Loss of Actuator Effectiveness Detection on a Quadrotor

B.A. Strack van Schijndel

October 2019

TU Delft

Loss of Actuator Effectiveness Detection on a Quadrotor

by

B.A. Strack van Schijndel

to obtain the degree of Master of Science
in Aerospace Engineering
at the Delft University of Technology.

Student number:	4100662	
Project duration:	June 2018 – October 2019	
Thesis committee:	Dr. ir. Q.P. Chu,	TU Delft, chair
	Dr. ir. E. Mooij,	TU Delft, external member
	Dr. ir. C.C. de Visser,	TU Delft, supervisor
	S. Sun, MSc	TU Delft, daily supervisor

TU Delft

Preface

Writing this finishes my years in Delft and – putting it a little bit simplistic – Bram 2010 is different than Bram 2019. Maybe the hair loss correlates with an increase in engineering skills?

With regard to this thesis, I would like to thank my supervisors Coen de Visser and Sihao Sun for their support and much needed guidance. That included not only a great deal of valuable feedback, but also feedforward. Their previous work was fundamental to doing this research.

I want to thank my crazy friends and fellow students from the SIM 008 room and STABILO for always laughing at my jokes. I want to especially thank Daan for talking me into STABILO and Gideon for talking me into SIM 008.

Bram Strack van Schijndel
Delft, October 2019

Contents

1

Report outline

This report is structured as follows. Part I contains the research paper as submitted to ICRA 2020, part II contains two appendices to this paper. Appendix A gives a tabular overview of the flight tests that have been done, followed by appendix B that gives an overview of the software build to support this testing. Finally, in part III the preliminary thesis report is included. This report has already been graded as deliverable for the course AE4020.

Research paper

Fast Loss of Effectiveness Detection on a Quadrotor using Onboard Sensors and a Kalman Estimation Approach

B.A. Strack van Schijndel[1], S. Sun[2] and C.C. de Visser[3]

***Abstract*— This paper presents a novel method for fast and robust detection of actuator failures on quadrotors. The proposed algorithm has very little model dependency. A Kalman estimator estimates a stochastic effectiveness factor for every actuator, using only onboard RPM, gyro and accelerometer measurements. Then, a hypothesis test identifies the failed actuator. This algorithm is validated online in real-time, also as part of an active fault tolerant control system. Loss of actuator effectiveness is induced by ejecting the propellers from the motors. The robustness of this algorithm is further investigated offline over a range of parameter settings by replaying real flight data containing 26 propeller ejections. The detection delays are found to be in the 30~130 ms range, without missed detections or false alarms ocurring.**

I. Introduction

Loss of actuator effectiveness (LOE) is one of the many *system failures* [1] that could happen. LOE can happen suddenly due to, for example, propeller faults or other structural failures such as motor arm or assembly breakage. Quadrotors especially lack redundancy in their actuators making actuator faults risky. One way to cope with (single) actuator failures on quadrotors is to sacrifice yaw control and use the remaining rotors to land directly [2], or maintain forward flight [3]. In order to apply an appropriate control strategy, these active fault tolerant control methods require a quick loss of effectiveness detection. Also without sacrificing overall system reliability.

More generally, a fault tolerant control system should prevent simple failures from developing into catastrophic failures. Instead, the system should gracefully degrade by giving up certain functions while maintaining (some) control over other. This is a desirable property of a quadrotor because it improves its safety.

The detection scheme presented in this paper takes RPM, gyro and accelerometer measurements as inputs and dependents on a very limited control effectiveness model. A Kalman estimator estimates a probabilistic LOE-factor for each actuator. Then, a hypothesis test identifies the faulty actuator. In order to arrive at the limited model assumptions had to be made. In order to test these assumptions flight tests were done on a Parrot Bebop 2 quadrotor unmanned air vehicle (UAV), validating the detection scheme.

In summary, the contributions of this research is:

1) Split the actuator fault detection problem in an actuator dynamics part and an actuator effectiveness part (as in INDI control [4]), by utilizing RPM feedback measurements. Then provide a solution to the loss of actuator effectiveness detection problem, by first applying a Kalman estimator, then a hypothesis test. (Section IV)
2) Validate its working in both real-time online and replayed offline environments, at scale across many different flights and varying parameters. (Section V)

II. Related work

Most work on LOE-detection for quadrotors has been on partial (10%-60%) LOE. Recently Nguyen and Hong proposed a method based on a sliding mode Thau observer [5]. Zhong, Zhang et al. proposed a three-stage Kalman filter approach to deal with external disturbances [6]. Avram et al. systematically designed and flight tested a method based on a fault detection and isolation framework [7]. Lu and Van Kampen estimated LOE-factors by taking the pseudoinverse of the effectiveness matrix [8], this method is unfortunately very sensitivity noise. Some methods require model information such as quadrotor inertia or mass [6], [8]–[11], knowledge that is often not readily available in real-world scenarios. Also the availability of inputs such as attitude [6], [8], [9] or position [6], [9] is often assumed.

Regarding other types of multirotor UAV, Frangenberg et al. developed and flight tested a method based on a bank of WLS-estimators on a octorotor [12]. Vey and Lunze investigated an approach based on a Luenberger observer on a hexarotor UAV [13].

There are techniques for detecting other classes of actuator faults that not *necessarily* result in loss of effectiveness. Vibration based methods exploit propeller or motor imbalances, challenging is the isolation to a specific actuator. Jiang et al. [14] propose a method that includes feature extraction using wavelets. These features then serve as an input to train an artificial neural network, after which this network can detect fractured and distorted propellers. Ghalamchi and Mueller have taken steps to detect *and isolate* faults by applying a Fourier transform [15]. Future work is to do the fault detection automatically in real-time. Other authors exploit motor current and/or acoustics measurements [16], [17].

Unrelated to UAVs, Wu et al. [18] applied a Kalman estimator for estimating the changes in control effectiveness of the control surfaces of civil aircraft.

[1]Graduate student, Section Control and Simulation, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands bramsvs@gmail.com

[2]Ph.D. student, Section Control and Simulation, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands s.sun-4@tudelft.nl

[3]Assistant professor, Section Control and Simulation, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands c.c.devisser@tudelft.nl

III. Problem formulation

The aim of this work is to detect loss of actuator effectiveness, by using onboard sensor measurements only, and validate the method on a real quadrotor in real world conditions. As a starting point a fault scenario is assumed.

A. Fault scenario

An actuator fault often does not happen spontaneously. It could be the result of an impact from another flying, static or moving object. One could think of an in-flight collision with another UAV.

In this research, we do not simulate an actuator failure by corrupting the rotor speed setpoint ω_{sp}, but instead we induce a real actuator failure by ejecting the propeller in-flight. This has the following two important implications: 1) the RPM-measurements *resulting* from the rotor speed setpoint are also *not* corrupted, thus can be used as input to the detection algorithm, and 2) the loss of effectiveness of an actuator is sudden and total, so no spin-down lag or percentage reduction of effectiveness.

For these reasons the propeller ejection method is critical to the detection method presented in this research.

B. Propeller ejection method

The propeller ejection method was developed specifically for the Bebop 2, yet it can be generalized. The main idea is to remove the friction in the locking mechanism (Figure 1). Then keep the propeller attached using the aerodynamic forces, and eject it by quickly reducing the motor speed significantly. The assumption is that if the rotational deceleration of the propeller only subject to the aerodynamic torque:

$$(\dot{\omega}_{\text{prop}})_{\text{aero}} = \frac{(\tau_{\text{prop}})_{\text{aero}}}{I_{\text{prop}}}$$

is less than the deceleration of the motor:

$$\dot{\omega}_{\text{motor}} < (\dot{\omega}_{\text{prop}})_{\text{aero}}$$

the propeller will eject.

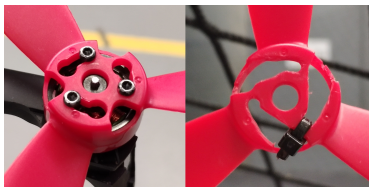


Fig. 1. Nominal locking mechanism (left), frictionless mechanism (right).

The motor speed can be reduced sufficiently fast to cause an ejection using one of the following three methods: 1) by setting a lower altitude setpoint, 2) by giving yaw rate commands, or 3) by adding a sawtooth signal to the actuator setpoint ω_{sp}. On the contrary, an ejection can be prevented by artificially lowering the control gains.

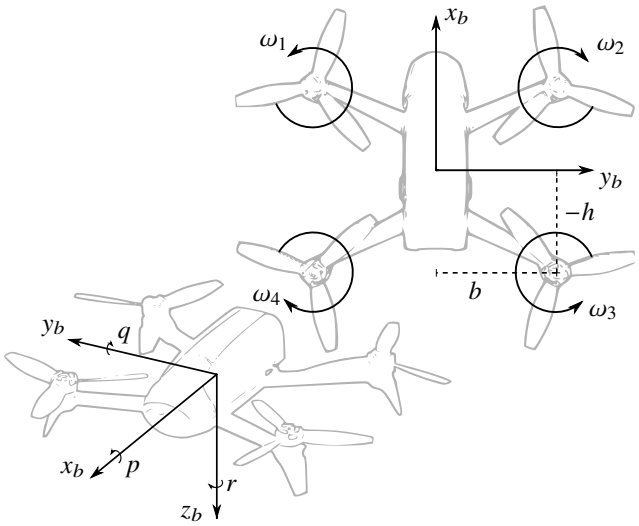


Fig. 2. Geometry of the Parrot Bebop 2 quadrotor with rotor speeds ω_i, geometry b and h, body axes $(\cdot)_b$ and angular rates p, q, r.

C. Control effectiveness model

All quantities are projected onto the body frame. Figure 2 defines its axes, the geometry b, h and the angular velocities of the actuators ω. Assuming in flight, (1) and (2) describe the dynamics of the quadrotor:

$$\boldsymbol{M}_{\text{act}} + \boldsymbol{M}_{\text{aero}} + \boldsymbol{M}_{\text{gyro}} = \boldsymbol{I}_v \dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times \boldsymbol{I}_v \boldsymbol{\Omega} \tag{1}$$

with the summed actuator moments $\boldsymbol{M}_{\text{act}}$, aerodynamic moment $\boldsymbol{M}_{\text{aero}}$, gyroscopic moments $\boldsymbol{M}_{\text{gyro}}$, inertia matrix $\boldsymbol{I}_v$ and angular rate vector $\boldsymbol{\Omega} = \begin{bmatrix} p & q & r \end{bmatrix}$.

Also assuming in flight, the linear acceleration in z-direction can be modeled as:

$$-\sum_{i=1}^{4} (T_{\text{act}})_i + (F_{\text{aero}})_z = m \cdot a_z \tag{2}$$

with actuator thrust force T_{act}, aerodynamic force F_{aero}, quadrotor mass m and proper acceleration a_z. That acceleration is measured by the accelerometer. Considering accelerometer measurements, only the measurements in z-direction are of interest assuming the actuator thrust points in that direction.

