field. In [36] and [3] an analysis on upset recovery for airplanes is presented. Although an airplane is not the same as a quadrotor, similarities can be found in approach to the problem. Some of the upset conditions considered in the papers are: nose-up, nose-down, inverted flight, stall, deep stall, spin and high sideslip [36]. In order to recover form these situations, several strategies are tested. The strategies always consist of a set of manoeuvres that are performed in series. This usually means giving more authority to a certain controller and less to another. In Figure 3-5 a proposed upset recovery strategy can be seen. Clearly, the recovery of some states is more important than others.

Chapter 4

Methodology

In this chapter the methodology for the thesis project will be presented. The literature review is done and the research can start. In section Section 4-1 the plan of attack for the remaining part of the research will be explained. In section Section 4-2 the experimental set-up will shown. Finally in section Section 4-3 the risks of this thesis project will be presented.

4-1 Plan of attack

In this section the plan of attack will be explained. The author has conducted a literature survey and is up to date with the state-of-the-art developments in the field of FTC and upset recovery. The research question has been set-up so now the actual research can start to answer it.

In the literature review many concepts for different sub-systems were presented. From the literature study some conclusions can be made on what concepts are promising and which are not. The research will use the work from Sun et al. [1] as a baseline. The system proposed has been validated even in high speed conditions. Using this baseline system gives a solid basis to start from and conduct the research without the need of first having to develop a complete FTC controller from scratch. The author has access to the code used in Sun et al. which will speed up the process.

4-1-1 Controller approach

The controller can be subdivided into five parts namely; position controller, altitude controller, yaw controller, attitude controller and the INDI control allocator. This can also be seen in Figure 4-1. The intention is to keep this structure and adapt some parts, preferably in the outer loop (eg. position controller and altitude controller). Ideally, a fault detection subsystem is added as well, as the current system assumes the fault is known. As faults are artificially added it is possible to give the information to the control systems. However in practice when an actual fault occurs this will not be the case and proper fault detection is required.

Regarding position control, little changes will have to be made. The primary axis method can be used for the nominal and damaged case and is therefore an ideal candidate for the FTC. Currently, there are no limits applied to the orientation of the primary axis. In order to avoid pushing the quadrotor in an upset condition due to itself, smart limits can be applied to the primary axis. Without limits the primary axis could end up pointing level with the ground, requiring either 90

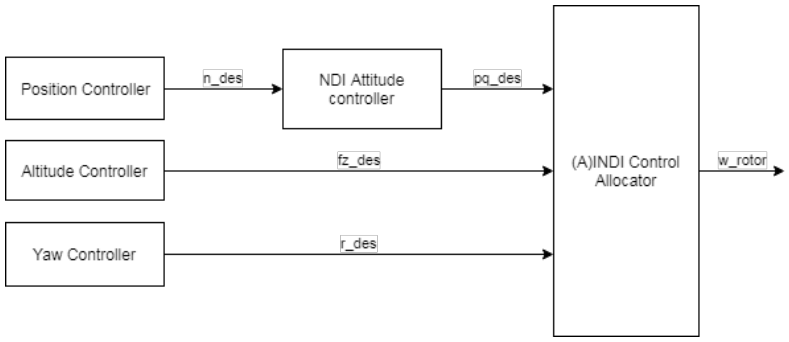

Figure 4-1: Control overview

°pitch or roll, which is an undesired condition. The position controller is a double PID controller in series. It is important that the integrals do no windup, so an anti-windup system might also be added.

The altitude controller will need some adaptations as currently it cannot handle upset conditions. From Equation (3-20) it can be seen that there are singularities. The current altitude controller works in nominal conditions, so only for extreme conditions it needs to be adapted. As proposed in [36] the altitude controller can be disabled when an upset condition is present, however one could make a more clever system. One could make a trade-off between lateral and altitude displacement, and scale the output of the altitude controller based on that. The trade-off parameter can then either be kept constant, or be varied if more info about the surroundings is known.

The normal yaw controller is only used when the quadrotor is in a healthy condition, so no effort will be put in this system. However, in the damaged case yaw rate can help the quadrotor to stabilise itself. This means that in order to recover properly, the quadrotor needs to have some yaw rate. One step of the recovery process could therefore be to increase the yaw rate as much as possible on purpose. Although yaw control is given up, it does not mean it should not be considered anymore in the research.

The attitude controller presented in Equation (3-24) has to change as currently it cannot handle all orientations of the quadrotor. The attitude controller will most likely be the core of the research because it will determine how the quadrotor will recover from an attitude perspective. Using the knowledge of the rotor failure, the attitude controller can made clever decisions on what angular rates are required. It is expected that most of the work done during the thesis project will be in the attitude controller, stating this with the current view.

The current INDI control allocator can be replaced by an AINDI in order to detect rotor damage and update the control effectiveness matrix. The use of an AINDI is not neccesary. If one knows what the damage on the quadrotor is and what the effect is on the control effectiveness, one can use the INDI. Implementing the AINDI could enhance robustness though. The FDI and AINDI system could work together. Feedback from the AINDI can be used to detect rotor failures. As presented in [8] the control effectiveness can be estimated. If this exceeds a certain threshold, the rotor is assumed to be damaged.

4-1-2 Code development

In this subsection the code development approach will be explained. The author has performed a literature survey and is now up to date with the latest developments in the field of FTC and upset recovery. New ideas and concepts for the controller are presented above. At some point this has to be translated into actual control systems in the form of code.

The bebop quadrotor (see Section 4-2-4) that will be used needs control systems in the form of C code. Different frameworks exist that can handle the basic functionality of a quadrotor. Two examples of this are the Paparazzi UAV [37] framework and the PX4 [38] framework. Using the framework allows the author to focus more on the specific parts of the control systems that are related to this thesis project. In the framework specific subsystems can be altered while keeping the other subsystems. For this thesis project the PX4 framework will most likely be used due to its flexibility. The control system will be developed in a structure that allows to implement new subsystems in the PX4 framework.

The control system will be developed using Simulink from MATLAB. In Simulink a framework will be setup in which the control systems can be developed and tested. In the framework a quadrotor simulator is present which replicates the real quadrotor as accurate as possible. The sensor inputs into the control systems are simulated as well. This enables the author to develop control systems quickly without having to use actual hardware. Using a virtual simulator can boost the development significantly. One can disable translational movement and only allow rotational movement in order to test the attitude controller separately. Once this works, one can move on to translational movement etc.. Using an isolated situation it is often easier to see the issues and solve them. This would be very hard to do in a real environment.

In order to deploy the code onto the quadrotor, the code will be built from Simulink. Not the entire control block will be built, but the built will be split up in a few different subsystems. These subsystems can then be implemented in the PX4 framework. A Simulink code wrapper will be needed that can call the automatically built code and manage the interface between the code and the PX4 framework. It is important that in the Simulink model all the interfaces are implemented the same as they will be in the PX4 framework. Otherwise the controller will not work the same in real life as it does in the virtual environment. In Figure 4-2 a very basic visual representation is given. Three arbitrary subsystems are used to show how they live in both Simulink and PX4.

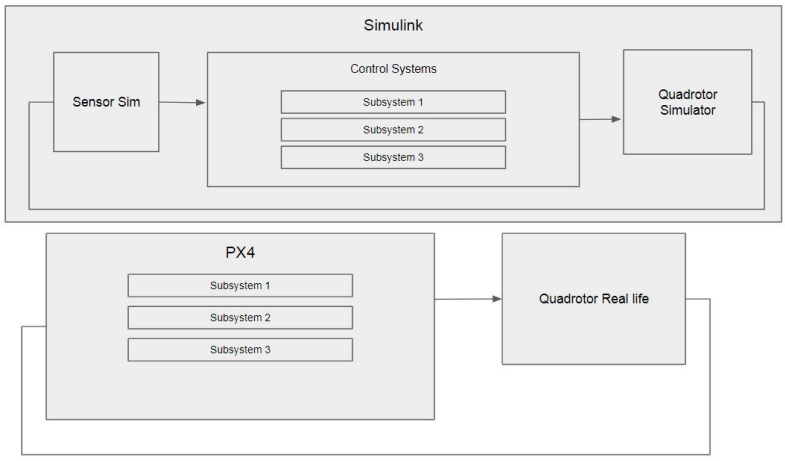

Figure 4-2: Code environment overview

In a big project where people have to collaborate, a source control system is needed to organise and manage the code. All the code in this thesis project will be source controlled using the *git* protocol. Using this method allows the users to track code progress and to develop different concepts without losing other progress. *git* allows users to collaborate in an efficient way. It manages differences in code between two users so that two users can work on the same subject at the same time. In *git* one can make use of branches in order to control the work that is being done. A convention that can be used is proposed by Driessen [4] and can be seen in Figure 4-3. The master branch is where the latest release of a code lives. During development, everyone works on the develop branch from which users can make feature branches in order to add new features or

try other ideas. One can at any time switch between branches. Every line in the figure represents a branch while every dot represents a commit. A commit is a package of work the user sends to *git*. Once the development of a feature is done, it can be merged into the development branch.

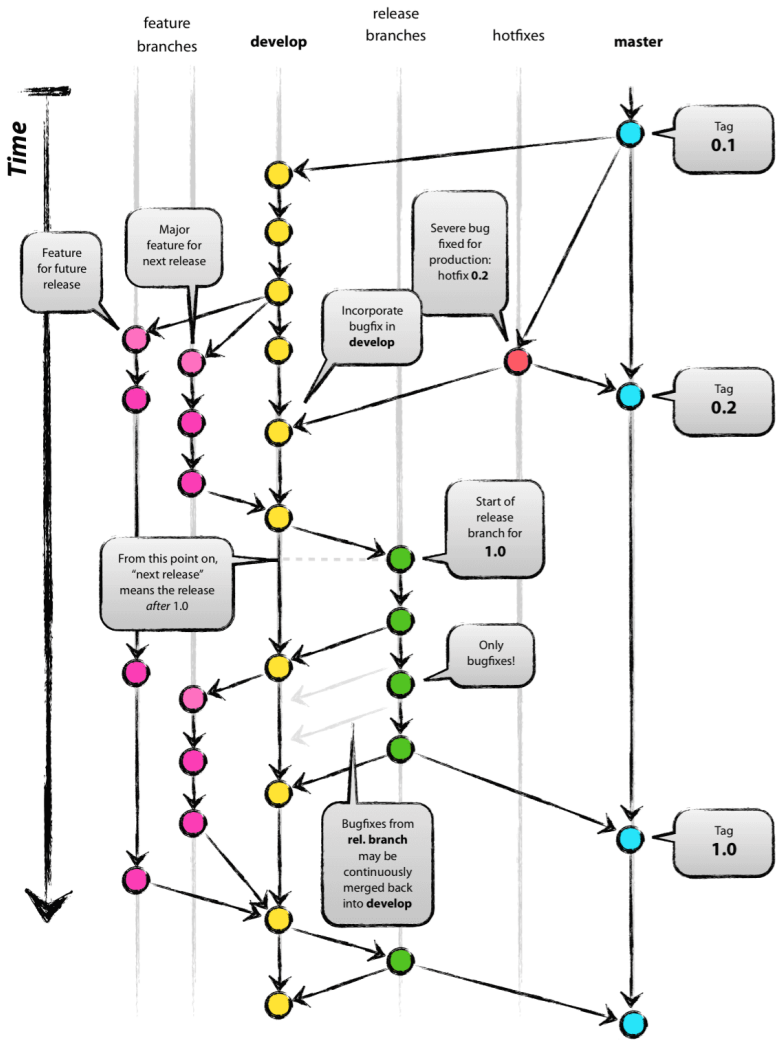

Figure 4-3: Git branching model [4]

4-1-3 Verification, validation and testing

In this subsection the verification, validation and testing approach will be explained. In the previous sections it was shown how the controller will be developed and how the code will be structured. It was also mentioned that the control systems will be developed with help of a virtual environment created in Simulink. This is an ideal place to do the first verification. With verification one will check if the design is doing what it is intended to do. As the virtual environment is very flexible, every part of the developed control system can be verified. Once the code is deployed on the bebop, another verification step needs to take place. Using basic inputs-output tests one can check that the deployed code is behaving in the same way as in the virtual environment. If this is the case, then the deployed code will also do what is intended.

