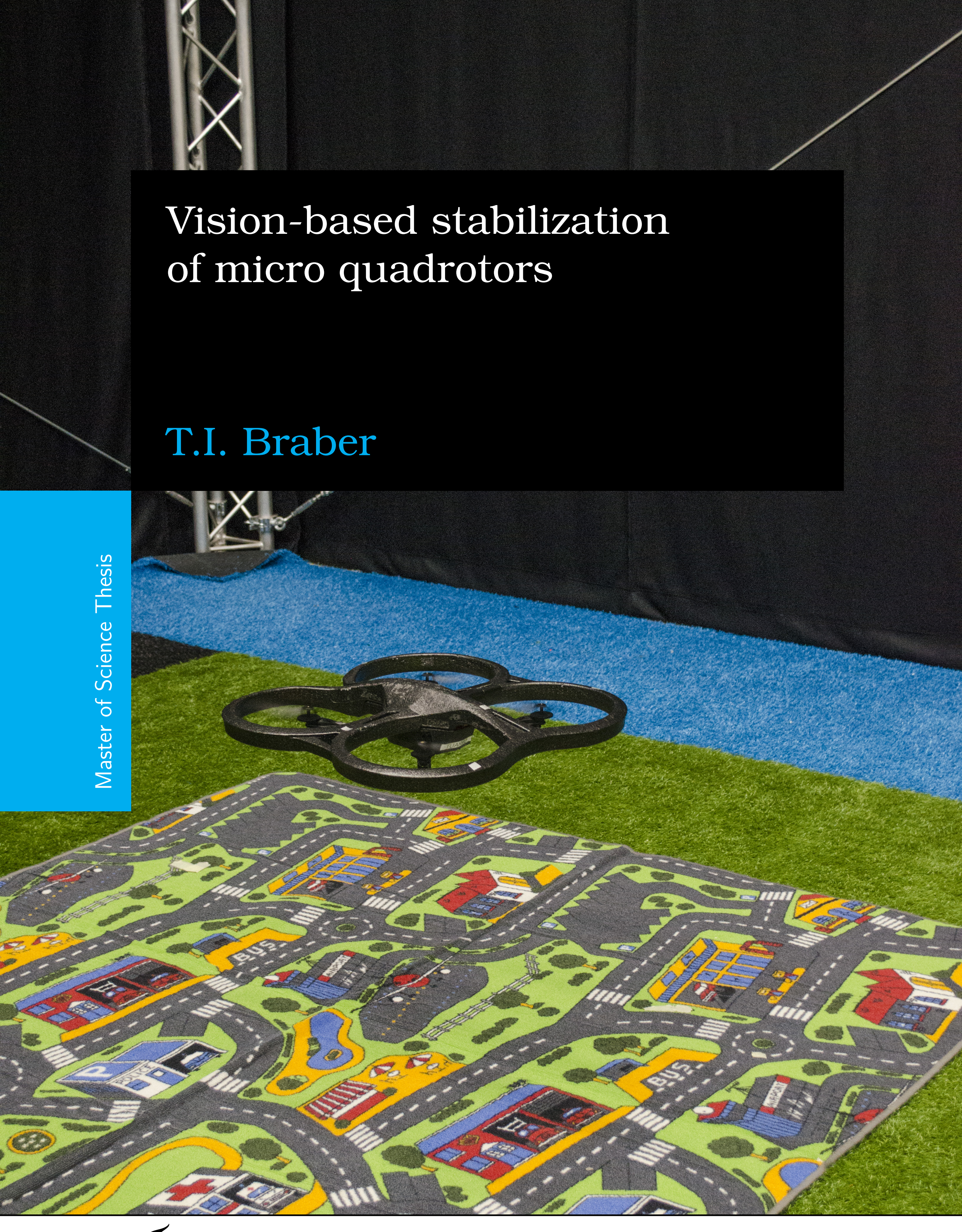


# Vision-based stabilization of micro quadrotors

T.I. Braber

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

# Vision-based stabilization of micro quadrotors

Master of Science Thesis

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

T.I. Braber

September 21, 2017

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

This MSc Thesis was done at the Micro Air Vehicle Laboratory (MAVlab) at the faculty of Aerospace Engineering.

# Abstract

On-board stabilization of quadrotors is often done using an Inertial Measurement Unit (IMU), aided by additional sensors to combat the IMU drift. For example, GPS readings can aid when flying outdoors, or when flying in GPS denied environments, such as indoors, visual information from one or more camera modules can be used.

A single downwards facing camera however cannot determine the absolute height of the quadrotor, leaving the results from the Optical Flow (OF) up to scale. To estimate the velocity of the quadrotor an additional range sensor, such as an Ultrasonic Sensor (US), is used to solve this scaling problem.