The actuator thrust forces are assumed to be linear with respect to the square of the rotor speed:

$$(T_{\text{act}})_i = c_T k_i \omega_i^2 \tag{3}$$

with actuator thrust coefficient c_T and actuator effectiveness scaling factors k_i. A value of $k_i = 1$ corresponds to nominal effectiveness while $k_i = 0$ corresponds to complete loss of effectiveness. These scaling factors k_i will be estimated within the LOE-detection algorithm as explained in Section IV.

The moment generated by the actuator thrust forces can be modeled as:

$$\boldsymbol{M}_{\text{act}} = \sum_{i=1}^{4} \left((\boldsymbol{r}_{\text{act}})_i \times \begin{bmatrix} 0 \\ 0 \\ -(T_{\text{act}})_i \end{bmatrix} \right) \tag{4}$$

with position vectors of the actuators $(r_{\text{act}})_i$:

$$\boldsymbol{r}_1 = \begin{bmatrix} h \\ -b \\ 0 \end{bmatrix},\ \boldsymbol{r}_2 = \begin{bmatrix} h \\ b \\ 0 \end{bmatrix},\ \boldsymbol{r}_3 = \begin{bmatrix} -h \\ b \\ 0 \end{bmatrix},\ \boldsymbol{r}_4 = \begin{bmatrix} -h \\ -b \\ 0 \end{bmatrix}$$

Solving (1) for rotational acceleration gives:

$$\dot{\boldsymbol{\Omega}} = \boldsymbol{I}_v^{-1}(\boldsymbol{\Omega} \times \boldsymbol{I}_v \boldsymbol{\Omega} - \boldsymbol{M}_{\text{act}} - \boldsymbol{M}_{\text{aero}} - \boldsymbol{M}_{\text{gyro}}) \tag{5}$$

This model can be simplified to:

$$\dot{\boldsymbol{\Omega}} = \boldsymbol{I}_v^{-1}(\boldsymbol{M}_{\text{act}}) + \boldsymbol{d} \tag{6}$$

with disturbance $\boldsymbol{d}$ containing $\boldsymbol{M}_{\text{aero}}$, $\boldsymbol{M}_{\text{gyro}}$ the coupling term $\boldsymbol{\Omega} \times \boldsymbol{I}_v \boldsymbol{\Omega}$ and other phenomena such as propeller imbalances, body deformations and center of gravity offsets. We assume that under normal flight conditions $\boldsymbol{d} \ll \boldsymbol{I}_v^{-1}(\boldsymbol{M}_{\text{act}})$.

Likewise, (2) can be solved for a_z and simplified to:

$$a_z = m^{-1} \sum_{i=1}^{4} (T_{\text{act}})_i + d \tag{7}$$

By assuming a diagonal vehicle inertia matrix $\boldsymbol{I}_v$, substituting (4) in (6), (3) in (7) and working out the result in a single matrix equation, the rotational and linear accelerations can be modeled as:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ a_z \end{bmatrix} = c_T \operatorname{diag}\left(\begin{bmatrix} I_x \\ I_y \\ m \end{bmatrix} \right)^{-1} \begin{bmatrix} b & -b & -b & b \\ h & h & -h & -h \\ -1 & -1 & -1 & -1 \end{bmatrix} \operatorname{diag}(\boldsymbol{k}) \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} + \boldsymbol{d} \tag{8}$$

with $\boldsymbol{k} = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix}$, which by defining:

$$G_p := \frac{c_T b}{I_x},\ G_q := \frac{c_T h}{I_y},\ G_{a_z} := \frac{c_T}{m}$$

can be rewritten as:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ a_z \end{bmatrix} = \operatorname{diag}(\boldsymbol{G}) \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & -1 & -1 \end{bmatrix} \operatorname{diag}(\boldsymbol{k}) \boldsymbol{\omega}^2 + \boldsymbol{d} \tag{9}$$

with $\boldsymbol{G} = \begin{bmatrix} G_p & G_q & G_{a_z} \end{bmatrix}$. These control effectiveness parameters $\boldsymbol{G}$ must be estimated. This can be done by using a LMS-estimator [4], or similarly to how in this research the actuator effectiveness factors k_i are estimated, using a Kalman estimator, as explained in the following section. Other stochastic gradient descent methods [19] could also be considered. This simple model serves as basis of the LOE-detection method.

IV. Methodology

A block diagram of the loss of effectiveness (LOE) detection system is given in Figure 3. The following subsections describe each step, starting with the filtering of the inputs. These signals are listed in Table I.

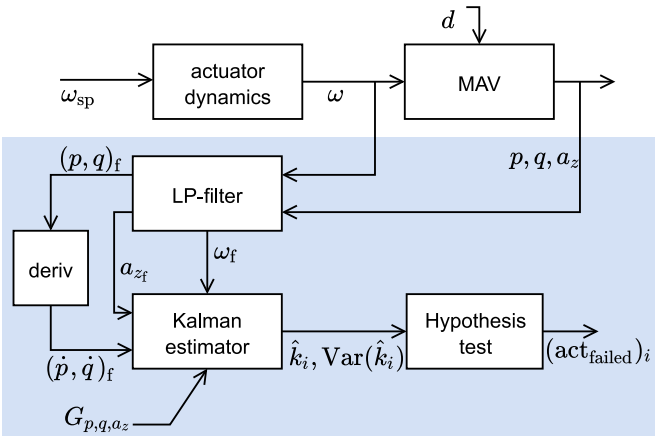


Fig. 3. Overview of the loss of effectiveness detection system (blue). The rotor speeds ω are measured by the ESCs, the rates p, q, and acceleration a_z are measured by the IMU. Only the rates are differentiated, yet all measurements are filtered with the same bandwidth to keep them synchronized.

TABLE I
Signals used for LOE-detection.

Signal	Symbol	Unit	Source	Contamination
Angular rate	p, q, r	rad/s	IMU	Noise, bias
Angular accel.	$\dot{p}$, $\dot{q}$, $\dot{r}$	rad/s^2	IMU + filter	Noise, delay
Proper accel.	a_z	m/s^2	IMU	Noise, bias
Rotor speed	ω	rad/s	ESC	None

A. Filtering of accelerations and motor speed measurements

We have to take the derivative of the measured rotational rates to get the angular accelerations. This is a risky operation because of the noise and vibrations. A second order, low-pass filter effectively filters out these disturbances. The same method as in [4] is applied, being a filter of the form:

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

with parameters $\omega_n = 50$ rad/s $\zeta = 0.55$. Note that by inspecting the raw sensor data — sampled at 500 Hz — it was observed that most vibrations are in the ~50-200 Hz range, corresponding to a minimum and maximum rotor speed of the Bebop 2 of ~3000 RPM, resp. ~12000 RPM. As in [4], this same filtering is also applied to the rotor speed ω and accelerometer a_z measurements in order to keep them synchronized with the gyro measurements.

The angular accelerations are simply approximated by taking the backward difference quotient of the filtered rotational rates:

$$(\dot{\boldsymbol{\Omega}}_{\text{f}})_k = \frac{(\boldsymbol{\Omega}_{\text{f}})_k - (\boldsymbol{\Omega}_{\text{f}})_{k-1}}{\Delta t} \tag{10}$$

this approach is similar to a washout filter [20].

B. Kalman estimator

We propose a Kalman estimator, its state vector:

$$\boldsymbol{x} = \hat{\boldsymbol{k}} = \begin{bmatrix} \hat{k}_1 & \hat{k}_2 & \hat{k}_3 & \hat{k}_4 \end{bmatrix}^{\mathsf{T}} \tag{11}$$

contains the estimate of the actuator effectiveness scaling factors k_i as defined in Section III. The state transition is modeled as a random walk:

$$\dot{\boldsymbol{x}} = \boldsymbol{0}$$

The control effectiveness model (9) can be rewritten into the following observation model:

$$\begin{aligned} \boldsymbol{z} &= \boldsymbol{H}(\boldsymbol{\omega})\boldsymbol{x} \\ &= \operatorname{diag}(\boldsymbol{G}) \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & -1 & -1 \end{bmatrix} \operatorname{diag}(\boldsymbol{\omega}^2)\boldsymbol{x} \\ &= \begin{bmatrix} G_p\omega_1^2 & -G_p\omega_2^2 & -G_p\omega_3^2 & G_p\omega_4^2 \\ G_q\omega_1^2 & G_q\omega_2^2 & -G_q\omega_3^2 & -G_q\omega_4^2 \\ -G_{a_z}\omega_1^2 & -G_{a_z}\omega_2^2 & -G_{a_z}\omega_3^2 & -G_{a_z}\omega_4^2 \end{bmatrix} \boldsymbol{x} \end{aligned} \tag{12}$$

This model is linear in the state parameters $\boldsymbol{x}$. The rotor speeds $\boldsymbol{\omega}$ are measured. Its observation vector is:

$$\boldsymbol{z} = \begin{bmatrix} \dot{p} & \dot{q} & a_z \end{bmatrix}^\mathsf{T} \tag{13}$$

The whole Kalman estimation algorithm is then defined by algorithm 1.

Input : $\boldsymbol{x}_{k-1}$, $\boldsymbol{P}_{k-1}$, $\boldsymbol{z}_k$

$\boldsymbol{P}_{k|k-1} = \boldsymbol{P}_{k-1} + \boldsymbol{Q}$ $\qquad$ $\boldsymbol{K} = \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^\mathsf{T}/\boldsymbol{S}$

$\boldsymbol{y} = \boldsymbol{z}_k - \boldsymbol{H}_k\boldsymbol{x}_{k-1}$ $\qquad$ $\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \boldsymbol{K}\boldsymbol{y}$

$\boldsymbol{S} = \boldsymbol{R} + \boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^\mathsf{T}$ $\qquad$ $\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}_k)\boldsymbol{P}_{k|k-1}$

return $\boldsymbol{x}_k$, $\boldsymbol{P}_{k|k}$

Algorithm 1: KALMAN ESTIMATOR

To increase the robustness of the algorithm the scaling factors k_i were bound to $[0, 1.5]$.

C. Hypothesis test

The estimated scaling factors $\hat{k}_i$ are stochastic variables of which the variances can be found in the diagonal of the covariance matrix:

$$\operatorname{diag}(\boldsymbol{P}) = \operatorname{Var}(\boldsymbol{x}) = \operatorname{Var}(\{\hat{k}_1 \;\; \hat{k}_2 \;\; \hat{k}_3 \;\; \hat{k}_4\}) \tag{14}$$

Now the failure probability per actuator i can be calculated:

$$(P_{\text{fail}})_i = P(\hat{k}_i < k_{\text{thres}})$$

with the estimated scaling factor $\hat{k}_i$, by assuming a parameter k_{thres} and by making the same Gaussian distribution assumption as done in designing the Kalman estimator, implying:

$$P(\hat{k}_i < k_{\text{thres}}) = 1 - \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{k_{\text{thres}} - \hat{k}_i}{\sigma_{k\,i}^2\sqrt{2}}\right)\right] \tag{15}$$

with variance $\sigma_{k\,i}^2 = \operatorname{Var}(\hat{k}_i)$ and $\operatorname{erf}(\cdot)$ being the Gauss error function.