When developing, many tests will be performed in the virtual environment. This is the first step of the validation process. With validation one will check if the intended design solves the problem

as expected. If a recovery strategy does not work in the virtual environment, it is validated to a certain extend that it does not work. Of course, in validation it is important that all conditions are representative enough in order to make a correct conclusion. If a recovery strategy works in the virtual environment, it does not mean it will work in real life necessarily. After testing in the virtual environment, testing in a real environment with a real quadrotor will take place. This will be done in the cyberzoo at TU Delft. More info about this in Section 4-2-2. The controlled environment is the perfect place to test the developed control system. During testing, problems will occur and the control systems will need to be updated. This will then be tested in the virtual environment again before testing in the cyberzoo. Once the results in the cyberzoo are satisfactory, testing in the wind tunnel will take place. More info about this in Section 4-2-3. In the wind tunnel more realistic wind or vehicle airspeed can be simulated, making it more realistic. For this project, this will likely be the final step of validation.

4-1-4 Work Flow Diagram

A work flow diagram can be seen in Figure 4-4. Important to note that there is an iterative structure in place. Developments are always made in the Simulink model which are then tested in the virtual environment before going to the cyberzoo and the wind tunnel. Using the test data, the controller can be improved and another development iteration can start.

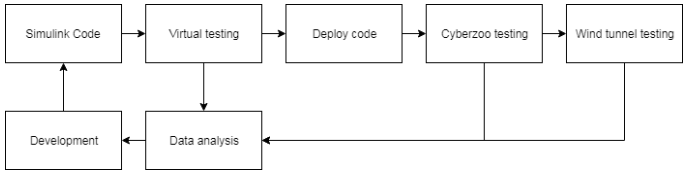

Figure 4-4: Work Flow Diagram

4-2 Experimental Set-up

The experimental setup can be split up into four parts; the simulator, cyberzoo, wind tunnel and the bebop quadrotor. For each element some info will be given as well as the limitations of the system.

4-2-1 Simulator

A simulator will be developed in which control systems can be developed very easily. The simulator will be made using MATLAB Simulink which is an excellent tool for this purpose. This tool also allows to compile the control systems code to deployable code onto the quadcopter in the future. Using the simulator, the control systems can be tested in ways that are hard to test in real life. for example, any initial condition can be given. In the simulator one can disable certain degrees of freedom such that only attitude control can be tested for example.

A disadvantage of using a simulator is that it might not resemble real life good enough. The dynamics of the system are most likely to be different to a certain degree. Also the sensor readings and sensor quality can be different in practise. A control system that works in the simulator might therefore not necessarily work in real life.

4-2-2 Cyberzoo

At the Aerospace faculty of TU Delft one can make use of the cyberzoo to test UAVs. The cyberzoo is an aera where UAVs can be tested in a safe manner as the whole area is surrounded by a net. The cyberzoo also features an *OptiTrack* system that is able to track the position and attitude of the vehicle at any time.

The cyberzoo has its limitations. The size is limited and is thus not suitable for high speed tests. This could also have consequences for the upset recovery, as the maximum altitude achievable in the cyberzoo might not be enough. During testing, problems can occur meaning the quadcopter can possibly crash. A crash in the cyberzoo can lead to damage to the quadcopter making it a non-ideal development tool. The *OptiTrack* system is a very accurate measuring system, but in real life where one has to use GPS or other systems the position and attitude data will not be as accurate. The data in the cyberzoo is therefore not realistic for applications where one wants to fly outdoors.

4-2-3 Wind tunnel

At the Aerospace faculty of TU Delft a wind tunnel (Open Jet Facility) is available for testing during a select amount of time. In the wind tunnel the quadcopter can be tested in high speed conditions (+10m/s). The wind tunnel also features the same *OptiTrack* system as the cyberzoo making transisiton from cyberzoo to wind tunnel more easy. Wind tunnel tests will show how big the aerodynamic effects are and how good or bad these were modelled and taken into account in the control systems.

The wind tunnel is also limited in size, this is specifically important during the high speed tests as a small instability will cause the quadcopter to deviate from its position very rapidly. The wind tunnel will only be available a couple of days for this project which can be vital for the results and outcome of this project. If the wind tunnel tests are unsuccessful for some reason, it is unlikely that an extra test session can be planned in.

4-2-4 Bebop Quadrotor

The quadcopter that will be used is the *Parrot Bebop 2 Drone* which is an off-the-shelve quadcopter. However some edits are made to the drone to optimise it for this problem. The camera is taken out to make the quadcopter lighter, resulting in a quadcopter that only weighs 0.41kg [1]. Four grey markers are added onto the drone such that the *OptiTrack* is able to track the vehicle while it is flying. The off-the-shelve software can be overwritten by self developed software making it a suitable drone for this project. Propellers can also be replaced such that damage can be simulated for example.

Although the problem is made as general as possible, by only using this quadcopter the thesis project will be biased towards this quadcopter only. It is not guaranteed that the outcome of this thesis will also apply to other quadcopters. The Electronic Control Unit (ECU) of the drone has processing limits meaning one has to write code efficiently and the code cannot be expanded infinitely. Trajectory generating is a computationally intensive project and might not be possible to perform in real time on board.

4-3 Risks

Every project has risks associated with it. Risks are defined as things that can go wrong and have an impact on the project. Some risks can be mitigated but others might not. It is important to

have a good view on the project risks in advance. Below some of the risks will be enumerated with impact, description and how they could be mitigated. The risks are ordered arbitrarily. First some practical risks are presented.

- **Description:** If a computer or laptop with all the code on breaks down, or gets lost all progress can be lost.
  Impact: Code and data lost.
  Mitigate: All code is on a remote server and source controlled using *git*. Actual data needs to be saved on a remote server as well such that it can be accessed from several computers.
- **Description:** No working quadrotors available.
  Impact: No testing is possible.
  Mitigate: By testing in a virtual environment it is less likely to break quadrotors. Having a clear planning when people are using certain drones also decreases the chance of this risk.
- **Description:** No test opportunities in the cyberzoo. The cyberzoo might be too busy or not available.
  Impact: No real life testing in controlled environment.
  Mitigate: Apart from using the wind tunnel to test, there is no mitigation for this problem.
- **Description:** No wind tunnel test opportunities. There are only a select amount of test days in the wind tunnel for this thesis project.
  Impact: No validation in high speed conditions.
  Mitigate: One can mitigate this by preparing well for the wind tunnel tests to reduce the chance of this risk. There will be two separate test weeks in the wind tunnel with some time in between such that problems that occurred during the first test week can be solved.
- **Description:** Illness or other unexpected situations causing the author not to be able to continue working.
  Impact: No work can be performed.
  Mitigate: Making sure all work is done in time such that unexpected events at a crucial do not have a big impact.
- **Description:** Code deployment on the quadrotor fails.
  Impact: The developed control systems cannot be tested on the quadrotor.
  Mitigate: By checking if other people have done it before gives an indication if it is possible or not.
- **Description:** The PX4 framework does not work on the quadrotor.
  Impact: The PX4 framework cannot be used.
  Mitigate: Alternatives to the PX4 framework exist. It is important to check beforehand if the framework works for the selected quadrotor. This information should be available.

Secondly, risks regarding the research itself are presented.

- **Description:** The current control systems are able to recover the quadrotor from an upset condition. There is no problem to solve.
  Impact: The research becomes less useful.
  Mitigate: By doing the literature study, it should become clear if this is the case or not.
- **Description:** No upset recovery strategy can be found for the damaged quadrotor.
  Impact: The research question cannot be answered positively. The goal of the thesis project is not met.
  Mitigate: The literature study should give an indication how hard the problem is, reducing the chance of this risk. If it is not possible to recover however, answering the sub-questions posed still make the research useful.

- **Description:** The research takes too long. Either the problem is too hard, or too much work is required in certain areas.
  Impact: The thesis project is not done in the time span. The research question is not answered.
  Mitigate: Focus should remain on the research itself. Time should not be wasted on elements that are not essential to the research. Go for the easy solution if it is not essential for the research.

- **Description:** Virtual environment does not represent reality.
  Impact: All progress using the virtual environment might be useless.
  Mitigate: Know the limits of the virtual environment. Check with other papers how the correlation to reality is and what to look out for. Plan a simple test as soon as possible in order to validate the virtual reality.

Chapter 5

Thesis Plan

In this chapter the thesis plan is presented. In short the thesis focus and goals will be discussed in Section 5-1. The planning of the thesis project is presented in Section 5-2.

5-1 Thesis Focus and Goals

In this section the thesis focus and goals of the thesis project will be briefly explained. As mentioned before, the focus of this thesis project will be on the upset recovery of a damaged quadrotor. The focus will be on a rotor failure as form of damage. Both single and double rotor failures will be considered. From the literature review it seems that for three rotor failures only an analytical solution is available. The focus is on recovery of the quadrotor, this means that preferably the inner loop controls are kept the same as presented in literature. The focus should be on the setpoints towards the inner loop controls instead.

The goal of the thesis project is to manage to recover a damaged quadrotor from any condition. This because when damage occurs, the transition will cause the quadrotor to be in an upset condition from which it is hard to recover. The author is expected to develop such a system and deploy it on an actual quadrotor. The author is expected to perform tests in the cyberzoo as well as in the wind tunnel. From a performance point of view, the goal is to create a recovery strategy that works in both nominal case as the damaged case. The recovery strategy should be robust in the sense that it should be able to recover the quadrotor at all times.

5-2 Planning

In this section the planning for the thesis project will be presented. Different activities will be presented in Section 5-2-1 which are then visually presented in Section 5-2-2 using a Gantt chart. By making a planning, the author can see which deadlines are critical for the project. From the planning it will become clear how much time has to be invested in each part of the project. It is easy to get stuck on some part causing the project to be delayed.

5-2-1 Activity Plan

1. **Upset recovery development**
   An upset recovery strategy has to be developed in the virtual environment. First for a quadrotor without damage, then for a damaged quadrotor.

   1.1. Upset recovery for healthy quadrotor — 1 week

   1.2. Upset recovery for damaged quadrotor — 4 weeks

2. **Code deployment** — 1 week
   When the upset recovery has been developed, the code has to be compiled and deployed on the quadrotor. A lot of work for this has already been done by a colleague.

3. **Cyberzoo testing**
   When code is deployed on the quadrotor, testing in the cyberzoo can take place. First the basic functionality has to be tested. Checking that the quadrotor can fly and perform basic manoeuvres. Now the upset recovery of a healthy quadrotor can be tested. This can be done by inducing a high roll angle which can be done by a feed forward action. The quadrotor then has to recover from this condition. Next the FTC has to be tested. Giving some yaw rate before disabling one of the rotors can help the quadrotor not entering an upset condition. Once this is successful, the upset recovery for the damaged quadrotor has to be tested. This can be done by disabling one of the rotors and notifying the quadrotor with a small delay.

   3.1. Test basic quadrotor functionality — 1 week

   3.2. Test upset recovery for healthy quadrotor — 1 week

   3.3. Test damaged quadrotor hover — 1 week

   3.4. Test upset recovery for damaged quadrotor — 1 week

4. **Wind tunnel testing**
   The wind tunnel has been booked from 28 Jan to 1 Feb, and 18 Feb to 22 Feb. During this time various tests will take place from various people in the group. Some of the features developed in this thesis project will be tested in week 2.

   4.1. Test week 1 — 1 week

   4.2. Test week 2 — 1 week

5. **Data analysis**
   Data analysis is a continuous process and will be performed after every test. When testing in the cyberzoo, data analysis will probably be part of the design iteration. Between the two wind tunnel tests there is some time to analyse the data properly and make changes to the control systems. After the wind tunnel test 2 there is more time to analyse the data in depth.

   5.1. Data analysis during/after cyberzoo testing — 1 week

   5.2. Data analysis after wind tunnel tests — 1 week

6. **Cyberzoo testing 2**
   After the wind tunnel tests there is time left to do some extra testing in the cyberzoo. Depending on the progress the system can be improved even further or more data can be gathered. Some suggestions are given below:

   6.1. Increase controller performance — 3 weeks

   6.2. Improved state estimator — 1 week

   6.3. Data generation starting from various initial conditions — 1 week

7. **Midterm review**
   The midterm review will take place **April 1, 2019**.

8. **Greenlight review**
   The greenlight review will take place **June 3, 2019**.

9. **Thesis defence**
   The thesis will be handed in on **June 24, 2019**. The thesis defence will take place on **July 1, 2019**.

   9.1. Finalise results 1 week

   9.2. Writing 6 weeks

   9.3. Presentation preparation 1 week

5-2-2 Gantt Chart

The items from the activity plan presented in Section 5-2-1 can be seen in the gantt chart in Figure 5-1. The gantt chart was created using *instagannt*.