These solutions are difficult to scale down to micro quadrotors as the platform becomes too small to fit and lift additional sensors. Therefore stabilizing a quadrotor with a single camera and IMU only would pave the way for the development of even smaller quadrotors.

This master thesis presents an adaptive control strategy to stabilize a micro quadrotor in all three axes using only an IMU and a monocular camera. This is achieved by extending the stability based approach for a single, vertical, axis by De Croon in Distance estimation with efference copies and optical flow maneuvers: a stability-based strategy[1]. This stability based method increases the control gain in the visual feedback loop until the quadrotor detects it is oscillating by detecting that the covariance of the given thrust inputs and the measured divergence passes a threshold. Next the height can be estimated using the predetermined relationship between gain and height at which these self-induced oscillations occur and proper gains can be set for the estimated height.

An analysis is done in simulation to present proof of concept of the stabilization method in three axis and to determine the effects of scaling and the effects of varying effective Frames per Second (FPS) caused by computations. It was shown that the adaptive gain strategy can stabilize the simulated quadrotor and prevent it from drifting. Furthermore, the control gains were scaled such that the effects of scaling a quadrotor could be mostly negated, though at about a tenth of the scale the simulated noise had such an influence that the scaled gains could not negate it anymore. Furthermore, the minimum effective FPS required to stabilize an ARDrone 2 was determined to be 15 FPS, and it was shown that an increase in effective

FPS aids stabilizing the smaller scale quadrotors that became unstable due to the scaling effects.

Furthermore, flights on an Parrot ARDrone 2 and Parrot Bebop are performed to show the usability of this control strategy in real life. It was shown that both quadrotors could achieve stable hover without drifting at multiple heights, using various strategies.

# Table of Contents

# Preface

This thesis, "Vision-based stabilization of micro quadrotors", is the product of the graduation project for the degree of Master of Science at Delft University of Technology as a Mechanical Engineer with a specialization in Control Engineering.

For my graduation project I wanted to work on something related to robotics, with a tangible component. I have worked on various driving robots in my bachelor, but my first encounter with quadrotors was during the Autonomous Flight of Micro Air Vehicles course taught by Guido de Croon at the faculty of Aerospace Engineering. A competition was included in the course where each group had to make an ARDrone 2 autonomously avoid objects in the Cyber Zoo. I worked on the obstacle detection using vision, but that was only a part of the project.

Looking for a thesis I talked to Guido about his recent publication that allowed quadrotors to land and estimate their height using only a single camera and how that might be used to stabilize a quadrotor in all three axes. With Christophe de Wagter as an additional supervisor from the MAVLab and Robert Babuska as my supervisor from my faculty, Mechanical, Maritime and Materials Engineering (3mE), the thesis was started.

I have learned a great deal during my thesis on how such a project works, how open-source projects are organized and how quadrotors are controlled. Though it was not always easy this graduation project has been a valuable experience.

I would like to thank my supervisors Guido, Christophe and Robert for their support and feedback, Erik van der Horst from the MAVLab for helping me with hardware related issues with the quadrotors and my fellow students Laurens Valk and Tom van Dijk for our conversations and brainstorming sessions. Finally I would like to thank my girlfriend Monique van der Kooij for always being there for me, even in times when things were tough.

Delft, University of Technology — Titus Ignatius Braber
September 21, 2017

Chapter 1

# Introduction

Drones, also called Unmanned Aerial Vehicle (UAV), are becoming more common in our society everyday, from large unmanned fixed wing aircrafts used in military service for reconnaissance to flapping wing mechanical birds of prey to chase flocks away from airports. Quadrotors are a commonly used type of UAV. In the industry quadrotors are used for the inspection of buildings or infrastructure, (visual) surveillance of an area for intruders or forests for potential fires, photography and shooting film, the entertainment industry where quadrotors are being sold as toys and recently many more applications such as cargo delivery. The growing use of quadrotors in our lives can be explained by the recent developments in various scientific fields.

Quadrotors have been a popular subject of research due to their flying nature, capability of hovering and performing aggressive maneuvers and simple dynamics. In the past visually exciting research has been done on pole[2] and ball[3] juggling quadrotors, performing aggressive maneuvers[4], flying in swarms[5] and constructing structures[6]. Furthermore, theoretical exciting research has also been done, including bio inspired detection of landing height[1] and various other smart adaptations to improve computational efficiency[7] or counter drifting[8].

A quadrotor is a rotary-wing aircraft that relies on 4 rotors to create lift and maneuver. By dividing these 4 rotors into two pairs, diagonally matched rotors turning clockwise and counter clockwise respectively, the torque pairs cancel each other out, resulting in 0 angular acceleration about the yaw axis during hover. Furthermore, it is easy to change the thrust, yaw, pitch and roll of the quadrotor by changing the speeds of individual rotors. Having 4 actuators and 6 Degrees Of Freedom (DOF) a quadrotor is underactuated. Nonetheless it is still able to reach all states.