Given the failure probability, a decision can be made on actuator failure:

$$(\text{act}_{\text{failed}})_i = \begin{cases} 1 & \text{if } (P_{\text{fail}})_i > (P_{\text{fail}})_{\text{thres}} \\ 0 & \text{if } (P_{\text{fail}})_i < (P_{\text{fail}})_{\text{thres}} \end{cases} \tag{16}$$

The selection of these thresholds will be discussed in the following section.

V. EXPERIMENTS AND RESULTS

The LOE-detection algorithm was validated *offline* as well as *online* in real-time as part of an active fault tolerant control system (AFTC). Figure 4 gives an overview of this process.

The Bebop 2 quadrotor used in the experiments runs PX4 [21]. The fault detection algorithm is embedded in PX4 via a C++ library generated with Simulink Coder.

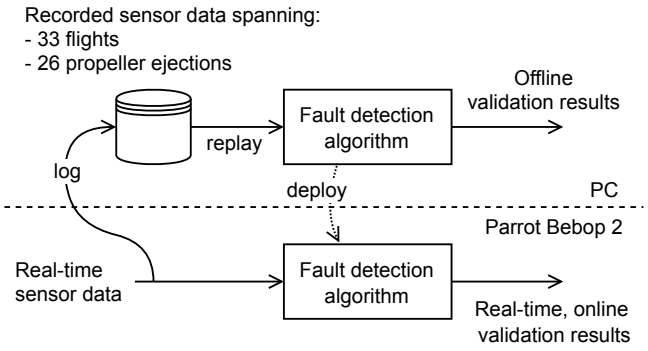


Fig. 4. The process used in validating the fault detection algorithm. The exact same algorithm is used in both the offline (PC) environment as in the online real-time environment (Bebop 2). All recorded sensor data is run through the algorithm with 19 different parameter sets.

In the following subsections, first of all, a close-up of a single flight is presented, showing the inner workings of the system (V-A). Then, the offline replay lets us vary parameters while giving real sensor data as inputs (V-B). We conclude with the key performance metrics (V-C) and a brief discussion on ground contact (V-D).

A. Close up of single flight

We study flight 30. In this flight the LOE-detector served as input to the AFTC, thus together preventing catastrophic failure. The applied parameters are given in Table II, Figure 5 shows some of the key states of the whole, short, flight. The frames in Figure 6 show the moment propeller number three gets ejected and the moment this failure gets detected.

Zooming in on both the failure itself, as well as the detection event, Figure 7 shows the accelerations that serve as the observations 13 to the Kalman estimator. The measured and filtered rotor speeds are shown in Figure 8. Then moving to the Kalman estimator itself, in Figure 9 the Kalman state vector (11), its variances (14) and failure probabilities (15) are shown. Figure 10 shows the probability density function (PDF) of the scaling factors k_i at the indicated timestamps. Following the failure, a clear and distinct shift of the PDF can be observed. Note that when the excitation of the system is low, the variances are trending upward. This is because the observation model (12) is an underdetermined system.

B. Sensitivity analysis across multiple flights

The purpose of this analysis is two-fold: 1) to investigate the robustness of the output of the algorithm against a variety of flight dynamics and uncertainties that occur during real flights, and 2) to identify the sensitivity of the parameters.

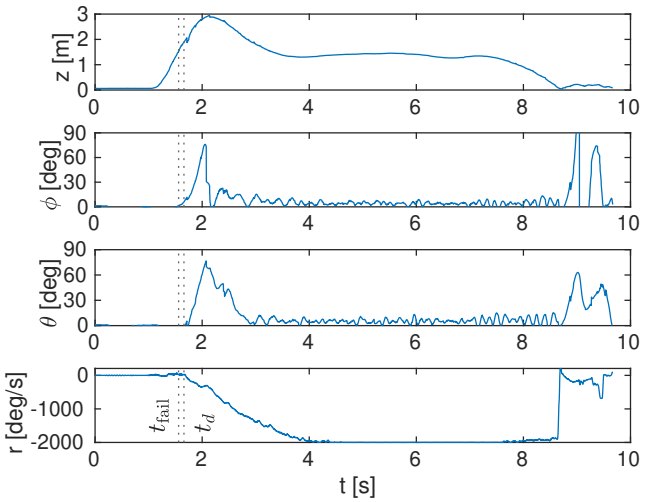


Fig. 5. Key states of flight 30. From top to bottom: altitude z, roll ϕ, pitch θ, yaw rate r. Actuator 3 fails at $t_{\text{fail}} = 1.56$, detection is at $t_d = 1.66$. After detection and during increased pitch and roll angles, apogee is reached. Then, the vehicle levels off and reaches a very impressive yaw rate of at least −2000 deg/s, hitting the angular rate limit of the MPU-6050 gyroscope onboard the Bebop 2 MAV.

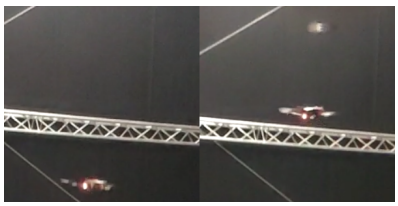


Fig. 6. Video frames of the quadrotor at t_{fail} (left) and the time of detection t_d (right). The ejected propeller is visible in the right frame, in the top right corner.

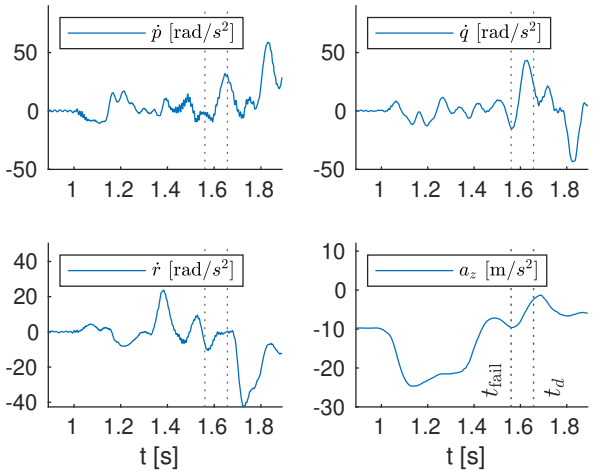


Fig. 7. The rotational and linear accelerations, $\dot{p}$, $\dot{q}$ and a_z are the observations to the Kalman estimator. t_{fail} and t_d are indicated with the dotted lines. After failure, a positive roll and pitching acceleration can be observed, as well as a reduction in acceleration and an increasingly negative acceleration around the yaw-axis.

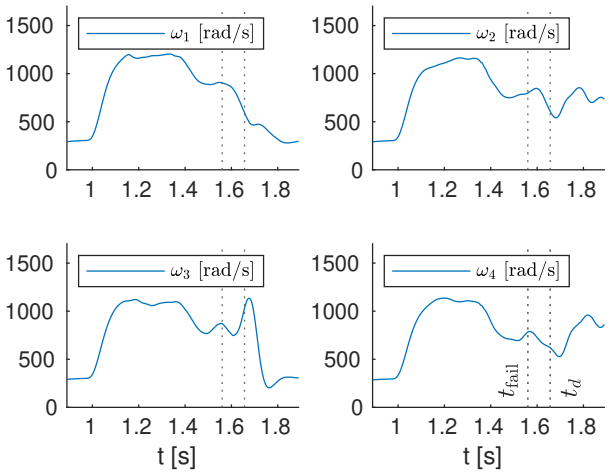


Fig. 8. Motor speed measurements during a propeller ejection. In this case, number 3 is the failed one. t_{fail} and t_d are indicated with the dotted lines. At detection its setpoint is set to idle by the controller. Actuator 1 is opposite to actuator 3, thus going to idle already *before* detection in an attempt to reach the commanded roll and pitch rates.

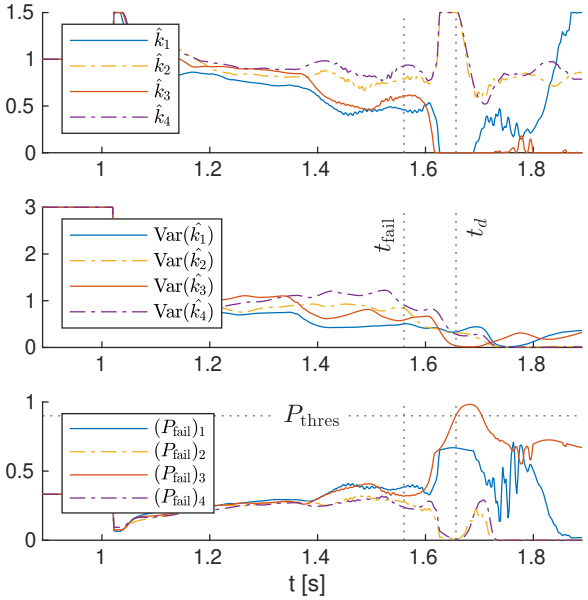


Fig. 9. Scaling factors $\hat{k}_i$ with variances $\text{Var}(\hat{k}_i)$ are the Kalman estimator outputs. The failure probabilities $(P_{\text{fail}})_i = P(\hat{k}_i < k_{\text{thres}})$ with $k_{\text{thres}} = 0.25$ is the generated signal that triggers the detection. $(P_{\text{fail}})_{\text{thres}} = 0.9$ gives the indicated fault detection time $t_d = 1.66$ s.

The sensitivity analysis was done by replaying the algorithm with recorded flight data and varying *one* of the parameters, keeping the others as in Table II. The recorded flight data contains 33 flights, of which 26 experienced a failure per the method described in Section III. Some of these flights were done inside a wind tunnel experiencing wind speeds up to 10 m/s.

Figure 11 presents the detection delays for varying parameters. The control effectiveness parameters $\boldsymbol{G}$ vary a lot between airframes but can be identified automatically [4]. They were varied by ±20% and show little sensitivity, therefore they are not considered critical. Q, $(P_{\text{fail}})_{\text{thres}}$ and

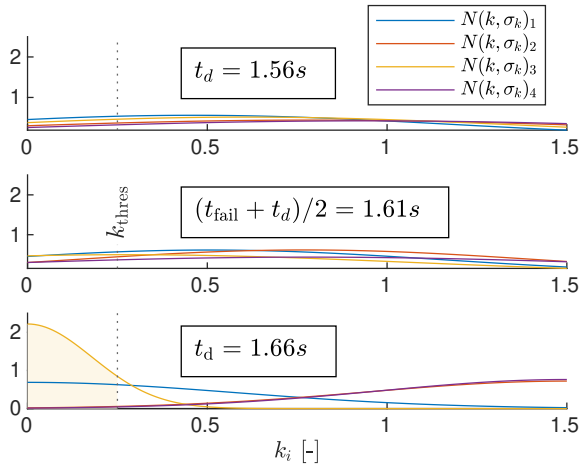


Fig. 10. The probability density functions of the actuator effectiveness scaling factors k_i at t_{fail} (top), $(t_{\text{fail}} + t_d)/2$ (middle) and t_d (bottom). With k_{thres} indicated at $k = 0.25$. The shaded area is $(P_{\text{fail}})_3 = (P_{\text{fail}})_{\text{thres}} = 0.9$.