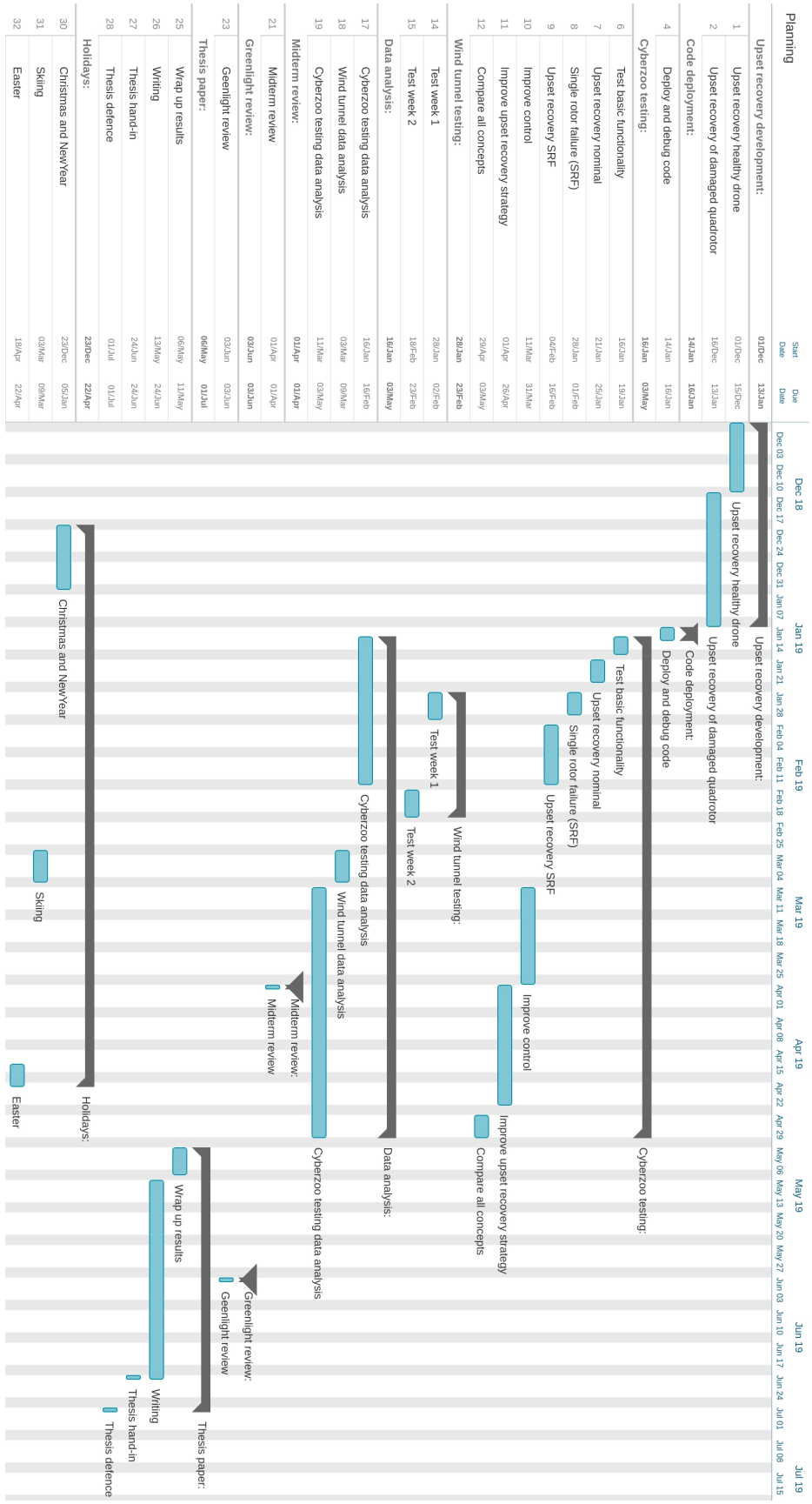

Figure 5-1: Gantt chart

Chapter 6

Preliminary results

Using the knowledge gathered during the literature research, some preliminary tests were setup. In this chapter the preliminary results will be presented. In Section 6-1 some initial tests are performed with the current control methods presented in the literature study. In particular, the control systems presented in [1] will be used as a baseline. In Section 6-2 some adaptations to the control system will be made in an attempt to solve some of the issues already. The results of the adaptations are presented in Section 6-3.

6-1 Initial testing

As a first step in the research, the current method presented in [1] was tested. The author was given the Simulink model of the control system as well as a Simulink model in which a simulator was present. This greatly helps understanding the control method and gives the opportunity to test the system and discover the limitations of it. In the simulator the Euler angle approach was used for the EOM. As this research is interested in attitudes beyond the normal conditions, a solution had to be found. The simulator was rewritten such that it would use the quaternion approach as described in 3-1-2. The tests will only vary the initial condition and see how the quadrotor reacts to it. The tests will only consider the quadrotor without damage. In further research the damaged case will also be investigated. The three tests that will be performed are:

1. Level flight with a position offset from the target.
2. A high initial roll angle of 80 °and see how the quadrotor recovers.
3. An upset condition where the drone is completely upside down. An initial roll angle of 180 °will be used.

Test 1

In this test only a position offset is given. The initial position of the quadrotor is at $x = -1, y = -1$ and the target is at $x = 0, y = 0$. The result is shown in Figure 6-1. One can see that without issue, the quadrotor flies towards the target location. During hover, the rotor speeds are very stable.

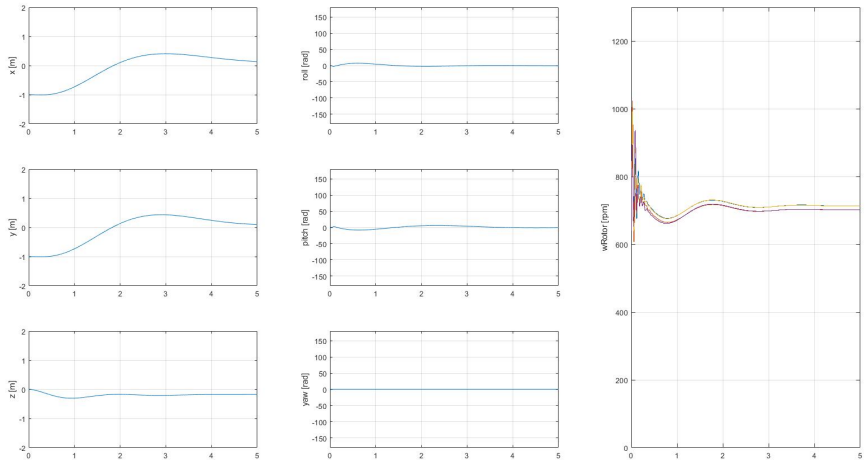

Figure 6-1: Test 1 with base controller

Test 2

In this test a roll offset of 80 °is given as initial condition. This is a very high roll angle in which the drone cannot properly control altitude anymore. The result can be seen in Figure 6-2. Clearly, something goes wrong. The attitude cannot be recovered and the quadrotor starts spinning as can be seen in the roll angle plot. The y position drifts further and further away and the altitude is lost as well (note that positive z-axis is 'falling'). When looking at the rotor speeds it becomes clear that all propellers quickly saturate, resulting in a loss of attitude control. The saturation is due to the altitude controller requiring full force. This becomes clear when looking at Equation (3-20).

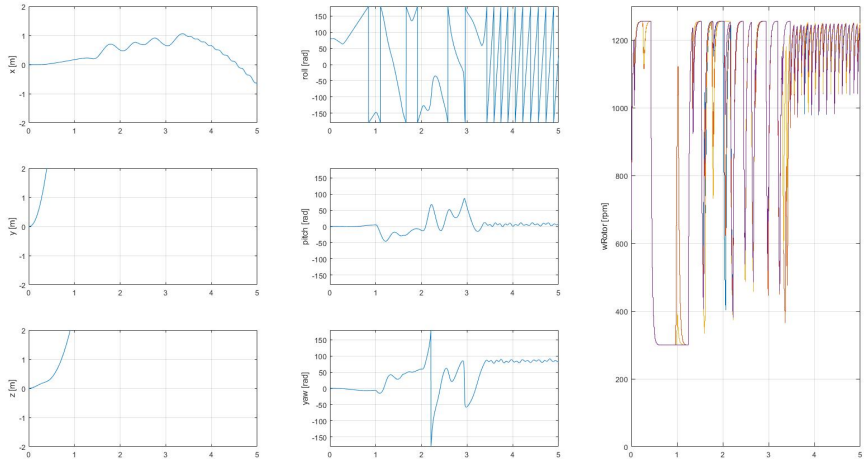

Figure 6-2: Test 2 with base controller

Test 3

In this test a roll angle of 180 °is given as initial condition. This is basically an upside down position. The result can be seen in Figure 6-3. As one can see, the quadrotor more or less stays exactly in the upside down condition and loses altitude drastically. Looking at Equation (3-24)

one can see that if the quadrotor is in an upside down position ($n_{des} = [0, 0, 1] = [h_1, h_2, h_3]$) no attitude action is required from the controller. Hence, the method only works for conditions where $-\frac{\pi}{2}$ or $\frac{\pi}{2}$.

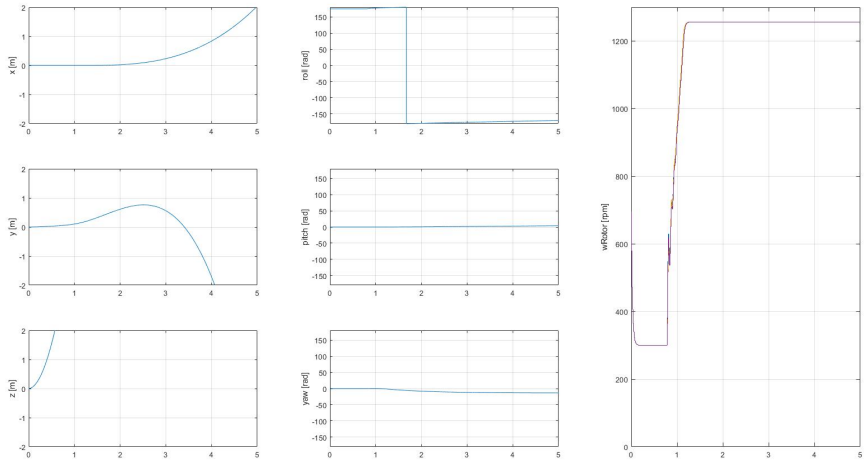

Figure 6-3: Test 3 with base controller

6-2 Adaptations

In this section some adaptations that will be made are presented. In order to get the control systems to work in extreme conditions, only small changes have to be made to certain controllers. Some concepts that were thought of will be implemented and the tests will be performed again to see how it affects the performance.

Limit on desired primary axis

One way to solve the upset recovery problem is to prevent an upset condition from happening. Currently there is no limit on the output of the position controller. This means that if the position error is too big, the desired primary axis will be more or less level with the ground. The attitude controller will try to match this desired primary axis and by doing so entering an upset condition similarly to what is described in test 2. In order to avoid these situations, a limit on the desired primary axis is introduced. The limit defines the maximum angle between the inertial z-axis and the desired primary axis. The limit can be set by the user, but should in future developments also take damage and other effects such as wind into account.

NDI Attitude controller

In the baseline controls the desired roll and pitch rate are derived from Equation 3-24. However, at 90°, h_3 becomes zero and thus there is no solution. Also in case h_3 is positive (so drone is 'upside down') the solution would control the drone into upside down position.

A different solution was implemented in order to solve this problem. Given the current primary axis and the target primary axis, the rotation axis is calculated using the cross product between target and current primary axis which can be seen in Equation (6-1). The rotation axis will be normalised as well. Using this rotation axis and the total angle α_{total} between the two vectors, a simple proportional controller can be made described in Equation 6-2. More research is needed in order to improve this controller such that it also takes yaw rate into account as in 3-24. If n_{des}^B

and n^B are aligned, no rotational axis will be found using the method. In this case an arbitrary rotational axis can be chosen as long as it is perpendicular with the vectors.

$$n_{rotaxis} = n^B_{des} \times n^B \tag{6-1}$$

$$\begin{bmatrix} p_{des} \\ q_{des} \\ \sim \end{bmatrix} = n_{rotaxis} \cdot \alpha_{total} \cdot k_p \tag{6-2}$$

This method allows for more research into the rotation axis. The determination of the rotation axis could depend on the current state of the quadrotor in terms of what damage is present or what angular velocities are working on the quadrotor. One can imagine that if a rotor has failed, it is more favourable to rotate the quadrotor around a certain axis.