Being naturally unstable platforms, quadrotors require accurate and frequent estimations of their velocity and position to perform (aggressive) maneuvers. Inside controlled environments this is easier to realize, however in order to be of use in our daily lives quadrotors also need to be able to perform in unknown and uncontrolled environments. Outside, Global Positioning System (GPS) has proven to aid in these estimations, leading to commercial quadrotors following a path of waypoints or returning to the pilot autonomously when batteries run low.

Indoors however, quadrotors often have limited to no access to GPS, leading to new research and different solutions. Furthermore, the error margin is much smaller indoors as obstacles are always near.

Autonomous quadrotors have become smaller with the recent advances in technology, allowing them to be used indoor and be inherently safer to operate around humans. However, even the smallest autonomous quadrotors are still larger than some Remote Controlled (RC) (toy) versions. To create autonomous quadrotors that can safely operate in swarms and fly in even smaller areas than currently possible, more research is needed.

This leads to the goal of this MSc thesis, to determine what is needed to realize the smallest autonomously stabilizing quadrotor to date.

## 1-1 Size definitions

With Defense Advanced Research Projects Agency (DARPA)'s push towards smaller UAVs, the Micro Aerial Vehicle (MAV) program initiative[9] was born in 1997. This program defined the goal of MAVs to be autonomous air vehicles that are smaller than $15cm$, weighing around 50 grams, flying $20-60$ minutes or less than $10km$. It is clear that these requirements are still not met, if only due to the weight to power ratio of batteries.

However, with quadrotors being a class of MAVs, literature has used the term *micro quadrotors* to define various sizes of quadrotors, including a $51.7cm$ Parrot ARDrone and 21 cm micro quadrotors flying in a swarm[5]. The authors of [10] use *nano* to describe the size of their 15 cm *nano* quadrotor. DARPA has also coined Nano Aerial Vehicle (NAV) in 2005 [11][12], with a maximum diameter of $7,5cm$ and a maxmium weight of 10 gram.

The definition of pocket drone[7] is an interesting one for this MSc thesis as it captures the size of the quadrotor better than *micro*, with a diameter of 10 cm being able to fit inside your pocket.

## 1-2 Challenges of micro quadrotors

There are several challenges in making quadrotors smaller. When scaling things scaling laws apply. Making something 10 times smaller, already reduces the weight by a factor 1000, assuming uniform scaling. This potentially has great influence on the dynamics of smaller quadrotors. An analysis is done in Section 4-4.

Furthermore, mechanically speaking, smaller mechanisms are more complex to produce, and parts such as motors cannot be scaled down all the way without losing performance. This limits the producibility at certain scales. With smaller quadrotors there is also less thrust available due to usage of smaller motors, leading to reduced load carrying capacity. This in turn results in less weight for sensors and a Central Processing Unit (CPU). Power is one of the largest challenges in daily operation as limited battery power lowers the maximum continuous flight time. A long surveillance flight therefore isn't possible.

A smaller CPU means less computational power which influences the complexity of the algorithms that can be run on-board in real time. Especially when processing images from a camera this is a limiting factor. And finally limited weight restricts the type of (additional) sensors that can be fitted on-board.

These restrictions form a major challenge for this MSc thesis as the proven solutions for autonomous stabilization are excluded due to the lack of sensors and processing power on-board, in combination with faster dynamics. This MSc Thesis will not focus on the mechanical or electrical design but on the control strategy for an existing platform given the mentioned limitations.

## 1-3 Goal of the MSc Thesis

This MSc thesis aims to achieve the following:

- *Develop a scalable control algorithm that can autonomously stabilize micro quadrotors*

Stable hover can be described as hovering at the roughly the same position, without large vibrations or drift. Note that it is not necessary to return to that position when moved away from it by hand, instead stable hover should be achieved in the new position. Thus there is no position control, rather stabilizing the velocity around 0.

In this research the challenges highlighted in Section 1-2 will be taken into account. As some of these challenges are easily overcome by for example moving the computations and sensors offboard or structuring the environment, additional requirements must be set:

- All sensing and computations must be performed on-board.
- The environment must be unstructured, though is assumed to be sufficiently textured.

These additional requirements ensure that the quadrotor can navigate autonomously in unknown environments without the need of setup of external equipment. Experiments can however be done in an environment with equipment set up to obtain a ground truth for evaluating performance of the quadrotor.

Given this goal the main research question can be defined as:

- *Is it possible to stabilize a quadrotor as small as possible, using only a mono camera, an IMU and on-board computation?*

Following the main research question sub-questions can be formulated:

- How accurate will the stabilization be?
- How does the performance change when the dimensions of the quadrotor scale down?