TABLE II
USED PARAMETERS AND VALUES.

Parameter	Symbol	Value
Roll control effectiveness	G_p	100×10^{-6}
Pitch control effectiveness	G_q	100×10^{-6}
Z-acceleration effectiveness	G_{a_z}	5×10^{-6}
Process noise	Q	0.1
Measurement noise	R	1
Scaling factor failure threshold	k_{thres}	0.25
Failure probability threshold	$(P_{\text{fail}})_{\text{thres}}$	0.9
Step size	ΔT	0.02

k_{thres} should vary less across airframes, are design variables currently set manually, thus are considered more critical.

Also little sensitivity was observed in the false alarm and missed detection rates against changes in parameter settings. Though, high detection delays t_d tend to correlate with an increasing probability in missed detections.

C. Final results and real-time performance

The following performance metrics are of interest in evaluating fault detection systems [22]: 1) detection delay, 2) false alarm rate, and 3) missed detection rate. Note that for our fault scenario — a sudden failure — the *detection delay* is of special importance because a quadrotor's lack of redundancy requires fast control reconfiguration to prevent upset conditions. Figure 12 shows the detection delays, no missed detections and no false alarms occurred, with the parameters as in Table II. The 95% confidence bounds of the detection delay is $[28, 132]$ ms.

On the Bebop 2 UAV, equipped with a ARM Cortex-A9 processor, one step of the whole LOE-detection algorithm takes 18 µs. Thus, the total running time is 500 Hz · 18 µs = 9 ms/sec.

D. Discussion on ground contact

The models in Section III all assume the UAV is in flight. Ground contact will trigger false alarms because the perceived effectiveness of an actuator will be zero. Therefore, we implemented a simple take-off detector that activates the fault detection algorithm after a certain thrust level is reached. Although this works sufficiently in a research setting, more work on land detection is needed to make the system robust against repeated take-offs and landings.

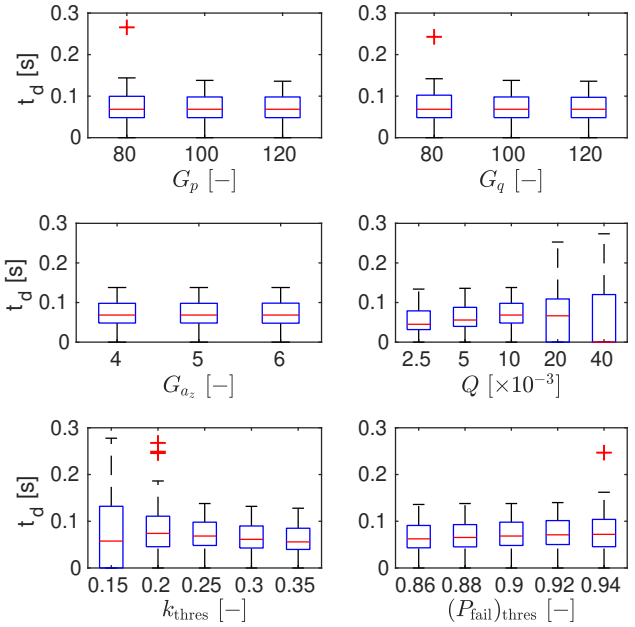


Fig. 11. Box plots of the detection delays for varying parameters. The box plots contain the detection delays for the set of 26 flights experiencing an actuator failure. Red crosses indicate the outliers.

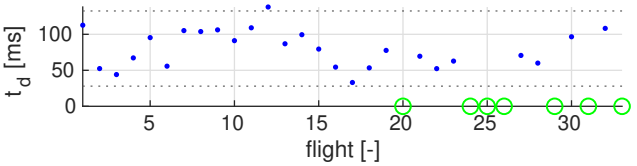


Fig. 12. The detection delay t_d of all 26 flights experiencing a propeller ejection. The flights not experiencing a propeller ejection and without any false alarms are indicated as a green circle. The dotted lines indicate the 95% confidence interval of the detection delays.

VI. FUTURE WORK

Future work should be aimed at maximizing the efficiency and reliability of the method, especially while integrated within the UAV-system:

- Investigate possibilities for simplifying the algorithm, ideally reducing the parameter count.
- Find better methods for dealing with ground contact.
- Evaluate the method under a wider range of flight conditions, properly taking into account the aerodynamic moment occurring during high-speed flight.

ACKNOWLEDGMENTS

We would like to thank the TU Delft MAVLab for making the drone testing facility available to us, and the PX4-community for building excellent open-source flight control software.

REFERENCES

[1] C. M. Belcastro, D. H. Klyde, M. J. Logan, R. L. Newman, and J. V. Foster, "Experimental Flight Testing for Assessing the Safety of Unmanned Aircraft System Safety-Critical Operations," in *17th AIAA Aviation Technology, Integration, and Operations Conference*, Denver, Colorado: American Institute of Aeronautics and Astronautics, Jun. 5, 2017, ISBN: 978-1-62410-508-1. DOI: 10.2514/6.2017-3274.

[2] M. W. Mueller and R. D'Andrea, "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 45–52. DOI: 10.1109/ICRA.2014.6906588.

[3] S. Sun, L. Sijbers, X. Wang, and C. de Visser, "High-Speed Flight of Quadrotor Despite Loss of Single Rotor," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3201–3207, Oct. 2018, ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2851028.

[4] E. J. Smeur, Q. P. Chu, and G. C. de Croon, "Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Aerial Vehicles," in *AIAA Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.2016-1390.

[5] N. P. Nguyen and S. K. Hong, "Sliding Mode Thau Observer for Actuator Fault Diagnosis of Quadcopter UAVs," *Applied Sciences*, vol. 8, no. 10, p. 1893, Oct. 11, 2018, ISSN: 2076-3417. DOI: 10.3390/app8101893.

[6] Y. Zhong, Y. Zhang, W. Zhang, J. Zuo, and H. Zhan, "Robust Actuator Fault Detection and Diagnosis for a Quadrotor UAV With External Disturbances," *IEEE Access*, vol. 6, pp. 48 169–48 180, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2867574.

[7] R. C. Avram, X. Zhang, and J. Muse, "Quadrotor Actuator Fault Diagnosis and Accommodation Using Nonlinear Adaptive Estimators," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 6, pp. 2219–2226, Nov. 2017, ISSN: 1063-6536. DOI: 10.1109/TCST.2016.2640941.

[8] P. Lu and E.-J. van Kampen, "Active fault-tolerant control for quadrotors subjected to a complete rotor failure," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany: IEEE, Sep. 2015, pp. 4698–4703, ISBN: 978-1-4799-9994-1. DOI: 10.1109/IROS.2015.7354046.

[9] A. Freddi, S. Longhi, and A. Monteriù, "Actuator fault detection system for a mini-quadrotor," in *2010 IEEE International Symposium on Industrial Electronics*, Jul. 2010, pp. 2055–2060. DOI: 10.1109/ISIE.2010.5637750.

[10] R. C. Avram, X. Zhang, and J. Muse, "Quadrotor Actuator Fault Diagnosis and Accommodation Using Nonlinear Adaptive Estimators," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 6, pp. 2219–2226, Nov. 2017, ISSN: 1063-6536. DOI: 10.1109/TCST.2016.2640941.

[11] A. Hasan and T. A. Johansen, "Model-Based Actuator Fault Diagnosis in Multirotor UAVs," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, Dallas, TX: IEEE, Jun. 2018, pp. 1017–1024, ISBN: 978-1-5386-1354-2. DOI: 10.1109/ICUAS.2018.8453420.

[12] M. Frangenberg, J. Stephan, and W. Fichter, "Fast Actuator Fault Detection and Reconfiguration for Multicopters," in *AIAA Guidance, Navigation, and Control Conference*, ser. AIAA SciTech Forum, 0 vols., American Institute of Aeronautics and Astronautics, Jan. 2, 2015. DOI: 10.2514/6.2015-1766.

[13] D. Vey and J. Lunze, "Experimental evaluation of an active fault-tolerant control scheme for multirotor UAVs," in *2016 3rd Conference on Control and Fault-Tolerant Systems (SysTol)*, Sep. 2016, pp. 125–132. DOI: 10.1109/SYSTOL.2016.7739739.

[14] Y. Jiang, Z. Zhiyao, L. Haoxiang, and Q. Quan, "Fault detection and identification for quadrotor based on airframe vibration signals: A data-driven method," in *2015 34th Chinese Control Conference (CCC)*, Jul. 2015, pp. 6356–6361. DOI: 10.1109/ChiCC.2015.7260639.

[15] B. Ghalamchi and M. Mueller, "Vibration-Based Propeller Fault Diagnosis for Multicopters," *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, Jun. 2018.

[16] J. M. Brown, J. A. Coffey, D. Harvey, and J. M. Thayer, "Characterization and Prognosis of Multirotor Failures," in *Structural Health Monitoring and Damage Detection, Volume 7*, C. Niezrecki, Ed., ser. Conference Proceedings of the Society for Experimental Mechanics Series, Springer International Publishing, 2015, pp. 157–173, ISBN: 978-3-319-15230-1.

[17] P. Misra, G. Kandaswamy, P. Mohapatra, K. Kumar, and P. Balamuralidhar, "Structural Health Monitoring of Multi-Rotor Micro Aerial Vehicles," in *Proceedings of the 4th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications - DroNet'18*, Munich, Germany: ACM Press, 2018, pp. 21–26, ISBN: 978-1-4503-5839-2. DOI: 10.1145/3213526.3213531.

[18] N. E. Wu, Y. Zhang, and K. Zhou, "Control effectiveness estimation using an adaptive Kalman estimator," in *Proceedings of the 1998 IEEE International Symposium on Intelligent Control (ISIC) Held Jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA) Intell*, Sep. 1998, pp. 181–186. DOI: 10.1109/ISIC.1998.713657.

[19] D. P. Mandic, S. Kanna, and A. G. Constantinides, "On the Intrinsic Relationship Between the Least Mean Square and Kalman Filters [Lecture Notes]," *IEEE Signal Processing Magazine*, vol. 32, no. 6, pp. 117–122, Nov. 2015, ISSN: 1053-5888. DOI: 10.1109/MSP.2015.2461733.

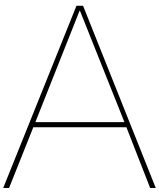
[20] B. J. Bacon, A. J. Ostroff, and S. M. Joshi, "Reconfigurable NDI controller using inertial sensor failure detection amp;amp; isolation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 37, no. 4, pp. 1373–1383, Oct. 2001, ISSN: 0018-9251. DOI: 10.1109/7.976972.