Altitude controller

In the baseline controls, the altitude controller takes roll and pitch angle into account as described in Equation (3-20). This method works perfect for normal conditions but will be singular at 90 °. Also the control action at high pitch and roll angles will become very high, causing saturation of the propellers. A simple solution to this problem is implemented in which limits to θ and ϕ are applied in Equation Equation (3-20), but also a reduction factor is implemented such that at extreme angles the altitude control would be given up. Calculation of the reduction factor (η_{red}) can be found in Equation (6-3) and Equation (6-4) where α_{peak} and α_{LOC} are settings and α_{total} is the combined pitch and roll angle. The reduction factor is limited between 0 and 1. Practically this means that for all $\alpha_{total} < \alpha_{peak}, \eta_{red} = 1$ and for all $\alpha_{total} > \alpha_{LOC}, \eta_{red} = 0$. In this way the altitude control is scaled down if pitch and roll angle are too extreme. Equation (6-5) shows the final form.

$$\eta_{red} = 1 - (\alpha_{total} - \alpha_{peak})/(\alpha_{LOC} - \alpha_{peak}) \tag{6-3}$$

$$\eta_{red} = min(max(\eta_{red}, 0), 1) \tag{6-4}$$

$$\sum f_i = \frac{m(\ddot{z}_{des} - g)}{cos(min(\theta, \theta_{max})cos(min(\phi, \phi_{max})} \cdot \eta_{red} \tag{6-5}$$

6-3 Results

The tests performed in Section 6-1 can now be performed again to see if the adaptations solve some of the issues. The tests are presented in the same fashion.

Test 1

In this test only a position offset is given. The initial position of the quadrotor is at $x = -1, y = -1$ and the target is at $x = 0, y = 0$. The result is shown in Figure 6-1. The quadrotor moves to the target smoothly without any issue. The tuning of the controller is clearly a bit more aggressive compared to the baseline controller.

Test 2

In this test a roll offset of 80 °is given as initial condition. This is a very high roll angle in which the drone cannot properly control altitude anymore. The result can be seen in Figure 6-2. The

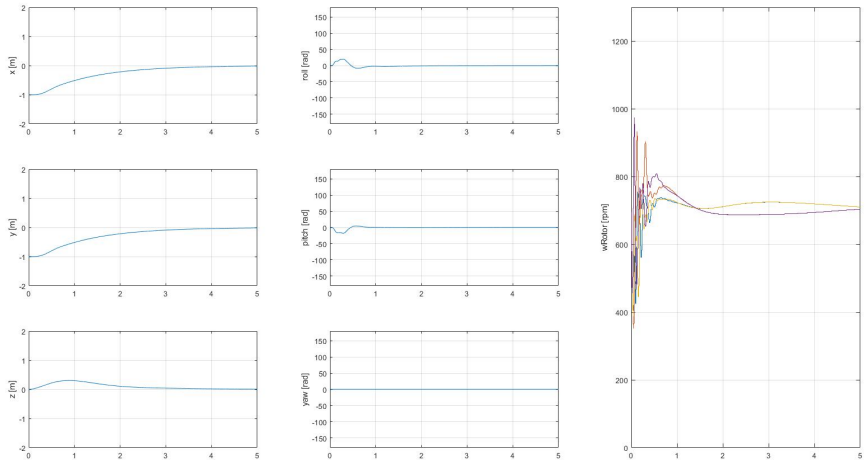

Figure 6-4: Test 1 with adapted controller

quadrotor seems to recover well. The attitude controller manages to recover the high initial roll angle in about half a second. The quadrotor does not lose more than $0.5m$ in altitude and does not deviate more than $0.5m$ laterally. The rotors are not being saturated which can be seen in the rotor speed graph.

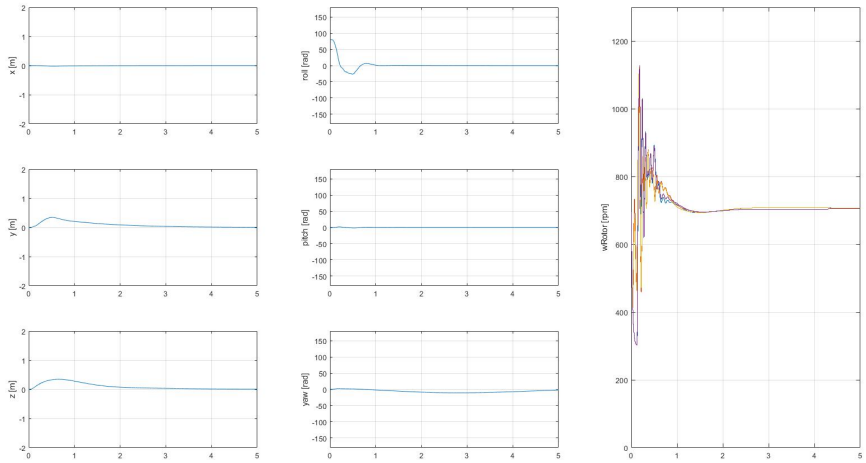

Figure 6-5: Test 2 with adapted controller

Test 3

In this test a roll angle of 180 °is given as initial condition. This is basically an upside down position. The result can be seen in Figure 6-6. The quadrotor manages to recover without any issue. The roll angle is recovered in less than a second. This means less than $1m$ is lost in altitude. These are very promising results and show that the quadrotor can quickly recover. Again one can see that the rotors are not being saturated.

Test conclusion

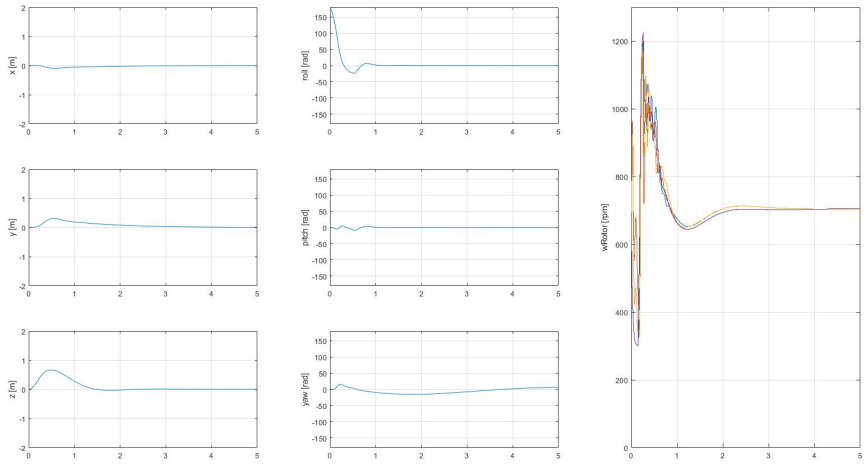

Figure 6-6: Test 3 with adapted controller

Clearly, with the small adaptations to the baseline controller significant performance gains are obtained with regards to recovery of a nominal quadrotor. The two key aspects that caused the baseline controller to have difficulties were the saturation of the rotors due to the altitude controller and the handling of high pitch and roll angles in the attitude controller. These results are very promising for the research. The next step in the research is to develop a recovery strategy that also works for the damaged quadrotor.

Part III

Appendices

Appendix A

State estimation

The presented control system needs position, velocity and attitude information. Position and attitude are measured using an external motion capture system from *Optitrack*, velocity has to be estimated. The external motion capture system consists of 12 cameras tracking reflecting markers on the quadrotor. However, in case the markers are not visible for the cameras, no position and attitude information can be given. Tracking can get lost if the quadrotor has extreme pitch and/or roll angles. One can imagine that when the quadrotor is upside down, the cameras are not able to see the markers anymore. Also at high rotational rates the quality of the motion tracking decreases. The update frequency of *Optitrack* is lower than the update frequency of the control system. In order to improve the quality of the position and attitude estimation and estimate the velocity of the vehicle, an Extended Kalman Filter is proposed. The Extended Kalman Filter also has to cope with the temporary unavailable measurements from *Optitrack* and the different update frequency.

A-1 Solution

An Extended Kalman Filter was chosen due to the non-linear nature of the quadrotor dynamics. A general non-linear system can be described by

$$\dot{x} = f(x, u) + w \tag{A-1}$$

$$z = h(x, u) + v \tag{A-2}$$

where x indicates the state of the system, u indicates the inputs to the system, z are the measurements, w is the process noise while v is the measurement noise. f is the state update function and h is the measurement function. The states that will be chosen for the quadrotor are the position, velocity and attitude described by a quaternion. The state update function (f) for a quadrotor can be described by

$$\begin{bmatrix} \dot{\boldsymbol{d}} \\ \dot{\boldsymbol{v}} \\ \dot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{g} + \boldsymbol{q} \otimes \boldsymbol{a}^B \otimes \boldsymbol{q}* \\ \frac{1}{2}\boldsymbol{\Omega} \otimes \boldsymbol{q} \end{bmatrix} + \boldsymbol{w} \tag{A-3}$$

where $\boldsymbol{d}$ is the position, $\boldsymbol{v}$ is the velocity, $\boldsymbol{q}$ is the quaternion describing attitude, $\boldsymbol{g}$ is the gravity vector, $\boldsymbol{a}^B$ is the body acceleration measurement and $\boldsymbol{\Omega}$ are the measured rotational rates. The $\otimes$ indicates the Kronecker product which is used in quaternion mathematics. The measurement function (h) can be described by

$$\begin{bmatrix} \boldsymbol{d}_{meas} \\ \boldsymbol{q}_{meas} \end{bmatrix} = \begin{bmatrix} \boldsymbol{d} + \boldsymbol{q} \otimes \triangle \boldsymbol{d} \otimes \boldsymbol{q}* \\ \boldsymbol{q} \end{bmatrix} + \boldsymbol{v} \tag{A-4}$$

where $\triangle \boldsymbol{d}$ indicates the offset position between the center of gravity and the measurement from *Optitrack*. $\boldsymbol{d}_{meas}$ and $\boldsymbol{q}_{meas}$ indicate the measured position and measured quaternion.

In order to solve the EKF first the Jacobian of the state update function ($\boldsymbol{F}_k$) and the measurement function ($\boldsymbol{M}_k$) have to be calculated as shown in

$$\boldsymbol{F}_k = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \mid_{x_k} \tag{A-5}$$

$$\boldsymbol{M}_k = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \mid_{x_k} \tag{A-6}$$

This is not done manually but by using the *jacobian* function in MATLAB. Subscripts k and $k-1$ indicate if the value is from the current or previous step. A combination like $k, k-1$ indicates a prediction of the current step from the previous step. The prediction part of the EKF is given by

$$\boldsymbol{x}_{k,k-1} = f(\boldsymbol{x}_{k-1,k-1}, \boldsymbol{u}_k) \tag{A-7}$$

$$\boldsymbol{P}_{k,k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1,k-1} \boldsymbol{F}_k^T + \boldsymbol{w}_k \tag{A-8}$$

where $\boldsymbol{P}$ is the state covariance matrix. The measurement part of the EKF is given by

$$\boldsymbol{y}_k = \boldsymbol{z}_k - h(\boldsymbol{x}_{k,k-1}) \tag{A-9}$$

$$\boldsymbol{S}_k = \boldsymbol{M}_k \boldsymbol{P}_{k,k-1} \boldsymbol{M}_k^T + \boldsymbol{v}_k \tag{A-10}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_{k,k-1} \boldsymbol{M}_k^T \boldsymbol{S}_k^{-1} \tag{A-11}$$

where $\boldsymbol{y}$ is the measurement residual and $\boldsymbol{S}$ is the residual covariance. $\boldsymbol{K}$ is the Kalman gain. The state estimation is completed with

$$\boldsymbol{x}_{k,k} = \boldsymbol{x}_{k,k-1} + \boldsymbol{K}_k \boldsymbol{y}_k \tag{A-12}$$

$$\boldsymbol{P}_{k,k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{M}_k) \boldsymbol{P}_{k,k-1} \tag{A-13}$$

where $\boldsymbol{x}_{k,k}$ is the final estimated state and I the identity matrix.

To check if the measurement has updated one can compare it to the previous value and see if it changed as shown in

$$\boldsymbol{z}_k - \boldsymbol{z}_{k-1} < \delta \tag{A-14}$$

where δ is tunable. In the case the measurement did not update, the measurement part of the EKF is skipped and only the prediction is used. Practically one can achieve this by forcing a Kalman gain $\boldsymbol{K}$ of 0.