## 1-4 Structure

The remainder of this MSc thesis is structured as follows. In Chapter 2 the literature study that was performed before this MSc thesis is summarized. Chapter 3 presents the approach that will be taken using simulation and experiments on a quadrotor. The simulator design and implementation in MATLAB and Simulink will be discussed in Chapter 4. Furthermore, the effects of scaling will be investigated. Chapter 5 will show the implementation on the quadrotor using the Paparazzi autopilot and the test environment. The results of both the simulation and flights on the quadrotor will be presented and discussed in Chapter 6. Finally the MSc thesis will be concluded in Chapter 7 and recommendations for future work will be done in 8.

Chapter 2

# Literature Summary

This chapter summarizes the relevant parts of the literature study done at the beginning of the master thesis. The study was performed to gain an insight in the current state of the scientific field and possible solutions to the problem. An overview was made of methods used for stabilizing quadrotors, which was used to determine a suitable approach. Furthermore, a widely used technique of processing images to obtain estimates of movement was analyzed. The following sections will give an overview of the most important findings, followed by a section summarizing the identified key papers and a section in which the research direction is concluded.

## 2-1 Quadrotor Stabilizing

To stabilize a quadrotor in flight one requires a control algorithm based on an estimation of the (angular) velocities of the quadrotor. These estimates can be based on various types of sensors. In order to select a suitable method for stabilizing micro quadrotors, the solution has to scale properly.

Note that due to the indoor requirement, GPS will be unavailable. Also taking into account the requirement of autonomous stabilization without external sensing, as described in section 1-3, many of the conventional solutions can be excluded. These include using camera systems fixed to the environment, and using light and sound range finding solutions. Essentially only the use of a single small camera can be allowed to taken aboard in terms of weight. Note that it was assumed only using an IMU would not be sufficient, due to the integration of biases and noise leading to sensor drift.

Within the monocular vision branch, solutions requiring knowledge of the environment were excluded, because of the requirements of operating in unknown environments as defined in section 1-3. Furthermore, the algorithms have to run fast enough and fit on-board microcontrollers, leaving only the Reactive Optical Flow as an option, excluding the computational more expensive Visual Odometry (VO) and Simultaneous Localization And Mapping (SLAM).

This leaves us with the following branch path from the tree that was created in the literature study. The full tree can be found in Appendix A.

**Stabilization**
|
Fused IMU
|
Vision
|
On-board
|
Mono
|
Unknown Target
|
Reactive Optical Flow

## 2-2 Reactive OF

From [13] OF is the apparent relative motion between observer and scene. This motion can described in a motion field, constructed by the displacement of brightness patterns between frames. Hence constant illumination of the scene is assumed in most cases. Furthermore, in order to be detectable displacements must be uniform (spatial smoothness) and small with respect to the previous frame.

As in nature, the use of a single eye gives insufficient information to estimate depth correctly. The results from OF are up to scale and therefore not able to differentiate between a small movement close to the scene, and a large movement further away from the scene. This difference is important for the control however. To solve this problem there are multiple solutions. As the OF is defined as the ratio between velocity over height one option would be to simply measure the height using an second camera, or a range sensor. This method however is excluded due to the addition of an additional sensor.

### 2-2-1 Model based

The observable ratio between velocity and height can not be decoupled, however if one would have an accurate model of the quadrotor a known thrust input could be given to determine the resulting velocity. Using this estimated velocity the height can be determined from the OF.

The drawback of this model is that in practice it is difficult to obtain a model accurate enough, let alone account for external disturbances that influence the speed of the quadrotor.

### 2-2-2 Inertial Measurement Unit (IMU) based

As [14] shows, if one controls for a constant rate of optic flow the distance can be estimated using the control input and the desired rate of optic flow. As appendix B.2[1] adds for horizontal movement: with a non zero input in x direction, constant ventral flow and non zero divergence, the height can be estimated using the observed divergence and ventral flow, and the measured acceleration in x direction.

The main drawback in these cases is that an active acceleration is required, either in horizontal and vertical direction. Therefore the height estimate cannot be performed during hover.

### 2-2-3 Instability based

In [1] it was analytically shown that given a fixed control gain on the observable divergence, there is a matching height for which the system becomes unstable and starts oscillating. These self-induced oscillations can be detected and distinguished from disturbances due to external influences such as wind gusts. If the quadrotor varies the control gain until self-induced oscillations are detected, the height can be estimated. Summarizing the proof for the vertical Z axis:

Take a state space system with state $\mathbf{x} = [z(t), \dot{z}(t)]^T$ and assuming $u_z$ directly and only influences the vertical acceleration $\ddot{z}(t) = u_z$. Note that for this to be a valid assumption the effect of gravity is included in $u_z$.