[21] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 6235–6240. DOI: 10.1109/ICRA.2015.7140074.

[22] R. J. Patton, "FAULT-TOLERANT CONTROL SYSTEMS: THE 1997 SITUATION," p. 22,

II

Appendices



Flight logs

Table A.1 contains an overview of the 33 logged flights. These flights are logged and stored per the method described in Figure 4 of the research paper. The full dataset is available upon request.

Table A.1: Some metadata of the flight logs.

Nr.	Date, time [UTC]	Failed rotor	Fail time [s]	Wind speed [m/s]	Notes
1	2019-02-01, 11:43:10	2	12.39	0.0	nice eject
2	2019-02-01, 13:06:06	2	100.2	0.0	uncommanded eject above mattress
3	2019-02-01, 13:11:20	2	127.1	0.0	noisy, delayed
4	2019-02-01, 13:17:31	2	75.21	0.0	eject caused by lower altitude setting
5	2019-02-01, 13:20:36	2	14.54	0.0	
6	2019-02-01, 13:24:30	2	35.69	9.8	
7	2019-02-01, 13:27:41	2	69.3	9.0	slow eject
8	2019-02-01, 13:40:22	2	24.79	10.0	eject caused by yaw input, slow (detection?)
9	2019-02-01, 14:05:13	2	47.71	0.0	fail + FTC
10	2019-02-01, 14:09:17	2	39.59	0.0	
11	2019-02-01, 14:17:31	2	62.59	5.0	
12	2019-02-01, 14:33:23	2	40.27	4.0	
13	2019-02-01, 14:36:17	2	23.94	2.0	
14	2019-02-01, 14:38:41	2	56.15	2.0	
15	2019-02-01, 15:10:41	2	25.87	0.0	
16	2019-08-07, 16:38:17	3	1.7	0.0	uncommanded, during takeoff
17	2019-08-07, 16:39:55	3	3.885	0.0	uncommanded, from hover
18	2019-08-07, 16:46:38	3	1.53	0.0	partial recovery
19	2019-08-07, 16:48:49	3	1.58	0.0	partial recovery
20	2019-08-08, 15:54:02	n/a	n/a	0.0	no failure
21	2019-08-08, 15:55:14	3	1.49	0.0	during takeoff
22	2019-08-08, 15:57:39	3	2.375	0.0	partial recovery
23	2019-08-08, 15:58:58	3	1.46	0.0	during takeoff
24	2019-08-08, 16:02:36	n/a	n/a	0.0	
25	2019-08-08, 16:09:26	n/a	n/a	0.0	hard crash
26	2019-08-08, 16:12:41	n/a	n/a	0.0	fail override, spinning
27	2019-08-09, 17:23:05	4	2.09	0.0	during takeoff
28	2019-08-09, 17:28:36	3	2.17	0.0	during late takeoff
29	2019-08-09, 17:30:44	n/a	n/a	0.0	lost tracking
30	2019-08-09, 17:48:56	3	1.56	0.0	during late takeoff, good recovery
31	2019-08-09, 17:50:39	n/a	n/a	0.0	lost tracking
32	2019-08-09, 17:53:51	1	1.01	0.0	during takeoff
33	2019-08-09, 17:55:04	n/a	n/a	0.0	just takeoff and landing

B

Software overview

Generally, flight testing new flight control algorithms is costly. A big component of this cost is the time it takes to implement such algorithms on a UAV. Implementing these algorithms often requires extensive programming in low-level languages such as C, as well as extensive knowledge about autopilot software. These skills are difficult to acquire and apply, taking up a large chunk of available project time. All while the algorithm is often already implemented in a higher level language such as Simulink®, Python or known 'on paper'. The software designed for this thesis project attempts to help streamline this process of implementation and testing, now and in the future.

It is good practice that software documentation should be in the software repository. This keeps it in sync with the software and is available to who need it the most. In order to give an overview of the software in this report, Figure B.1 shows the dependency graph of the Python modules. The nodes reference Python modules and packages. The arrows indicate the direction of an 'import'. The `common`-package (yellow) serves as a shared library for the other packages that perform higher level tasks. The `simulink_model`, `px4`, `qgroundcontrol`, `mavros` and `matlablib` modules are Python-wrappers around the software components they are named after.

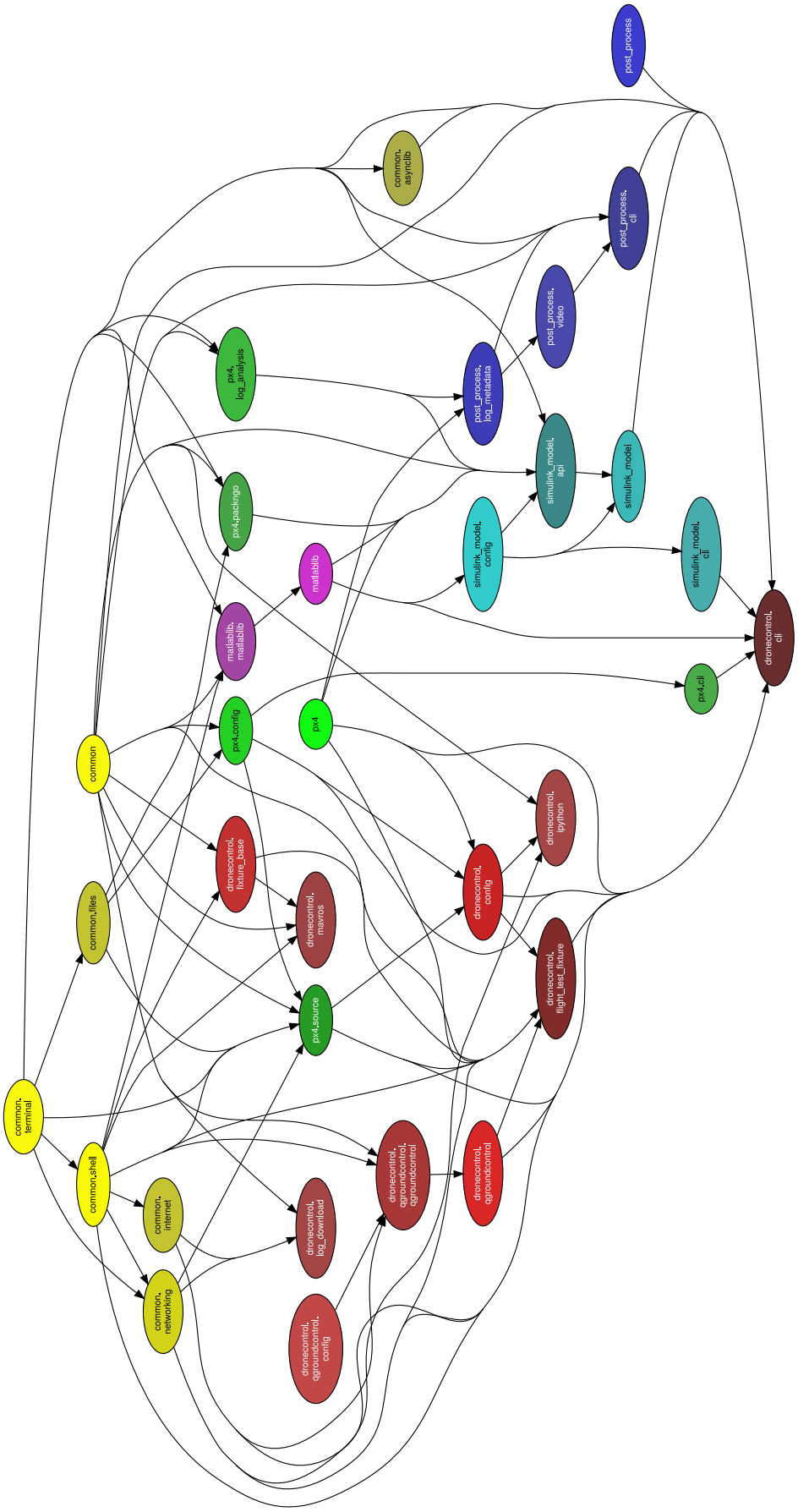


Figure B.1: Dependency graph of the Python modules in the software repository.

III

Preliminary thesis report

1 Introduction

This introduction aims to give the reader a clear idea about why this research is done, its general approach undertaking it and how to further navigate this report.

1.1. Research motivation

The European Aviation Safety Agency (EASA) reports an upward trend in safety risk associated with Unmanned Aerial Systems (UAS) [1]. Also, in that report they identify *system failures* as one of the main safety risks in UAS operations. These operations span the professional, as well as the recreational domains. Especially in the recreational domain quadrotors are becoming increasingly popular because of affordability and ease of use.

Loss of actuator effectiveness (LOE)[1] is one of the many *system failures* that could happen [2]. Especially quadrotors lack redundancy in their actuators, thus making actuator faults risky. One way to cope with (single) actuator failures on quadrotors is to sacrifice yaw control and use the remaining rotors to land directly [3], or to maintain forward flight [4].

Loss of actuator effectiveness can happen suddenly due to, for example because of propeller faults, electronics faults or structural failures such as motor arm or assembly breakage. All this requires a quick loss of effectiveness detection, resulting in a switch to the fault tolerant controller. The main goal of this work is to find such a fast method. This will result in an Active Fault Tolerant Control (AFTC) system.

More generally, a fault tolerant control system should prevent simple failures from developing into catastrophic failures. Instead, the system should gracefully degrade by giving up certain functions while maintaining (some) control over other. This is a desirable property of a quadrotor because it improves safety.

1.2. Report outline

This preliminary reports consists of two main parts. Firstly, a literature review in Chapter 2, then the project design in Chapter 3. Also a brief overview is given on flight control software in Appendix A.

[1] The report [2] refers to this as 'Loss of Control Effectiveness (Rotors)'.

2

Literature review

The aim of this chapter is to review the current (1H 2019) state of the art in loss of actuator effectiveness (LOE) detection methods for multirotor (MR) micro air vehicles (MAVs).

The outline of this chapter is as follows. After this introduction, the key literature in this field is summarized in a table format in Section 2.1. This summary forms the core of this review. It is followed by a critical analysis (Section 2.2), conclusions and recommendations for further research (Section 2.3).

Research methodology A search was done in the following online databases:

- Google Search `https://www.google.com/`
- Semantic Scholar `https://www.semanticscholar.org/`
- ResearchGate `https://www.researchgate.net/`
- Scopus `https://www.scopus.com/`

Keywords included (but were not limited to): 'actuator fault detection', 'quadrotor', 'multirotor', 'loss of effectiveness'.

There found resources where used as a starting point. If seemed compelling, its references where traversed back- and/or forward. ResearchGate was especially helpful for forward reference searching. A set of quality literature is selected that encompasses the state of the art in this field.