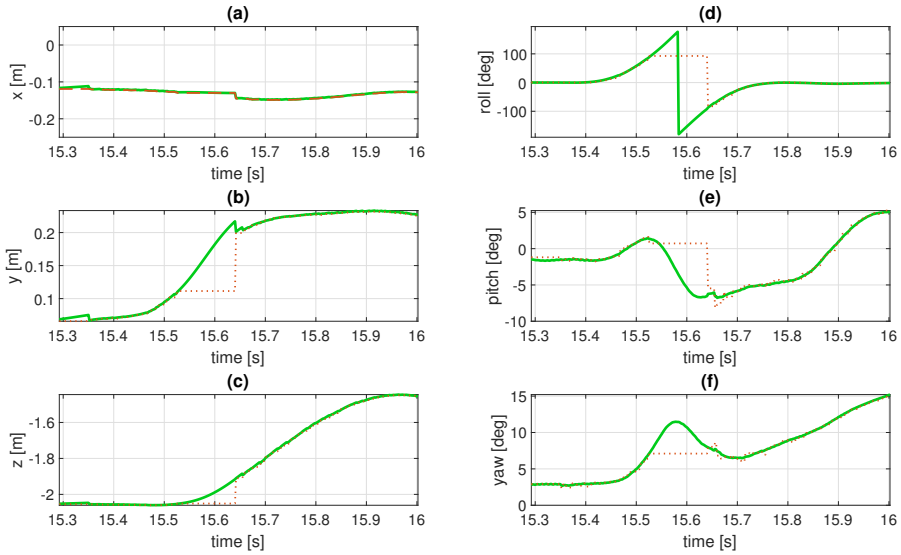

Figure A-1: State estimation during flip. Red line measurement, green line estimated value. (a) x-position. (b) y-position. (c) z-position. (d) roll angle. (e) pitch angle. (f) yaw angle.

A-2 Results

In order to show the performance of the EKF, the attitude and position estimation during a flip is shown in Figure A-1. The red line indicates the measured value while the green line indicates the estimated value. One can see that at some point the measured value is not updating anymore and is ignored. Once the measurement comes back online, it is immediately used to correct the estimation. A more detailed view is given in Figure A-2 where one can see that even for brief amounts of time the measurements are not updating due to the update frequency difference between the control systems and *Optitrack*. In Figure A-3 the velocity estimation is shown in green, while the red line is the derivative of the position measurement. Clearly the EKF is needed for velocity estimation, although one could try to estimate the velocity by smoothing the derivative shown, or by only using the position measurements when they are updated.

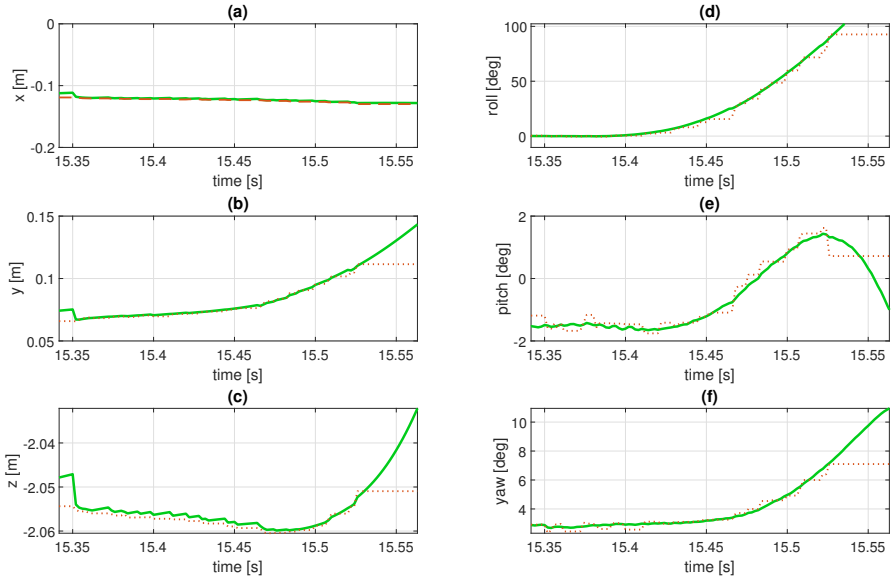

Figure A-2: State estimation during flip (detail). Red line measurement, green line estimated value. (a) x-position. (b) y-position. (c) z-position. (d) roll angle. (e) pitch angle. (f) yaw angle.

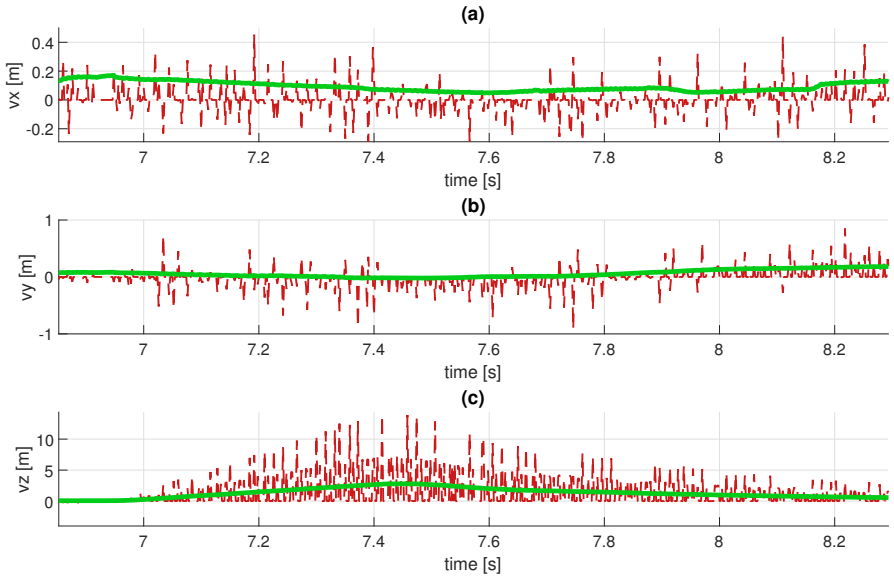

Figure A-3: Velocity estimation during flip. Red line position derivative, green line estimated value. (a) x-velocity. (b) y-velocity. (c) z-velocity.

Appendix B

Control Allocation

The control allocator has been discussed in detail in the paper. However some extra information on how the quadratic programming problem is solved will be given in this chapter. A big issue when solving a problem of this kind is the execution speed. As the control allocator has to run real-time on-board, some execution speed optimisation was required.

B-1 Solution

The quadratic programming problem that will be solved is of the standard form shown in Equation (B-1)

$$\begin{aligned} \text{minimise} \quad & \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{c}_0 \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} \end{aligned} \tag{B-1}$$

where $\boldsymbol{H}, \boldsymbol{c}^T, \boldsymbol{c}_0$ describe the cost function, $\boldsymbol{A}$ and $\boldsymbol{b}$ describe the constraints and $\boldsymbol{x}$ is the solution to be found. The key to finding the solution to the quadratic programming problem is to find the active Lagrange multipliers ($\boldsymbol{y}$). As there are 8 constraints and the constraints can either be active or not, this leaves us with $2^8 = 256$ options to choose from. However, as a rotor cannot be at its minimum and maximum constraint at the same time (unless they are equal but then this assumption still works) there are only $3^4 = 81$ feasible options left.

An initial guess for the active Lagrange multipliers ($\boldsymbol{y}$) can be made by looking at the previous solution of the problem. By taking the same Lagrange multipliers as the final result of the previous step the execution performance of the optimisation is greatly increased.

Another option for the initial guess of the active Lagrange multipliers is by solving the INDI problem without taking rotor saturation into account as shown in

$$\boldsymbol{E} = \begin{bmatrix} \triangle\boldsymbol{\Omega}_{des} \\ \boldsymbol{a}^B_{ref,z} \end{bmatrix} - \begin{bmatrix} \dot{\boldsymbol{\Omega}} \\ a_z \end{bmatrix} \tag{B-2}$$

$$\boldsymbol{E} = \boldsymbol{G} \triangle \boldsymbol{\omega} \quad \rightarrow \quad \triangle\boldsymbol{\omega} = \boldsymbol{G}^{-1}\boldsymbol{E} \tag{B-3}$$

where $\triangle\boldsymbol{\Omega}_{des}$ is the desired change in rotational rate, $\dot{\boldsymbol{\Omega}}$ is the measured change in rotational rate, $\boldsymbol{a}^B_{ref,z}$ is the desired body acceleration in the thrust direction and a_z is the measured acceleration. G is the control effectiveness matrix and $\triangle\boldsymbol{\omega}$ is the desired change in rotor speed. If the solution $\triangle\boldsymbol{\omega} > \omega_{max}$ or $\triangle\boldsymbol{\omega} < \omega_{min}$ the according Lagrange multiplier can be made active as an initial guess. Once an initial guess is made, the solution according the active Lagrange multipliers can be found by solving

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{A}_{eq}^T \\ \boldsymbol{A}_{eq} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{c} \\ \boldsymbol{b}_{eq} \end{bmatrix} \tag{B-4}$$

where $\boldsymbol{A}_{eq}$ and $\boldsymbol{b}_{eq}$ are the $\boldsymbol{A}$ and $\boldsymbol{b}$ matrices with only the rows of the active Lagrange multipliers. The solution $\boldsymbol{x}$ represents the incremental change in rotor speeds $\triangle\boldsymbol{\omega}$. The solution for $\boldsymbol{y}$ are the values of the chosen Lagrange multipliers. Finding a solution does not mean the quadratic programming problem has been solved. The solution has to satisfy

$$\boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} \tag{B-5}$$

$$\boldsymbol{y} \geq 0 \tag{B-6}$$

In the case that these equations are not satisfied, a new iteration has to be performed with different Lagrange multipliers. Depending on which condition has been breached the according Lagrange multiplier will change as can be seen in Algorithm 1.

if $\boldsymbol{y} < 0$ **then**
| Remove biggest violation from Lagrange multipliers
else
| Add biggest constraint violation ($\boldsymbol{A}\boldsymbol{x} < \boldsymbol{b}$) to Lagrange multipliers;
end

Algorithm 1: Update Lagrange multipliers

B-2 Results

In this section the execution performance of the optimisation algorithm is presented. The amount of iterations for the different methods to select the initial Lagrange multipliers can be seen in Figure B-1. One can see that using the previous Lagrange multipliers results in the best performance. Using the INDI solution to determine the Lagrange multipliers is slightly better compared to using no initial guess.

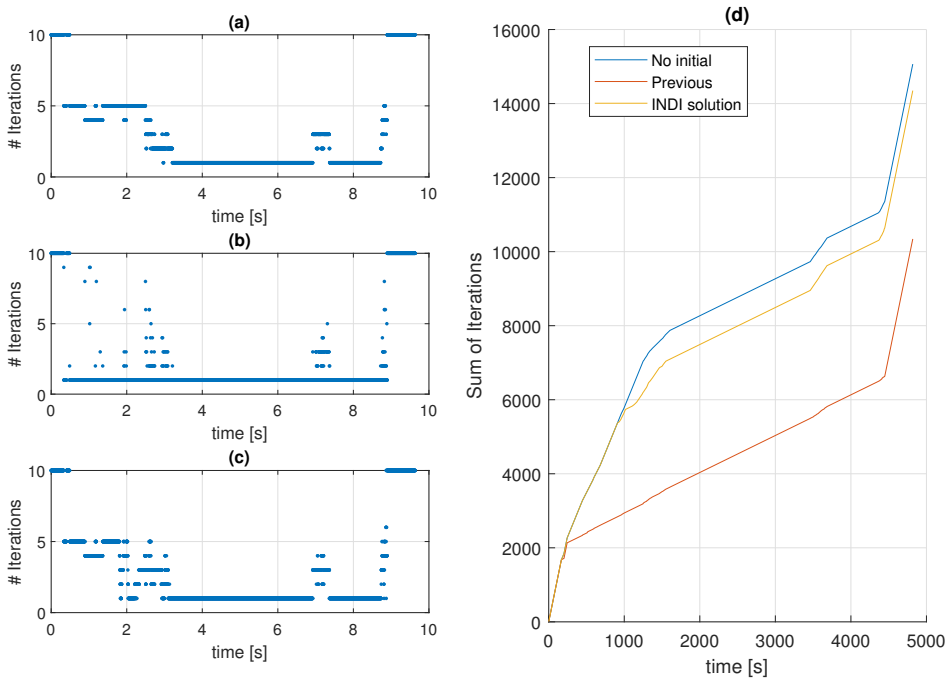

Figure B-1: Control allocator iterations. (a) No initial guess for $\boldsymbol{y}$. (b) Previous $\boldsymbol{y}$ as initial guess. (c) INDI solution as initial guess for $\boldsymbol{y}$. (d) Cumulative iterations for the three methods.