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + Bu_z(t)$$

The matrices $A$ and $B$ can be determined as

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2-1}$$

Furthermore, divergence, $g(\mathbf{x}) = Div = \frac{\dot{z}}{z}$, which is nonlinear function of the state, is taken as the output $y$. In order to represent it in a state space it is linearized

$$\mathbf{\Delta y} = C\mathbf{\Delta x}(t) + D\mathbf{\Delta}u_z(t) \tag{2-2}$$

Now matrices $C$ and $D$ can be computed as

$$\begin{aligned} C &= \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial Div}{\partial \mathbf{x}} = \frac{\partial \frac{\dot{z}}{z}}{\partial z} + \frac{\partial \frac{\dot{z}}{z}}{\partial \dot{z}} = \begin{bmatrix} -\frac{\dot{z}}{z^2} & \frac{1}{z} \end{bmatrix} \\ D &= \begin{bmatrix} 0 \end{bmatrix} \end{aligned} \tag{2-3}$$

Next the system can be discretized by applying a Zero Order Hold (ZOH) with sample time $T$ resulting in the following matrices

$$\Phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \Gamma = \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix}$$
$$C = \begin{bmatrix} -\frac{\dot{z}}{z^2} & \frac{1}{z} \end{bmatrix}, D = \begin{bmatrix} 0 \end{bmatrix} \tag{2-4}$$

From this, using $w$ as the $Z$-transform variable the Open Loop (OL) transfer function can be computed

$$G_{ol}(w) = C \cdot (wI - \Phi)^{-1} \cdot \Gamma$$
$$G_{ol}(w) = \frac{(zT - \frac{1}{2}\dot{z}T^2)w - zT - \frac{1}{2}\dot{z}T^2}{z^2(w-1)^2} \tag{2-5}$$

Then applying negative feedback gives the Closed Loop (CL) transfer function

$$G_{cl} = \frac{K \cdot G_{OL}}{1 + K \cdot G_{OL}}$$
$$G_{cl} = \frac{K\left((zT - \frac{1}{2}\dot{z}T^2)w - zT - \frac{1}{2}\dot{z}T^2\right)}{z^2(w-1)^2 + K\left((zT - \frac{1}{2}\dot{z}T^2)w - zT - \frac{1}{2}\dot{z}T^2\right)} \tag{2-6}$$

Assuming $z > 0$ and $T > 0$, the poles and zeros of Equation 2-6 can be determined for a given set of $z, \dot{z}, T$. There is an implicit zero at infinity and a finite zero at

$$w_0 = \frac{zT + \frac{1}{2}\dot{z}T^2}{zT - \frac{1}{2}\dot{z}T^2} \tag{2-7}$$

Now assuming $z > 0$, $T > 0$ and $\dot{z} < 0$, in the case of landing, the fraction will result in a zero slightly smaller than 1. Furthermore, there are two poles at $w = 1$ when $K = 0$. When the gain $K$ increases the poles in the Root Locus plot will move towards the finite and infinite zero. The pole moving towards the infinite zero exits the unit circle for $w = -1$. Therefore plugging that in the denominator in Equation 2-6 and solving for 0 gives us the gain $K$ for which the system will become unstable given height $z$

$$K = \frac{2}{T}z, \tag{2-8}$$

It can also be expressed in terms of height at which the system becomes unstable given a certain gain $K$

$$z = \frac{T}{2}K \tag{2-9}$$

This means that there is a direct relationship between the height and the gain at which the system becomes unstable, which can be used to estimate the one property given the other.

Though the author used the divergence as the research was focused on landing, the proof is generalized for horizontal axes in appendix B.1[1].

The main drawback of this method is that in order to determine the scale the control gain must be varied up to the point of instability during hover. Furthermore, the relationship between the height and the gain must be determined by calibrating the setup. This could be done with a single successful landing, assuming the gain height relation goes through the $(0, 0)$ point or multiple oscillations at different heights.

## 2-3 Identification of key papers

During the literature study three papers have been marked as key papers, to be used during the research. [1] has been mentioned in previous sections as a solution to solve the scaling issue of OF during hover. [7] focuses on very lightweight algorithm for OF estimation. And finally [8] proposes a way to counter drift by taking snapshots during hover. The following subsections will each summarize the findings of these papers.

### 2-3-1 Stability based approach

In [1] the detection of self induced oscillations due to instability during OF based landing is used to trigger the final touchdown. Furthermore, the author shows that varying the control gain at a specific height until oscillations occur can be used to determine at what height the quadrotor is flying as described in 2-2-3. Finally by descending or ascending on the edge of oscillations the quadrotor can continuously estimate its height. These oscillations can be detected using the Fast Fourier Transform, however as computational efficiency is of great importance an other method is proposed. Either the covariance of the divergence and efference copies, the past control inputs, or the auto covariance of the divergence is used to detect oscillations.

A Parrot ARDrone is used to perform experiments with the algorithm running on-board. The research also shows this method is accurate despite heavy wind gusts applied in simulation. Note that this research is purely focused on the vertical movement as the position in xy-plane is being controlled by an external vision system, however the same principle could be useful for stabilizing the quadrotor in the xy-plane.