Historical context While most research done on multirotor MAVs specifically is relatively young, research done on fault tolerant control systems is a lot older. Patton [5] describes the state of the art in 1997. Later, in 2010, Edwards et al. [6] bundle the state of art in fault tolerant *flight* control systems, work motivated by the desire to make aviation safer.

Related work There are techniques for detecting other classes of actuator faults that not *necessarily* result in loss of effectiveness.

Vibration based methods exploit propeller or motor imbalances, challenging is the isolation to a specific actuator. Jiang et al. [7] propose a method that includes feature extraction using wavelets. These features then serve as an input to train an artificial neural network. After which this network can detect fractured and distorted propellers. Vibration detection *and isolation* by Fourier transform is proposed by Ghalamchi and Mueller [8]. No work was carried out yet to do the fault detection automatically in real-time.

Other authors exploit motor current and/or acoustics measurements [9], [10].

2.1. Overview of LOE-detection methods

The main methods for detecting LOE on multirotor (MR) MAVs are summarized in Table 2.1.

Table 2.1: Overview of methods, listing key properties. The order is chronological, the first method [11] is published in 2010, the other eight in the 2015-2019 period. The meaning of the symbols are: learning rate λ, adaptation rate μ, control effectiveness matrix $\mathbf{G}$, filter cutoff frequency f_0, LOE-threshold l_T, residual 'ratio' δ, residual threshold r_T, z-acceleration measurement a_z and vertical inertial velocity v_z.

	Context			**Design choices**				**Results**	
Author	MR cfg.	Fault case(s)	Environment	Sensor inputs[1]	(Model) params	Estimator or observer	Detection signal	Detection delay[2]	Remarks
Freddi et al. [11]	4+	unresponsive 'stuck'	simulated	$[\phi\ \theta\ \psi]$, $[x\ y\ z]$	$[l_x\ l_y\ l_z\ m]$, δ, r_T	Thau	residuals	$\sim 2\ s$	can only isolate to diagonal
Frangenberg et al. [12]	8	motor fail	on/off-line	$[p\ q\ r\ a_z]$	$\mathbf{G}$, f_0, l_T	WLS, bank	low pass-filtered LOE	$1.55\ s$	
Lu and Van Kampen [13]	4+	motor fail	simulated	$[p\ q\ r\ a_z]$, $[\phi\ \theta\ \psi]$	$[l_x\ l_y\ l_z\ m]$, $[a_0\ a_1\ a_2]$, l_T	pseudoinverse	LOE	$0.03\ s$	not dealing with noise/disturbances
Vey and Lunze [14]	6	motor fail	online[4]	$[\phi\ \theta\ \psi]$, z	$\mathbf{G}$, r_T	Luenberger, bank	residuals	$\sim 0.3\ s$	
Smeur et al. [15]	4x	*no fault*[3]	online	$[p\ q\ r\ a_z]$	$\mathbf{G}$, ω_0, ζ, μ, I_r	LMS	*no detection*, 20 control eff. params	n/a	part of adaptive INDI
Avram et al. [16], [17]	4x	6-10% LOE	online[4]	$[p\ q\ r\ v_z]$	$[l_x\ l_y\ l_z\ m]$, c_d, λ	nonlinear estimators	residual and adaptive threshold	$\sim 0.3\ s$	systematic design by FDI framework
Hasan and Johansen [18]	4+	10% LOE	simulated	$[p\ q\ r]$, $[x\ y\ z]$	$[l_x\ l_y\ l_z\ m]$, l_T, etc.[5]	Thau, Kalman	n/a	$\sim 2\ s$	
Zhong, Zhang et al. [19]	4x	20-60% LOE	simulated	$[\phi\ \theta\ \psi]$, $[x\ y\ z]$	$[l_x\ l_y\ l_z\ m]$, etc.[5]	three-stage Kalman	n/a	$< 1\ s$	only fault estimation, linearized model
Nguyen and Hong [20]	4+	30% LOE	fixed test stand	$[\dot\phi\ \dot\theta\ \dot\psi]$, $[\phi\ \theta\ \psi]$	$[l_x\ l_y\ l_z\ m]$, λ, etc.[5]	sliding mode Thau	n/a	$> 1\ s$	only fault estimation

[1] All methods also require a control input $\bar{u}$, being force/moment or actuator setpoints. The inputs are grouped by its sources: IMU, attitude, position estimators.
[2] If the method has a separate *detection* and *isolation* delay, its *isolation* delay is listed in this column.
[3] The goal of this method was not to perform fault detection, yet it does demonstrate online control effectiveness estimation.
[4] For the video of the experiment performed in [14] see `https://www.youtube.com/watch?v=_MB4yDbz7pw`, for [17] see `https://www.youtube.com/watch?v=Acz4ZZiq5ek`
[5] It is often unclear which sensor signals and model or design parameters are exactly needed.

Methods not included in comparison There are more methods published such [21]–[23], that are not included in Table 2.1 because they were superseded by methods that were included, or the necessary information could not be extracted from the publication.

2.2. Major trends

Some simple observations can be made by looking at Table 2.1:

- Many different assumed fault cases
- Large range in detection delay
- Many different assumptions and design choices made
- Often no or limited real-world validation

This section aims to explain and comment on these observations.

Fault scenarios In the articles, no real arguments are given why certain fault scenarios are assumed. All faults are created by corrupting the control input signal, thus simulating a LOE. Questionable is if a 'clean' partial LOE is a realistic scenario. For example, it could be that (heavy) vibrations accompany this fault, caused by propeller imbalance because an external impact preceded the LOE-event.

MR configuration and detection delay Most experimental studies were performed on a multirotor with more than four actuators [12], [14], thus providing redundancy and allowing long detection delays and full LOE of an actuator. The only flight-tested method on a quadrotor [17], thus not having redundancy in its actuators, limits the LOE to just 10%, probably to keep flying.

Different assumptions Many different, sometimes implicit, assumptions are made, such as:

- Availability of IMU, attitude, or position measurements
- Linearized model [19], [24]
- Absent noise, disturbances [13]

Also, all methods assume *global* fault detection, usually done by only looking at the (attitude) controller input and outputs (rotor speed setpoints). Maybe other sensors could be used resulting in more *local* methods.

Robustness Another crucial element is *robustness*. How do these algorithms fare in a non-ideal environment with external disturbances and model mismatch? Although some authors have done some real-world validation, it is hard to demonstrate robustness at scale. Algorithms that lack robustness can give rise to false alarms or missed detections, both decreasing the reliability of the whole quadrotor system.

Different goals and approaches Looking at the platform on which a work is published, and the background of its authors, it becomes clear that this particular fault detection problem draws interest from different research fields, such as:

- Aeronautical Engineering [12], [13], [19]
- Fault Tolerant Systems [14]
- Control Systems [17]
- Mechanical Engineering [8]

It seems that researchers from all these different backgrounds apply the methods to the problem that they are familiar with, without attempting to reach a common goal. Formulating that goal could be helpful in providing directions designing these systems and making better comparisons. A good first step would be to optimize for the three metrics Patton describes [5]:

- Detection delay
- False alarm rate
- Missed detection rate

Other goals could be improving computational efficiency, or metrics that are harder to quantify such as ‘operational’ efficiency. For example, lots of tunable parameters make a MAV more expensive to operate in the field.

2.3. Conclusion and recommendations

Although research is conducted proposing different fault detection methods, these studies often lack a realistic validation. Also, it is often unclear how the performance of these methods should be quantified, let alone what the exact performance of these methods is, and under what set of assumptions. Therefore, it is very difficult to draw specific conclusions. This absence of validation is a pity because validation is an essential step in proving the merits of these fault tolerant control methods. Thus lack of it inhibits the adoption of fault detection methods in the MAV sector. Resulting in a higher than needed risk associated with MAVs.

The reason these methods are often not validated is probably because it takes too much effort to do so, thus this validation gets reasoned out of scope. Therefore, it is recommended to first aim to decrease the workload needed to complete a design, implementation and validation cycle by building a (software) framework accelerating this process. Then, also use this newly designed process to iterate on an improved method, grounded in more realistic assumptions. Directions for further research are given in Chapter 3.

3

Project design

3.1. Research objectives

This research design is the result of an iterative process. The end state is described here.

Research objective The research objective is to contribute to the development of *active fault tolerant* flight control for quadrotor MAVs with the purpose of mitigating the risks associated with actuator faults, **by** the iterative *design, implementation and testing* of a loss of actuator effectiveness detection module on the Bebop 2 quadrotor MAV.

Research question How can a Kalman filter-based approach be used for detecting sudden loss of total actuator effectiveness, using onboard sensors and computations only, that is quick enough to prevent loss of control only when applied within an active fault tolerant control system?

Research strategy The steps that will lead to reaching the research objective are indicated by the **Gs**. The first goal is:

G1 To identify existing loss of actuator effectiveness detection methods and understand their limitations, if there are any, through a literature review.

This literature review is reported in Chapter 2. It was found that these methods have been through little real-world validation, this results in questionable assumptions and designs. With the understanding that improving these designs requires an iterative approach, the next goal is:

G2 To accelerate the design, implementation and test cycle, by applying workflow automation and the state of art in flight control software.

Then use the tools and processes designed in **G2**:

G3 To identify a method for producing relevant (simulated) actuator faults during flight testing.

Then use found method, as well as, tools and processes designed in **G2**:

G4 To gather data of a quadrotor subject to those faults.

Then use that data:

G5 To develop an efficient algorithm that can detect the loss of actuator effectiveness quickly, with low detection delay, low missed detection rate and low false alarm rate.

Then, again, use the tools and processes designed in **G2**:

G6 To integrate the proposed loss of actuator effectiveness detection algorithm in an active fault tolerant control system and evaluate its performance.

The results from steps **G1**, **G3**, **G5** and **G6** form the basis for the research paper.

Research questions These research questions are based on the research goals, out of **G2** follows:

RQ1 How can the design, implementation and test cycle be accelerated?

RQ1.1 How can a software pipeline, incorporating PX4 and Simulink, be developed that speeds up this process?

RQ1.2 What are suitable processes and tools to sustainably maintain and improve this process?

From **G3**, **G4**, and **G5**:

RQ2 How can a Kalman filter based approach provide satisfactory efficiency, reliability and detection delay?

RQ2.1 What actuator faults are most relevant to detect and can be simulated and/or reproduced?

RQ2.2 How can efficiency and reliability be defined and measured?

Finally, from **G6** follows:

RQ3 How does the proposed method perform when integrated within a fault tolerant control system?

RQ3.1 Can loss of control be prevented when an actuator failure happens?

RQ3.2 What are the implications of the changing dynamics during different flight phases?