Appendix C

Estimated time to rotate

In the paper the time estimated to rotate was estimated in order to determine which direction would be quickest given initial rotational rate (Ω_0) and an initial angle (θ_0) around a certain axis as visualised in Figure C-1. Assuming bang-bang control, so fully accelerating followed by full braking, the total time (t) is given by

$$t = t_A + t_B \tag{C-1}$$

where t_A is the time accelerating and t_B is the time braking. There should be no rotational rate at the end of the manoeuvre meaning

$$0 = \Omega_0 + t_A a_{max} + t_B a_{min} \tag{C-2}$$

where a_{max} and a_{min} are the maximum and minimum rotational accelerations around that axis. The total rotation angle can be described as follows

$$\theta_0 = \Omega_0 t + \frac{t_A^2 a_{max}}{2} + \frac{t_B^2 a_{min}}{2} \tag{C-3}$$

Using the above three equations, one can come up with the following equation

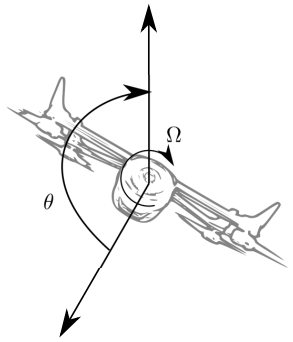

Figure C-1: Rotation of the quadrotor

$$0 = -\theta_0 + \Omega t + \Omega_0^2 \left(\frac{t \cdot a_{min} + \Omega_0}{a_{min} - a_{max}} \right) \frac{a_{max}}{a_{min}} + \left(\frac{t \cdot a_{min} + \Omega_0}{a_{min} - a_{max}} \right)^2 \left(1 + \frac{a_{max}}{a_{min}} \right) \frac{a_{max}}{2} \tag{C-4}$$

Solving for t will give the time needed to rotate the quadrotor. In order to calculate the time to rotate for the other direction one has to change the sign of Ω, switch a_{max} and a_{min} and the conjugate angle of θ.

This method only considers rotational motion around a single static axis, which makes it a linear motion and easy to solve. In the case of significant rotational rates around another axis, the estimation of time rotate will not be correct and a quicker way to rotate might exist. This is however not considered in this thesis project.

Appendix D

Simulation

In order to develop a control system, one needs a platform to develop and test new concepts. As real life testing is costly, time consuming and requires a good setup in order to analyse data properly, a simulation environment is used for developing and testing the control systems. Once the performance is sufficient, real life tests can take place. In this chapter the simulation environment will be elaborated upon as it is an essential part of the development of the final control system.

D-1 Framework

A common tool for developing control systems is *Simulink*. For this thesis project a framework was setup in *Simulink* in which different controllers can be tested in the same environment. The framework is setup to be as similar as real life as possible, this means that the controller and the vehicle dynamic simulations are separated completely. Sensor simulation is also done, which means the sensors will be rotated, biased and noise will be added as required. The interface between the controllers and the other parts of the framework are the same as it would be when deployed in real-life. The framework can be seen in Figure D-1.

Model referencing is used as much as possible in order to separate the files. This makes source controlling the project a lot easier as *Simulink* files are binaries. As mentioned before, different controllers can be tested in this environment. This is done by model referencing in a variant selector block as shown in Figure D-2. This makes the framework easy to use for other people in the project.

The vehicle simulation part of the framework also consists of several modules namely: wind simulation, actuator dynamics, aerodynamic model, and equations of motion based on quaternions. For every model one can make several modes. In this project for example, a high and low fidelity aerodynamic model is available. For initial testing of a controller it can be useful to test with a simple model in which the output is quite straight-forward.

Another advantage of having a common framework is that data analysis can be made standardised. This means that several controllers can be simulated and analysed in the same way, which makes comparing performance very easy.

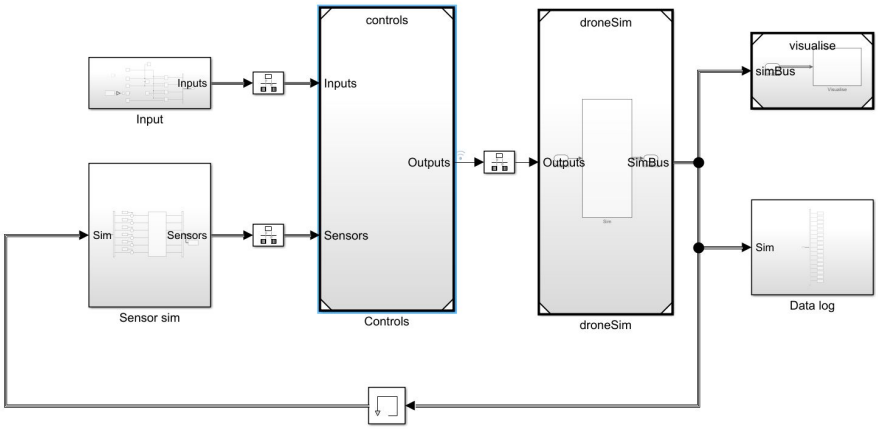

Figure D-1: *Simulink* framework

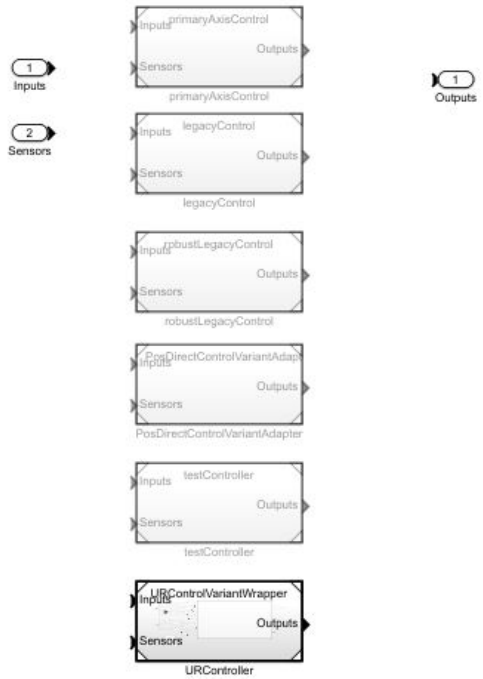

Figure D-2: *Simulink* variant controllers

D-2 Actuator Model

The actuator response is also modelled in the simulation. This is very relevant as the actuator has some delayed response to inputs. This delay has a major effect on the inner loop control system. A common way of modelling the actuator is to use a simple Low Pass Filter (LPF) as shown in Equation (D-1)

$$\boldsymbol{\omega} = \text{LPF}(\boldsymbol{\omega}_{set}) \tag{D-1}$$

where $\boldsymbol{\omega}$ and $\boldsymbol{\omega}_{set}$ are the rotor speeds and rotor speed setpoints. Although this method is very simple to implement there are some downsides. The behaviour when the rotor speeds are increasing or decreasing is different, which is not modelled with the LPF. Another downside is that an LPF implementation is hack-able. Requesting a very high rotor speed delta will give a quicker response meaning that if more subtle changes are requested, the actuators will react very slowly. This behaviour can be solved by using a rate limiter instead of a LPF as shown in Equation (D-2)

$$\boldsymbol{\omega} = \boldsymbol{\omega} + (\triangle\boldsymbol{\omega}_{min} \leq \boldsymbol{\omega}_{set} - \boldsymbol{\omega} \leq \triangle\boldsymbol{\omega}_{max}) \tag{D-2}$$

where $\triangle\boldsymbol{\omega}_{min}$ and $\triangle\boldsymbol{\omega}_{max}$ should be set according to the rotors used. In this method a LPF with smaller delay is still used as the response of the rotors will never be instantaneous. In this project, the rotors were not able to produce negative torque, meaning that the response of decreasing rotor speed is slower compared to increasing rotor speed. This is now able to be modelled. Modelling the aerodynamic drag of the propellers would increase the model fidelity even more, but this was not done in this project.

D-3 Aerodynamic Model

As mentioned, one can implement various aerodynamic models in the framework. In this project two models are used: a low fidelity model and a high fidelity model. The low fidelity model assumes the lift force to have a quadratic relationship with rotor speed as shown in

$$f_i = c_{f,i}\omega_i^2 \tag{D-3}$$

where f_i is the lift force of rotor i, $c_{f,i}$ is the lift coefficient of rotor i and ω_i the rotor speed of rotor i. One can make the lift coefficient of each rotor slightly different from each other in order to create some disturbance in the model to test the robustness of the controller. In real-life the propellers will also have slightly different properties. Given the individual rotor force, the total force and moment on the vehicle can be calculated which is presented in the paper. As body drag force is not modelled, wind can not be simulated or has no effect.

The high fidelity model is a lot more complex and uses wind tunnel test data in order to make an estimation of the forces and moments acting on the quadrotor. With this model wind can also be taken into account. The model takes air speed, rotational rate and rotor speeds into account as shown in

$$[\boldsymbol{F}_b, \boldsymbol{M}_b] = \text{aeroMap}(\boldsymbol{V}_{air}, \boldsymbol{\Omega}, \boldsymbol{\omega}) \tag{D-4}$$

where $\boldsymbol{F}_b$ and $\boldsymbol{M}_b$ are the forces and moments acting on the vehicle. $\boldsymbol{V}_{air}$ is the airspeed of the vehicle which depends on the inertial velocity of the vehicle and the wind. This model was developed by Sihao Sun.

D-4 Sensor simulation

In simulation it is often tempting to forget about imperfect sensor measurements. Noise and bias is always present on sensor measurements in real-life and should also be modelled in order to truly test the robustness of the controller. In this project, every sensor is simulated by adding a bias and band-limited noise with a certain power

$$m_{sens} = m_{sim} + v + b \tag{D-5}$$

where m is the measured state, b the bias and v the noise. The values for the bias (b) and noise (v) power can be determined from real-life data, however one can also change them in order to check the consequences. The noise on the sensors is generally not that high in standstill, however when the quadrotor is flying the noise seems to be a lot higher. This 'noise' comes from high frequency vibrations caused by the rotors. In the simulator this is simplified to more powerful noise.

As mentioned in Appendix Appendix A the *Optitrack* external motion capture system is not able to track the quadrotor in extreme conditions. This is also modelled in the sensor simulation. If the attitude exceeds a certain threshold ($n_z > c_{lim}$), the signal will not update anymore like in real-life.

Appendix E

Software implementation

In this chapter the practical information concerning the implementation will be given. In order to validate the developed control strategy the code has to be converted into C/C++ and deployed onto the quadrotor. In Appendix E-1 the code generation and code deployment will be explained. In order to control the quadrotor before, during and after flight an application called *QGroundControl* is used which is further explained in Appendix E-2.

E-1 Code Generation and Deployment

As mentioned in Appendix D the control systems are being developed in *Simulink* in a framework together with the simulator. The goal of this framework was to emulate the real world as good as possible. It is important that in the framework the interface (inputs/outputs) are exactly the same as when deployed on the real quadrotor. It will now become clear why this is so important.

Code developed in the *Simulink* environment can not be used directly on a real-time machine (* *Simulink* does offer hardware-in-the-loop solutions, but these are not being used). The code therefore has to be converted into C++. In many projects the developed control system is then being written into C++ by hand, which is very time-consuming and often not error-prone. For this project the *Simulink* code will be compiled using the *Simulink* compiler, meaning the code will automatically be converted into C++ code.

The compiled code can then be implemented on the quadrotor. PX4 is used as framework on the quadrotor. In PX4 one can add modules which is very useful for this project. A module is made that is able to connect the correct input and outputs to the compiled code. Using this method, one can be sure that the control system used in the simulator is the same as the one deployed on the quadrotor. This method also allows for very quick development as a change in the control strategy can be tested in the simulator, compiled, deployed and validated in the timespan of 5 minutes. This process is visualised in Figure E-1.