### 2-3-2 EdgeFlow

In efforts of making even smaller quadrotors fly autonomously, [7] presents a novel algorithm called EdgeFlow that is faster and more accurate than Lucas-Kanade (LK). The authors use it to estimate the velocity of a 45 gram *pocket drone*. This algorithm runs on-board using a 4 gram stereo camera setup that also determines the height of the quadrotor.

It uses a Sobel filter to detect horizontal and vertical edges. These are converted to histograms which are used to determine the shift in features. Multiple tests using ground truth and estimated velocity in the control loop show the potential of this computationally efficient algorithm.

To obtain sub-pixel flow the authors suggest a time horizon adaptation, which compares the current frame with an older frame rather than the previous. The age of this older frame, the time horizon, is implemented adaptive, inversely proportional to the absolute flow calculated at the previous step. However, a maximum horizon is set due to limited memory requirements.

With their novel EdgeFlow algorithm the authors have set the stage for new algorithms and even smaller autonomous quadrotors.

### 2-3-3 Snapshot based approach

[8] proposes a snapshot based approach where the drift, during hovering using OF, is countered by taking a snapshot to compare against new frames. One could say this algorithm partly achieves position control instead of stabilization, as it returns to the position it was initialized at. However, if the tracking is lost, a new snapshot is taken.

In order to make the algorithm more robust against changes in illumination of the scene the intensity images are converted to binary images, using the incremental sign correlation algorithm. Furthermore, the binary image is encoded in such a way that a lookup table can improve computational efficiency. To further decrease computational costs only the center of each new image is taken to find a match in the original snapshot.

Tests are performed on an ARDrone both indoors and outdoors at various heights. The snapshot algorithm proves effective against positional drift.

## 2-4 Conclusion of literature study

This chapter has given a summary of the literature research on creating smaller micro quadrotors than currently available that can still achieve stable hover autonomously. Commercially available (toy) quadrotors like Parrot ARDrones achieve this easily using a set of sensors including a downwards facing camera + Ultrasonic Sensor (US), and optionally a GPS receiver when flying outdoors. However, when scaling quadrotors down to the range of the CX-10 RC toy quadrotor or smaller, their lifting capabilities are not sufficient to carry all kinds of sensors, nor a powerful computer to perform calculations on-board. It was concluded that this severely limits the quadrotors ability to hover autonomously as proven solutions are no longer feasible.

The best chance of achieving the goal of this master thesis would be to use a single camera and an IMU in combination with the lightweight EdgeFlow algorithm, extended with the snapshot approach. To solve the scaling issue the stability based approach will be implemented, which still leaves various strategies open for stabilizing the quadrotor.

# Chapter 3

# Methods

This chapter will explain the approach taken to achieve the goal of this research. Various steps will be taken in simulation and with physical drones in order to gradually work towards the autonomous stabilization of a micro quadrotor.

## 3-1 Simulation

Simulation is a powerful tool to increase testing capabilities, as it can simulate many different scenarios and strategies without the risk of damaging physical setups. Furthermore, the number of experiments that can be done in simulation also greatly outnumbers the number of experiments one can do in the same time frame on a physical setup. Using these advantages, a simulation can be ideal to generate proof of concept, or give an insight into certain trends. These trends can be used to predict the behavior or requirements on a physical setup. The simulation will be build using MATLAB and Simulink.

The goal of this simulation will be to show that strategies using the instability based approach described in Section 2-2-3 can successfully estimate the height of a micro quadrotor and use this for stabilization. In order to achieve this the following steps will be taken:

- Design a simulator based on assumptions and simplifications
- Verify the working of the simulator by implementing the vertical control strategy described in [1]
- Extend the control strategy to the horizontal axis
- Show control strategies implementing a takeoff and stable hover maneuver

When the steps mentioned above have successfully been completed, it is important to investigate the effects of scaling down to a micro quadrotor. This will be done by looking at two properties, first of all the physical properties such as dimensions, mass and moment of inertia. Next to scaling the physical properties of the simulated quadrotor the computation power on board also scales down as mentioned in Section 1-2. This will be investigated by taking a look at the most computational expensive part, the vision computations. Summarizing:

- What is the influence of scaling the physical properties of a quadrotor?
- What is the influence of varying effective FPS in the vision algorithm?

## 3-2 Experiments

Before starting experiments on small quadrotors, it will be useful to start on a larger drone, a more stable platform, with more computational power to show proof of concept. This will also allow us to look at the effects of scaling and compare them to the simulation results. Therefore the experiments will be performed on an ARDrone 2, a toy quadrotor manufactured by Parrot SA.