3.2. Methodology

The theoretical basis of the methodology lies in the system description of the quadrotor. The quadrotor system encompasses the following (main) components:

- Sensor package [25]
  - Accelerometer + Gyro package (MPU-6050, datasheet: [26])
  - Rangefinder (unused)
  - Optical flow sensor (unused)
- ARM Cortex-A9 processor
- Actuators
  - Four three-bladed propellers driven by brushless motors

The real Bebop 2 behaves differently from its simulated counterparts in lots of ways. To name a few unmodeled phenomena:

- The physical connection between the sensor package and the actuator consists of a damper, thus the quadrotor is (partly) non-rigid
- All actuators have slight differences in control effectiveness
- The propellers are off-balance, shaking the sensors
- The center of gravity is (slightly) offset with respect to the geometric center
- Sensor measurements can be delayed
- Computations take time,
- (Wireless) data links can lose packets

Because no simulation currently covers all these and other uncertainties, flight testing is essential in improving the quadrotor model and its control system.

Research strategy, methodology This research will heavily make use of existing material, methods and processes and improves on them where possible. The main existing material consists of:

- An indoor drone test track (CyberZoo), equipped with a motion capture system
- A (modified) Bebop 2 quadrotor
- A high fidelity model of above Bebop 2 in MATLAB Simulink
- The Open Jet Facility (OJF) wind tunnel, also equipped with a motion capture system

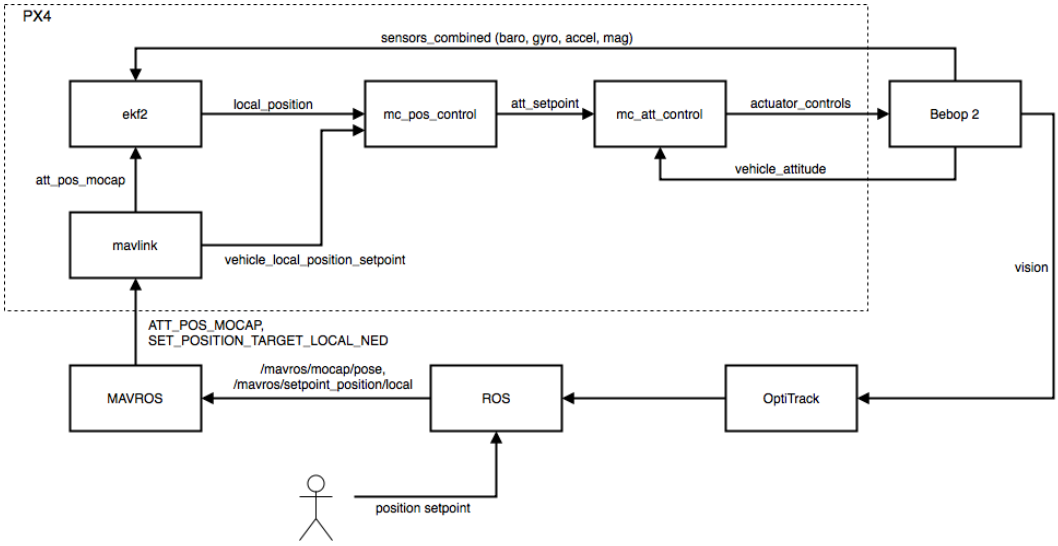


Figure 3.1: Overview of the tracking set-up, everything inside dashed PX4-box corresponds to PX4 modules and PX4 uORB-message topics.

Given the above items, there lies an opportunity to not only verify, but also validate its results using flight test data. To get to the best possible answers to the research questions the following steps should be taken iteratively:

1. Design, implement and simulate a FDD-method in the Simulink model
2. Implement that FDD-method on the Bebop 2 quadrotor
3. Test fly that method, simulate failures
4. Use the flight test data to off-line replay and iterate on that FDD method

Currently, all these steps take very long to take and thus must be accelerated. Especially the time item 2 and 4 take can be accelerated leveraging the Simulink C++ code generation capability. This will be done in combination with the PX4 flight controller. Parallel to this process the Simulink controllers and models should be validated and improved. This is necessary to validate the fault detection method and its assumptions. It also enables doing better simulations and making predictions on the performance of the method in a wider range of simulated conditions. As the amount and complexity of written software, and visually designed software in Simulink, increases it is extremely important to be organized in such a way that this does not inhibit progress. This will be done by applying tools such as:

1. Source control (Git/Bitbucket)
2. Workflow automation in general
3. Automated testing

The software is, as of the start of this project, not stored in a code repository. This severely inhibits code reuse and cooperation across multiple people. Putting the code under source control will improve the traceability and reproducibility of research, and will also improve productivity if used correctly.

3.3. Experimental set-up

Because the research strategy is driven by the available material, the experimental set-up consists of a set of available materials as discussed under Section 3.2. Considering the motion tracking set-up, the Bebop 2 will be tracked using OptiTrack in the CyberZoo or in the OJF. This tracking will allow indoor pose estimation. An overview is given in Figure 3.1

Simulating actuator failures One other interesting aspect of the experimental set-up is the method for propeller ejection, simulating an actuator failure. This failure can be enabled by giving a fast decrease in actuator setpoint on motor with a already very loose propeller. This will eject the propeller as in Figure 3.2, but on this method must be further improved on.

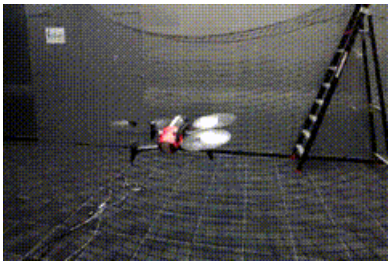


Figure 3.2: Preliminary test of the propeller ejection method.

3.4. Time planning

The planning follows from the requirements from the kick-off, and constraints in wind tunnel availability. The requirements specified on the kick-off form (AE 2) are adjusted to take into account: 1) the part-time job of the author of 0.2 FTE, one day a week, and 2) 2 weeks of holiday. The part-time job results in an adjusted week count of *nominal weeks times 1.25*. Counting up from the given start date of 6 July 2018, and adjusting the week count for absence, this results in the high-level planning given in Table 3.1.

Table 3.1: High-level (optimistic) project planning.

Event	Nom. wk	Adj. wk	Date
Start graduation project	0	0	11-6-2018
Kick-off	2	2	
Literature study report and presentation	12	15	wk-39-2018
Midterm review	∼25	32	wk-4-2019
Wind tunnel day, 1st week	-	37	wk-9-2019
Wind tunnel day, 2nd week	-	40	wk-12-2019
Green light review	∼34	44	wk-16-2019
Expected hand-in date	38	47	wk-19-2019
Master thesis presentation and defense	-	49	wk-21-2019

CyberZoo The CyberZoo can be reserved on a 'first-come first-served'-basis using the predesignated Google Calendar (`cyberzoo.tudelft@gmail.com`). Access to the calendar is granted via the MAVLab and requires a safety briefing. After access is granted, activities in this lab can be planned days or weeks in advance, or need no planning at all if the lab is available.

Wind tunnel (OJF) The wind tunnel tests are planned in weeks 9 and 12 of 2019. The wind tunnel testing will be done in cooperation with the daily supervisor of this project. The OJF will be allocated for approximately one day of each test week for this project, with the other days being allocated to the research of the daily supervisor of the author.

Interlinking and iterations The weeks before these tests must be used to get a workable set-up that enables quick adaption and iteration during the wind tunnel tests especially. Weeks 10 and 11 are the weeks between the wind tunnel tests, thus must be used for: 1) evaluating the results from the tests in week 9, and 2) making adaptations to the test plan for week 12, thus allowing for one more iteration.

3.5. Conclusion

This project builds on an excellent basis of available tools and facilities. These include the Bebop 2 UAV, wind tunnel (OJF), Simulink model of the Bebop. The unique availability of these tools, combined with a knowledge gap on the validation of Fault Detection and Diagnosis, give an excellent research opportunity. This research will bridge the gap between simulations studies and workable solutions.

A

Survey on flight control software

The objective of the flight control software (FCS) is to convert higher-level control objectives and sensor inputs to actuator commands. Therefore, it forms an integral part in research done on flight control. The goal of this survey is to give an overview of open source FCS and investigate the main trends.

FCSs not included in this survey It looks like the **-flight* family *(betaflight, cleanflight)* is used extensively in the hobby space. ArduPilot is a very big project, also wildly used in the hobby space, and also applied in research[1] but is not included in this comparison.

A.1. Comparison

In MAV research there are two main FCS projects, PX4 and Paparazzi (PPZ). Table A.1 compares some key properties of these projects.

Build system Paparazzi makes heavy use of the C pre-processor and all kinds of XML-based C-code generators, making the codebase difficult to maintain. As the PPZ-wiki[2] puts it: “It took all the people involved blood, sweat and tears to get where the paparazzi is now.” and “Paparazzi is a large project with years of development so it can be a daunting challenge to edit the source code (...)”. These systems were probably put in place so that the resulting PPZ-binary can be very memory efficient, even across different configurations.

A.2. PX4 roadmap

The PX4 roadmap for 2019 can be found here: `https://discuss.px4.io/t/px4-roadmap-2019`
In it, Figures A.1 and A.2 can be found. The roadmap presents some opportunities.

Attitude control It looks like there are no plans for improving attitude control. At time of writing the rate controller still consists of an old-fashioned PID[3] loop.

Events interface The current message bus in PX4, uORB, only supports signals. An events interface provides an interface for events-like messages. This is useful for events as mode switching, or fault detection, when the value does not change often.

Flight test automation Having better flight test automation will greatly speed up validation of algorithms and software.

[1] https://gitter.im/ArduPilot/Research
[2] http://wiki.paparazziuav.org/wiki/DevGuide/LearningToProgram
[3] For implementation see: `https://github.com/PX4/Firmware/blob/master/src/modules/mc_att_control/mc_att_control_main.cpp#L600`

Table A.1: Table comparing some key properties of the Paparazzi (PPZ) and PX4 projects.

	PPZ	PX4
Project		
Stargazers	~1000	~2600
Active contributors[1]	20	46
Start year	2005	2012
License	GPLv2	BSD 3-clause
S/W		
Build system	CMake, custom XML-based	CMake
Architecture	Multi[2]	Modules, pub/sub [27]
Main language	C	C++
H/W requirements		
Flash	256-1024 kB[3]	2 MB
RAM	16-192 kB[3]	256-512 kB[4]

[1] Number of people that made a contribution between 1-1-2018 and 5-5-2019.
[2] `http://wiki.paparazziuav.org/wiki/DevGuide/DesignOverview`
[3] Lisa/S - Elle0 boards.
[4] Pixracer - Pixhawk 4 boards.