Because the inputs/outputs of the code in the *Simulink* framework are the same as the ones of the deployed code on the real quadrotor, the test data can be logged and replayed in the *Simulink* environment in order to analyse the results even better. Problems or bugs can be quickly identified, solved and re-compiled using this method. This also allows for easy tuning of filters, controllers etc.. without having the risk of a crash.

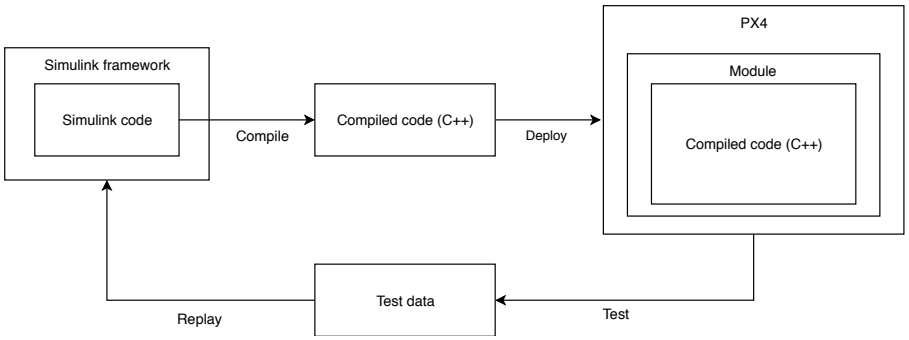

Figure E-1: Code generation and deployment

E-2 QGroundControl

Once the code is compiled and deployed, there has to be some way of controlling the quadrotor during the test. Several applications contribute to a flying drone but one will be highlighted which is *QGroundControl*. *QGroundControl* is the interface that the operator can use to control the quadrotor. An example use of *QGroundControl* is shown in Figure E-2. Using this interface one can easily calibrate the sensors, check battery voltage, check sensor outputs etc.. *QGroundControl* also has an interface to change parameters for the control system as shown in Figure E-3. With this interface the position setpoints can be changed, different controller modes can be enabled/disabled or specific controller parameters such as position control gains can be tuned before, during or after flying.

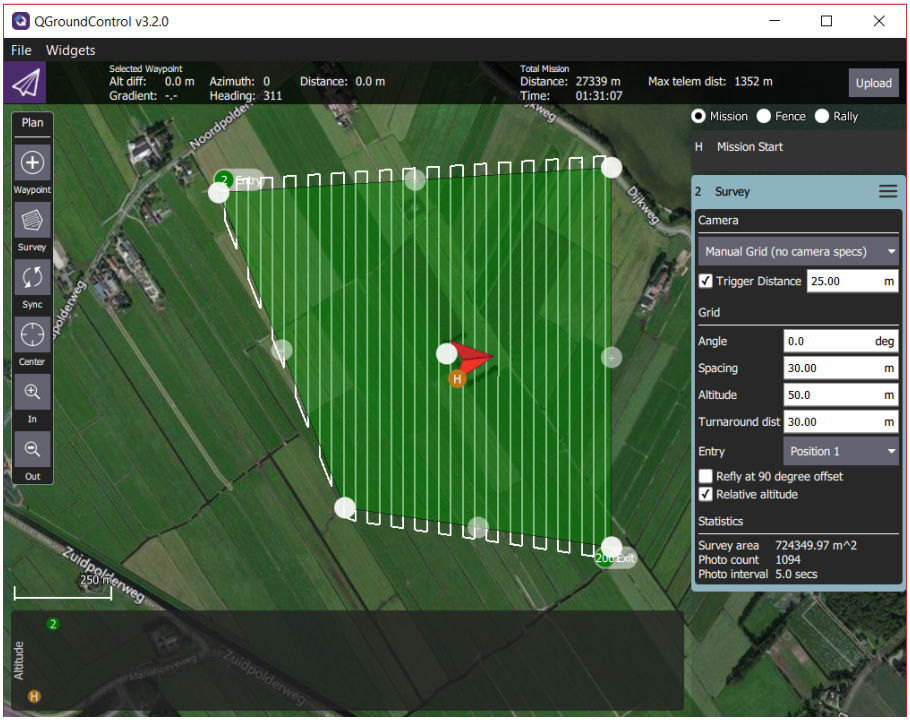

Figure E-2: QGroundControl main overview page example [5]

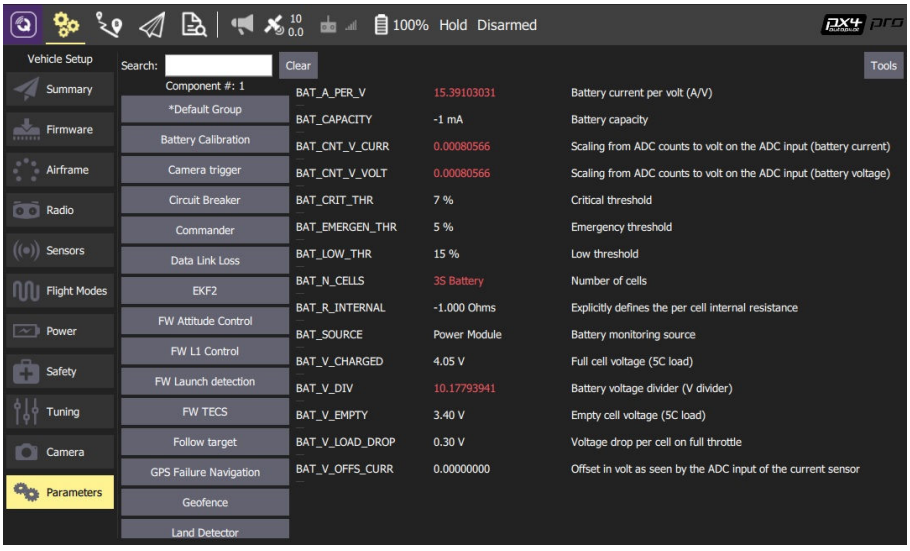

Figure E-3: QGroundControl changing control parameters example [6]

Appendix F

Extra results

In this chapter some extra graphs will be presented in order to visualise the 3D recovery data presented in the paper.

Figure F-1 visualises the recovery altitude needed based on $\Omega_{x'}$ vs $\Omega_{y'}$. One can see that negative $\Omega_{y'}$ has a bad impact on recovery, which makes sense as this is an uncontrollable rotation due to the failure on rotor 1.

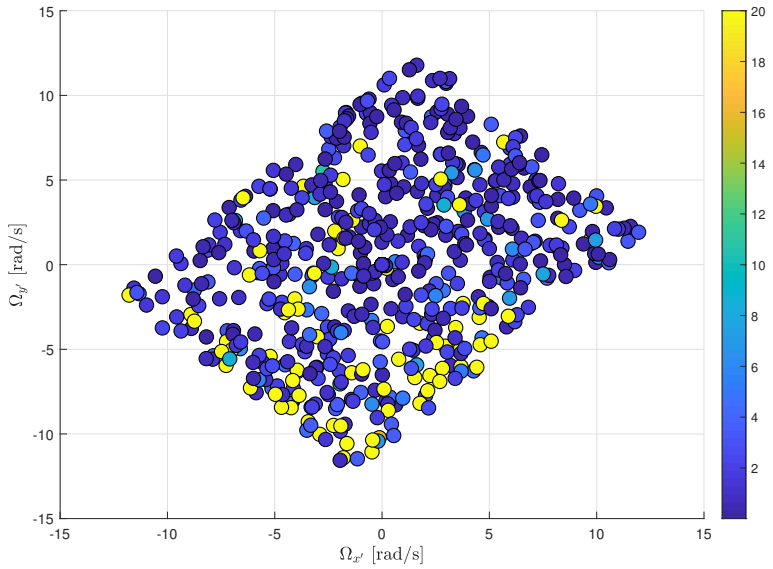

Figure F-1: Rotor 1 failure recovery altitude needed in meter. $\Omega_{x'}$ vs $\Omega_{y'}$ - Simulation data - 2D scatter.

In Figure F-2 the impact of positive Ω_z can very clearly be seen. With rotor 1 failed, the natural yaw rate will be negative, meaning recovery from an initial positive yaw rate will take longer.

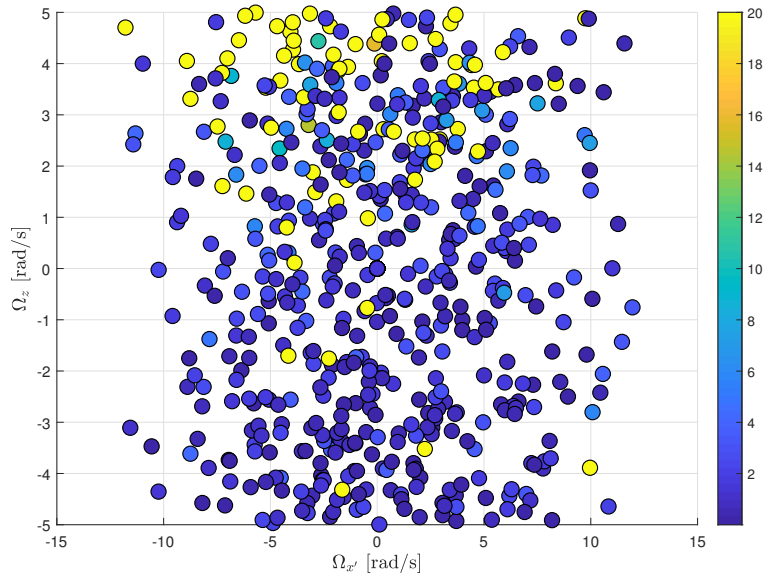

Figure F-2: Rotor 1 failure recovery altitude needed in meter. $\Omega_{x'}$ vs Ω_z - Simulation data - 2D scatter.

In Figure F-3 the combination of $\Omega_{y'}$ and Ω_z is plotted and one can still see the same effects as observed before. Negative $\Omega_{y'}$ and positive Ω_z are hard to recover from given failure on rotor 1.

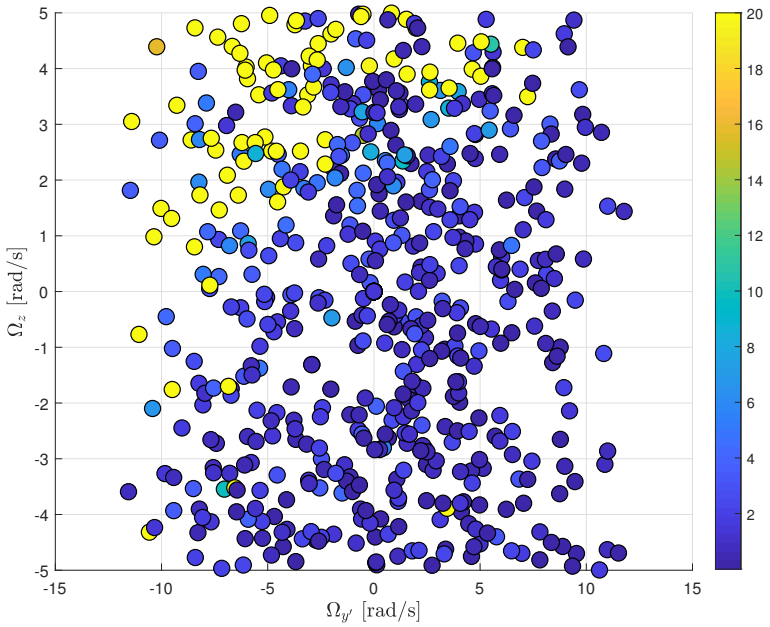

Figure F-3: Rotor 1 failure recovery altitude needed in meter. $\Omega_{y'}$ vs Ω_z - Simulation data - 2D scatter.

In Figure F-4 Ω_z is plotted vs n_z which is the z-component of the thrust vector. It was expected

that a positive n_z (upside down) with yaw rate would significantly impact recovery performance, this however can not be confirmed from this figure.