The steps that will be taken to achieve the goal of this thesis are similar to the simulation steps:

- Implemente the vertical control strategy described in [1]
- Extend the control strategy to the horizontal axis
- Show control strategies stabilizing the quadrotor

When the last step is achieved, the quadrotor will be able to stabilize in an unknown environment using only an IMU and a single camera. However, it still won't be the smallest autonomous quadrotor in the world. To achieve that, smaller quadrotors will have to be used to perform the same experiments with the same strategy:

- Scale down to smaller quadrotors

Suitable candidates for this scaling are the Bebop 1 and the Airborn Night *Blaze* drone, both by Parrot. Where the Bebop is a bit smaller than but has a similar weight compared to an ARDrone 2, the Blaze is both quite a lot smaller and lighter. The properties are compared in Table 3-1. Note that the camera properties are those as advertised by Parrot SA. The main advantage of using these drones is they all run the similar type of Linux computer, allowing for easy code compatibility between drones. Paparazzi UAV, an open-source autopilot in development by Delft University of Technology among others, will be the basis of this code.

**Table 3-1:** Properties of 3 Parrot SA quadrotors with indoor bumper or hull

| Physical quantity | ARDrone 2 | Bebop 1 | Blaze |
|---|---|---|---|
| Length $\times$ Width | $57 \times 57$ cm | $38 \times 33$ cm | $18 \times 18.5$ cm |
| Height | 12.7 cm | 8.5 cm | 4 cm |
| Weight | 460 g | 420 g | 63 g |
| Bottom camera resolution | $320 \times 240$ px | $640 \times 480$ px | $640 \times 480$ px |
| Bottom camera FPS | 60 FPS | 60 FPS | 60 FPS |

## 3-3 Control strategy

In this section the primary control strategy that will be used in both the simulation and the quadrotors is presented. This is the same adaptive gain strategy as in [1], where the gain is increased until the hovering quadrotor starts to oscillate. This oscillation is then detected, the gain decreased a bit, hence the quadrotor will stabilize. As explained in Section 2-2-3 the gain at which oscillations occur is dependent on the height the quadrotor is hovering at. Algorithm 1 shows the pseudo code for the strategy.

It is worth noting that this method is similar to one of the Ziegler-Nichols PID tuning methods[15], the ultimate gain tuning method. This manual PID tuning method sets PID gains all to zero and increases the P gain until the system becomes marginally stable, i.e. begins to oscillate. The P gain for which this happens is called the ultimate gain $K_u$. Furthermore, the period of the oscillation, $T_u$, should be measured. These two values can be used to construct a PID controller with different properties depending on the desired controller type. For example in case a P controller is desired $K_p = 0.5K_u$, or for a PD controller $K_p = 0.8K_u$ and $K_d = T_u/8$.

However, as the gain $K_u$ from the Ziegler-Nichols method is height dependent in the case of a quadrotor, as shown in Section 2-2-3, it is not sufficient to determine this once by hand. Instead it is done adaptive while flying. Furthermore, the I gain is not set to zero during the period where the P gain is increased as the quadrotor already drifted substantially with $K_i = 0$.

**Algorithm 1** Pseudo code for vertical loop PI control in hover

```
1:  while true do
2:      if new measurement $Div$ from vision module then
3:
4:          $error = -Div$
5:
6:          if $oscillating \neq$ TRUE then
7:              increase gain: $K_p += a \cdot dt$
8:
9:              if COVARIANCE($Div$,$input2$) $> threshold$ then
10:                 $oscillating =$ TRUE
11:                 Reduce $K_p$ to stabilize: $K_p = \alpha K_p$
12:             end if
13:
14:         end if
15:
16:         $e_{sum} += error$
17:         set $thrust = K_p \cdot error + K_i \cdot e_{sum}$
18:     end if
19: end while
```

Note that *input*2, the signal that is used to compute the covariance with the divergence with, is the given thrust input signal, in [1]. However, it can also be the divergence signal itself, delayed an $X$ number of samples. The latter case, called the autocovariance, can be used to detect a certain frequency of oscillation in the divergence itself, based on the delay. However, when the quadrotor would go op and down as a result of a external disturbance such as the wind, an oscillation would show up in the autocovariance.

If the covariance of the divergence and the commanded input thrust is taken as a measure instead, the oscillations that are detected are self induced can be differentiated and used as the trigger for the algorithm. It must be noted that the window length is important as it should be long enough to reduce noise, but not too long to avoid high computation times and memory allocation. Ideally it should contain at least one oscillation period.

When applying this method to the horizontal axes, the trust can no longer be used for *input*2. Instead the effective thrust in the respective axis could be used, or more simplistic the desired pitch or roll angle. Section 6-3-2 shows this measure is suited for the horizontal axes.

# Chapter 4

# Simulation

As described in Section 3-1, the first step in this thesis will be to build a simulation in MATLAB where the control strategies can be developed and tested. For this the accompanying graphical environment Simulink will be used, where signals can easily be routed between standard and custom blocks, and plotted real time.