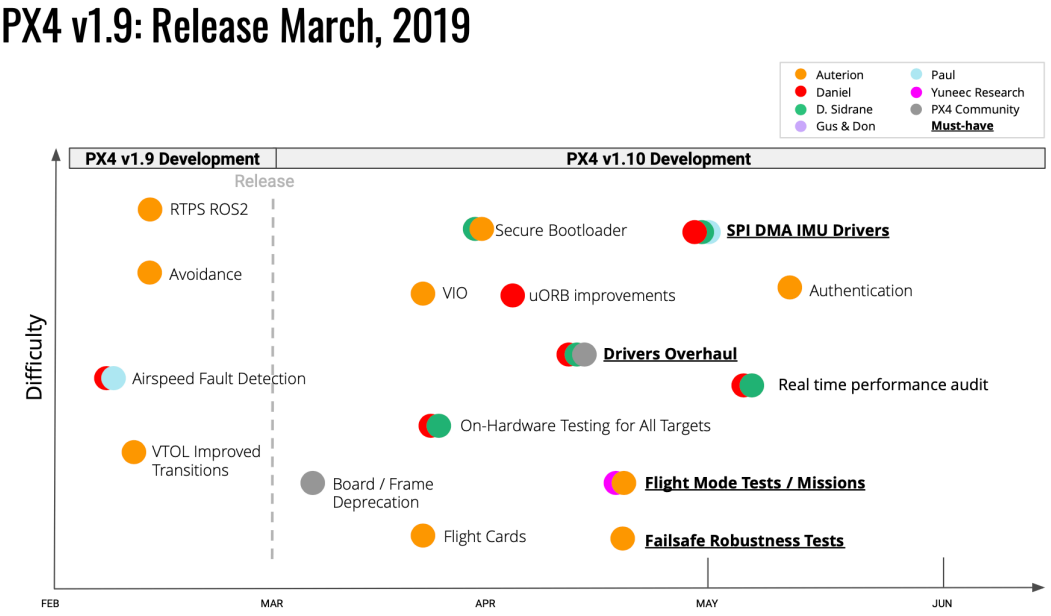


Figure A.1: PX4 2019 roadmap for version 1.9

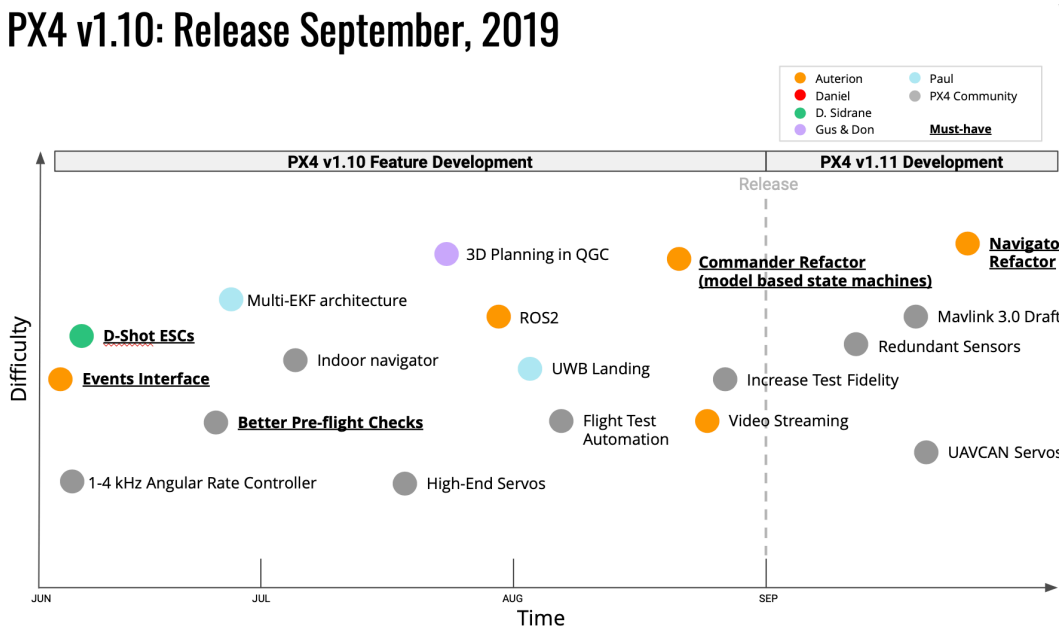


Figure A.2: PX4 2019 roadmap for version 1.10

Bibliography

[1] "UAS Safety Risk Portfolio and Analysis", EASA, 2016.

[2] C. M. Belcastro, D. H. Klyde, M. J. Logan, R. L. Newman, and J. V. Foster, "Experimental Flight Testing for Assessing the Safety of Unmanned Aircraft System Safety-Critical Operations", in *17th AIAA Aviation Technology, Integration, and Operations Conference*, Denver, Colorado: American Institute of Aeronautics and Astronautics, Jun. 5, 2017, ISBN: 978-1-62410-508-1. DOI: 10.2514/6.2017-3274.

[3] M. W. Mueller and R. D'Andrea, "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers", in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 45–52. DOI: 10.1109/ICRA.2014.6906588.

[4] S. Sun, L. Sijbers, X. Wang, and C. de Visser, "High-Speed Flight of Quadrotor Despite Loss of Single Rotor", *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3201–3207, Oct. 2018, ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2851028.

[5] R. J. Patton, "FAULT-TOLERANT CONTROL SYSTEMS: THE 1997 SITUATION", p. 22,

[6] C. Edwards, T. Lombaerts, and H. Smaili, Eds., *Fault Tolerant Flight Control: A Benchmark Challenge*, Lecture Notes in Control and Information Sciences 399, OCLC: ocn502034157, Berlin: Springer, 2010, 586 pp., ISBN: 978-3-642-11689-6.

[7] Y. Jiang, Z. Zhiyao, L. Haoxiang, and Q. Quan, "Fault detection and identification for quadrotor based on airframe vibration signals: A data-driven method", in *2015 34th Chinese Control Conference (CCC)*, Jul. 2015, pp. 6356–6361. DOI: 10.1109/ChiCC.2015.7260639.

[8] B. Ghalamchi and M. Mueller, "Vibration-Based Propeller Fault Diagnosis for Multicopters", *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, Jun. 2018.

[9] J. M. Brown, J. A. Coffey, D. Harvey, and J. M. Thayer, "Characterization and Prognosis of Multirotor Failures", in *Structural Health Monitoring and Damage Detection, Volume 7*, C. Niezrecki, Ed., ser. Conference Proceedings of the Society for Experimental Mechanics Series, Springer International Publishing, 2015, pp. 157–173, ISBN: 978-3-319-15230-1.

[10] P. Misra, G. Kandaswamy, P. Mohapatra, K. Kumar, and P. Balamuralidhar, "Structural Health Monitoring of Multi-Rotor Micro Aerial Vehicles", in *Proceedings of the 4th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications - DroNet'18*, Munich, Germany: ACM Press, 2018, pp. 21–26, ISBN: 978-1-4503-5839-2. DOI: 10.1145/3213526.3213531.

[11] A. Freddi, S. Longhi, and A. Monteriù, "Actuator fault detection system for a mini-quadrotor", in *2010 IEEE International Symposium on Industrial Electronics*, Jul. 2010, pp. 2055–2060. DOI: 10.1109/ISIE.2010.5637750.

[12] M. Frangenberg, J. Stephan, and W. Fichter, "Fast Actuator Fault Detection and Reconfiguration for Multicopters", in *AIAA Guidance, Navigation, and Control Conference*, ser. AIAA SciTech Forum, 0 vols., American Institute of Aeronautics and Astronautics, Jan. 2, 2015. DOI: 10.2514/6.2015-1766.

[13] P. Lu and E.-J. van Kampen, "Active fault-tolerant control for quadrotors subjected to a complete rotor failure", in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany: IEEE, Sep. 2015, pp. 4698–4703, ISBN: 978-1-4799-9994-1. DOI: 10.1109/IROS.2015.7354046.

[14] D. Vey and J. Lunze, "Experimental evaluation of an active fault-tolerant control scheme for multirotor UAVs", in *2016 3rd Conference on Control and Fault-Tolerant Systems (SysTol)*, Sep. 2016, pp. 125–132. DOI: 10.1109/SYSTOL.2016.7739739.

[15] E. J. J. Smeur, G. C. H. E. de Croon, and Q. Chu, "Cascaded Incremental Nonlinear Dynamic Inversion Control for MAV Disturbance Rejection", Jan. 25, 2017. arXiv: 1701.07254 [cs].

[16] R. C. Avram, “Fault Diagnosis and Fault-Tolerant Control of Quadrotor UAVs”, Wright State University, 136 pp.

[17] R. C. Avram, X. Zhang, and J. Muse, “Quadrotor Actuator Fault Diagnosis and Accommodation Using Nonlinear Adaptive Estimators”, *IEEE Transactions on Control Systems Technology*, vol. 25, no. 6, pp. 2219–2226, Nov. 2017, ISSN: 1063-6536. DOI: 10.1109/TCST.2016.2640941.

[18] A. Hasan and T. A. Johansen, “Model-Based Actuator Fault Diagnosis in Multirotor UAVs”, in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, Dallas, TX: IEEE, Jun. 2018, pp. 1017–1024, ISBN: 978-1-5386-1354-2. DOI: 10.1109/ICUAS.2018.8453420.

[19] Y. Zhong, Y. Zhang, W. Zhang, J. Zuo, and H. Zhan, “Robust Actuator Fault Detection and Diagnosis for a Quadrotor UAV With External Disturbances”, *IEEE Access*, vol. 6, pp. 48 169–48 180, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2867574.

[20] N. P. Nguyen and S. K. Hong, “Sliding Mode Thau Observer for Actuator Fault Diagnosis of Quadcopter UAVs”, *Applied Sciences*, vol. 8, no. 10, p. 1893, Oct. 11, 2018, ISSN: 2076-3417. DOI: 10.3390/app8101893.

[21] Z. Cen, H. Noura, and Y. A. Younes, “Robust Fault Estimation on a real quadrotor UAV using optimized Adaptive Thau Observer”, in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2013, pp. 550–556. DOI: 10.1109/ICUAS.2013.6564732.

[22] Z. Cen, H. Noura, and Y. A. Younes, “Systematic Fault Tolerant Control Based on Adaptive Thau Observer Estimation for Quadrotor Uavs”, *International Journal of Applied Mathematics and Computer Science*, vol. 25, no. 1, pp. 159–174, Mar. 1, 2015, ISSN: 2083-8492. DOI: 10.1515/amcs-2015-0012.

[23] M. H. Amoozgar, A. Chamseddine, and Y. Zhang, “Experimental Test of a Two-Stage Kalman Filter for Actuator Fault Detection and Diagnosis of an Unmanned Quadrotor Helicopter”, *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1-4, pp. 107–117, Apr. 2013, ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-012-9757-7.

[24] Y. Guo, B. Jiang, and Y. Zhang, “A novel robust attitude control for quadrotor aircraft subject to actuator faults and wind gusts”, *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 292–300, Jan. 2018, ISSN: 2329-9266. DOI: 10.1109/JAS.2017.7510679.

[25] Freek van Tienen et al. (). Bebop 2 details, [Online]. Available: https://wiki.paparazziuav.org/wiki/Bebop.

[26] InvenSense, “MPU-6000 and MPU-6050 Product Specification Revision 3.4”, Datasheet, 2013.

[27] L. Meier, D. Honegger, and M. Pollefeys, “PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms”, in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 6235–6240. DOI: 10.1109/ICRA.2015.7140074.