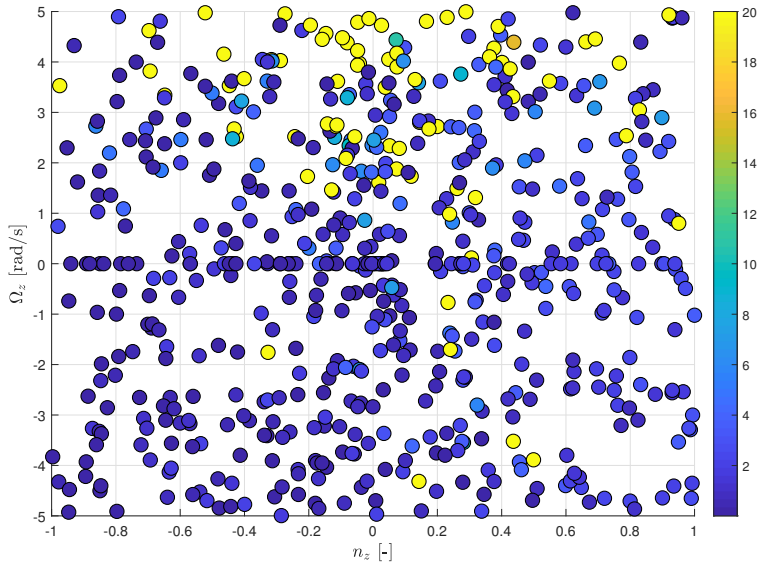

Figure F-4: Rotor 1 failure recovery altitude needed in meter. Ω_z vs n_z - Simulation data - 2D scatter.

In Figure F-5 the recovery of various initial attitudes without significant initial rotation rate is plotted. One can see that the performance is very good and the quadrotor manages to recover from any initial attitude in 10m or less.

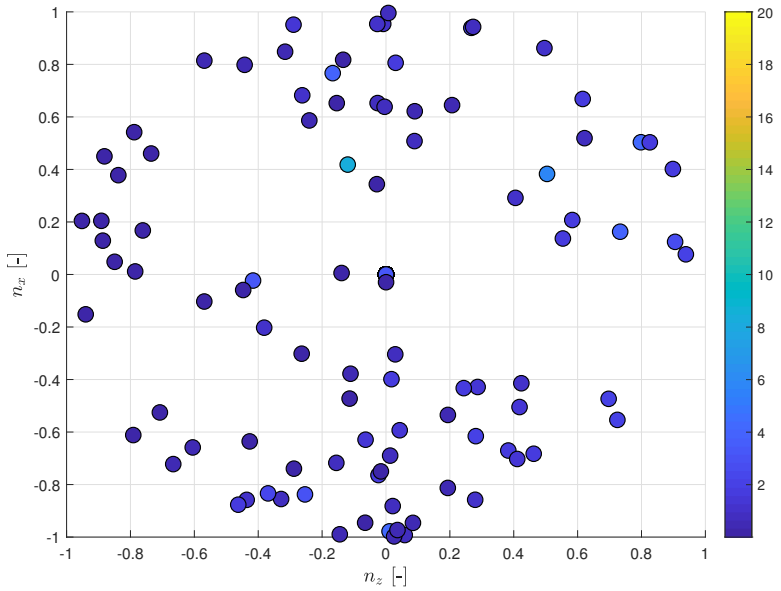

Figure F-5: Rotor 1 failure recovery altitude needed in meter assuming no significant initial rates. n_x vs n_z - Simulation data - 2D scatter.

Bibliography

[1] Sihao Sun, Leon Marinus Christiaan Sijbers, Xuerui Wang, and Coen de Visser. High-Speed Flight of Quadrotor despite Loss of Single Rotor. *IEEE Robotics and Automation Letters*, 3 (4):1–1, 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2851028. URL https://ieeexplore.ieee.org/document/8398406/.

[2] Wikipedia. Quaternion, November 2018. URL https://en.wikipedia.org/wiki/Quaternion.

[3] Jacobus A. Engelbrecht, Simon J. Pauck, and Iain K. Peddle. A Multi-Mode Upset Recovery Flight Control System for Large Transport Aircraft. *AIAA Guidance, Navigation, and Control (GNC) Conference*, pages 1–18, 2013. doi: 10.2514/6.2013-5172. URL http://arc.aiaa.org/doi/10.2514/6.2013-5172.

[4] Vincent Driessen. A successful git branching model, November 2018. URL https://nvie.com/posts/a-successful-git-branching-model/.

[5] Hamish. Qgroundcontrol 3.2 released, July 2017. URL http://qgroundcontrol.com/qgroundcontrol-3-2-released/.

[6] PX4 Dev Team. Px4 parameter configuration, May 2019. URL https://docs.px4.io/en/advanced_config/parameters.html.

[7] Ewoud J. J. Smeur, Qiping Chu, and Guido C. H. E. de Croon. Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Air Vehicles. *Journal of Guidance, Control, and Dynamics*, 39(3):450–461, 2016. ISSN 0731-5090. doi: 10.2514/1.G001490. URL http://arc.aiaa.org/doi/10.2514/1.G001490.

[8] Peng Lu and Erik Jan Van Kampen. Active fault-tolerant control for quadrotors subjected to a complete rotor failure. *IEEE International Conference on Intelligent Robots and Systems*, 2015-Decem:4698–4703, 2015. ISSN 21530866. doi: 10.1109/IROS.2015.7354046.

[9] Adam C. Uzialko. 10 cool commercial drone uses coming to a sky near you, May 2018. URL https://www.businessnewsdaily.com/9276-commercial-drones-business-uses.html.

[10] Pamela Cohn, Alastair Green, Meredith Langstaff, and Melanie Roller. Commercial drones are here: The future of unmanned aerial systems. URL https://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-insights/commercial-drones-are-here-the-future-of-unmanned-aerial-systems.

[11] Joshuah K. Stolaroff, Constantine Samaras, Emma R. O'Neill, Alia Lubers, Alexandra S. Mitchell, and Daniel Ceperley. Author Correction: Energy use and life cycle greenhouse gas emissions of drones for commercial package delivery (Nature Communications (2018) DOI: 10.1038/s41467-017-02411-5). *Nature Communications*, 9(1):1–13, 2018. ISSN 20411723. doi: 10.1038/s41467-018-03457-9. URL http://dx.doi.org/10.1038/s41467-017-02411-5.

[12] Mark W. Mueller and Raffaello D'Andrea. Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 45–52, 2014. ISSN 10504729. doi: 10.1109/ICRA.2014.6906588.

[13] The commercial use of drones. *Computer Law Review International*, 16(3), Jan 2015. doi: 10.9785/cri-2015-0302.

[14] W. F. Phillips, C. E. Hailey, and G. A. Gebert. Review of Attitude Representations Used for Aircraft Kinematics. *Journal of Aircraft*, 40(1):223–223, 2003. ISSN 0021-8669. doi: 10.2514/2.3083. URL http://arc.aiaa.org/doi/10.2514/2.3083.

[15] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor Aerial Vehicles. *Robotics & Automation Magazine, IEEE*, 19(SEPTEMBER):20–23, 2012. ISSN 1070-9932. doi: 10.1109/MRA.2012.2206474.

[16] R. Baxter, N. Hastings, A. Law, and E. J.. Glass. *Aircraft system identification*, volume 39. 2008. ISBN 1563478323.

[17] Alexander Lanzon, Alessandro Freddi, and Sauro Longhi. Flight Control of a Quadrotor Vehicle Subsequent to a Rotor Failure. *Journal of Guidance, Control, and Dynamics*, 37(2):580–591, 2014. ISSN 0731-5090. doi: 10.2514/1.59869. URL http://arc.aiaa.org/doi/10.2514/1.59869.

[18] Sir William Rowan Hamilton LL.D. V.P.R.I.A. F.R.A.S. Xi. on quaternions; or on a new system of imaginaries in algebra. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 33(219):58–60, 1848. doi: 10.1080/14786444808646046.

[19] Emil Fresk and George Nikolakopoulos. Experimental evaluation of a full quaternion based attitude quadrotor controller. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2015-Octob:3864–3869, 2015. ISSN 19460759. doi: 10.1109/ETFA.2015.7301555.

[20] Ayman A. El-Badawy and Mohamed A. Bakr. Quadcopter Aggressive Maneuvers along Singular Configurations: An Energy-Quaternion Based Approach. *Journal of Control Science and Engineering*, 2016, 2016. ISSN 16875257. doi: 10.1155/2016/7324540.

[21] H. Huang, G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin. Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In *2009 IEEE International Conference on Robotics and Automation*, pages 3277–3282, May 2009. doi: 10.1109/ROBOT.2009.5152561.

[22] Mark W. Mueller and Raffaello D'Andrea. Relaxed hover solutions for multicopters: Application to algorithmic redundancy and novel vehicles. *International Journal of Robotics Research*, 35(8):873–889, 2016. ISSN 17413176. doi: 10.1177/0278364915596233.

[23] C. de Visser, J. Mulder, and Q. Chu. Global Nonlinear Aerodynamic Model Identification with Multivariate Splines. *AIAA Atmospheric Flight Mechanics Conference*, (August), 2009. ISSN 00051098. doi: 10.2514/6.2009-5726. URL http://arc.aiaa.org/doi/10.2514/6.2009-5726.

[24] E. de Weerdt, Q.P. Chu, and J.A. Mulder. Neural Network Aerodynamic Model Identification for Aerospace Reconfiguration. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (August), 2005. doi: 10.2514/6.2005-6448. URL http://arc.aiaa.org/doi/10.2514/6.2005-6448.

[25] Daniel Mellinger. *Trajectory generation and control for quadrotors*. PhD thesis, 2012. URL http://ezproxy.net.ucf.edu/login?url=http://search.proquest.com/docview/1018692309?accountid=10003{%}5Cnhttp://sfx.fcla.edu/ucf?url{_}ver=Z39.88-2004{&}rft{_}val{_}fmt=info:ofi/fmt:kev:mtx:dissertation{&}genre=dissertations+{&}+theses{&}sid=ProQ:ProQuest+Dissertations+{&}+.

[26] Wei Dong, Guo Ying Gu, Xiangyang Zhu, and Han Ding. Development of a quadrotor test bed - Modelling, parameter identification, controller design and trajectory generation. *International Journal of Advanced Robotic Systems*, 12, 2015. ISSN 17298814. doi: 10.5772/59618.

[27] Vincenzo Lippiello, Fabio Ruggiero, and Diana Serra. Emergency landing for a quadrotor in case of a propeller failure: A PID approach. *IEEE International Conference on Intelligent Robots and Systems*, pages 4782–4788, 2014. ISSN 21530866. doi: 10.1109/IROS.2014.6943242.

[28] Taeyoung Lee, Melvin Leok, and N. Harris McClamroch. Control of Complex Maneuvers for a Quadrotor UAV using Geometric Methods on SE(3). (i), 2010. ISSN 16107438. doi: 10.1002/asjc.0000. URL http://arxiv.org/abs/1003.2005.

[29] J. Davies, T. Steffen, R. Dixon, R. Goodall, and A. Zolotas. Active versus passive fault tolerant control of a high redundancy actuator. In *2009 European Control Conference (ECC)*, pages 3671–3676, Aug 2009. doi: 10.23919/ECC.2009.7074970.

[30] Y. M. Zhang, A. Chamseddine, C. A. Rabbath, B. W. Gordon, C. Y. Su, S. Rakheja, C. Fulford, J. Apkarian, and P. Gosselin. Development of advanced FDD and FTC techniques with application to an unmanned quadrotor helicopter testbed. *Journal of the Franklin Institute*, 350(9):2396–2422, 2013. ISSN 00160032. doi: 10.1016/j.jfranklin.2013.01.009.

[31] Q P Chu and J A Mulder. Flight Envelope Protection. *Control*, (August), 2010.

[32] Luciano Beffa, Anton Ledergerber, and Raffaello D Andrea. State Estimate Recovery for Autonomous Quadcopters. pages 5704–5710, 2018.

[33] Mark W Mueller, Markus Hehn, and Raffaello D Andrea. A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation. *IEEE Transactions on Robotics*, 31(6): 1294–1310, 2015. doi: 10.1109/TRO.2015.2479878.

[34] Mark W Mueller, Markus Hehn, and Raffaello D Andrea. A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3480–3486, 2013. doi: 10.1109/IROS.2013.6696852.

[35] Markus Hehn and Raffaello D Andrea. Real-Time Trajectory Generation for Quadrocopters. *IEEE Transactions on Robotics*, 31:877–892, 2015. doi: 10.1109/TRO.2015.2432611.

[36] Luis Crespo, Sean Kenny, David Cox, and Daniel Murri. Analysis of Control Strategies for Aircraft Flight Upset Recovery. *AIAA Guidance, Navigation, and Control Conference*, pages 1–31, 2012. ISSN 02555476. doi: 10.2514/6.2012-5026. URL http://arc.aiaa.org/doi/10.2514/6.2012-5026.

[37] paparazzi, November 2018. URL https://wiki.paparazziuav.org/wiki/Main_Page.

[38] Px4, November 2018. URL http://px4.io/.