## 4-1 Assumptions

In order to design the simulation assumptions have to be made. In this section all assumptions will be listed and explained.

### 4-1-1 Quadrotor

The following assumptions have been made for modeling and scaling a quadrotor in simulation

- The simulation will be in 2D
- The 2D quadrotor will be modeled as a rigid body with uniform mass
- Two positive thrust forces will act perpendicular to the body at the edges facing up
- The outputs of the controller, thrust forces, will be discretized by a ZOH at 512 Hz
- Zero Mean White Noise (ZMWN) is added to these thrust forces
- The maximal total thrust is two times the quadrotors weight
- The simulated quadrotor will be scaled uniformly with an ARDrone 2 as basis of the simulations, at scale 1

For simplicity the simulation will be limited to 2 dimensions, the vertical Z dimension and the horizontal X dimension. The other horizontal dimension, Y, is left out as this case is very similar, if not symmetrical, to the X dimension.

Though many quadrotors have the majority of their mass in the center body, and the actuators on small beams in a cross formation, the 2D quadrotor is simplified to a single rigid beam, with actuating forces acting on the edges. The underlying motor mechanics are not modeled, but it is assumed the propellers can only spin in one direction and can therefore only a positive thrust can be generated. The quadrotor is assumed to be able to provide enough thrust to carry twice its weight.

To account for imperfections in the motors and rotors, ZMWN signals are added to each actuating force individually. The ZOH represents the 512 Hz frequency of the main loop of Paparazzi.

The starting point of the simulations will be a quadrotor based on the ARDrone 2. This quadrotor will be scaled uniformly which implies that all parts scale and their properties scale uniformly too. In reality these parts don't scale uniformly however, instead different types of motors, batteries, cameras, etc will have to be used, probably leading to a differences in the exact results in comparison to the simulations. The trend however should be visible.

A picture of the 2D quadrotor can be seen in Figure 4-1.

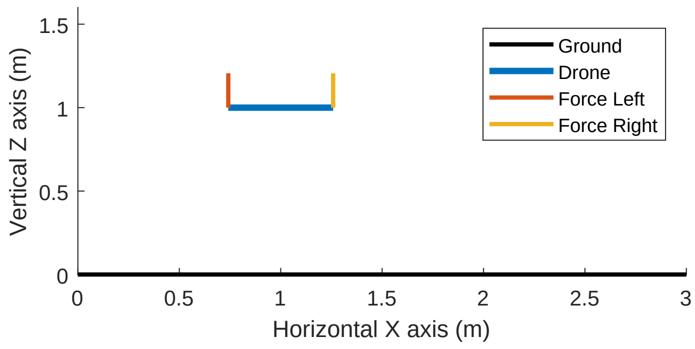


**Figure 4-1:** A 2D version of the ARDrone 2, note that the thrust forces are divided by 10 for displaying purposes

## 4-1-2 Control inputs

The assumptions and implementations regarding the control inputs including the Attitude and Heading Reference System (AHRS) and simulated vision will be covered here.

Assuming an AHRS will be used on the physical setup, there is no need to implement it here. Instead to simplify the simulation the ground truth signal of $\theta$ is taken and ZMWN is added to simulate the imperfect AHRS.

- To simulate an AHRS ZMWN is added to angle $\theta$ and used by the controller
- Ventral flow and divergence will be modeled as $\dot{x}/z$ and $\dot{z}/z$ respectively

- A ZOH is used to simulate the periodic update of the vision module at the *effective* FPS
- A unit delay will represent the computation time by delaying the signal 1 sample
- ZMWN will be added to the vision signal to simulate imperfect estimations of the ventral flow and divergence from vision data

To simulate the vision data the horizontal and vertical velocity will be taken divided by the height, representing Ventral flow and Divergence. The velocities and height will be taken from the state used in the simulation. Though the camera on-board might have an update frequency of $30Hz$ for example, representing 30 FPS, the time spend computing the ventral flow and divergence might take longer than 1/30 sec, resulting in a lower *effective* FPS. Therefore to prevent unrealistic results the vision signal will have to be discretized by a ZOH at a certain frequency and delayed by a unit delay, together representing this effective FPS. During experiments the effective FPS is fixed, but it will be varied to study the effect of computational weight of vision algorithms. ZMWN is added to the vision signals before the ZOH, representing the estimation errors made by the vision algorithms.

## 4-2 Model

In this section the model of the simulated quadrotor will be made, taking in account the assumptions made in Section 4-1. Looking at Figure 4-1, the X and Z axis are positive going right and up respectively. Furthermore, the ventral flow $\omega$ and the divergence $Div$ are thus defined as positive going right and up, using the following formulas

$$\begin{aligned} \omega &= \frac{\dot{x}}{z} \\ Div &= \frac{\dot{z}}{z} \end{aligned} \tag{4-1}$$

The input $u(t)$ of the system is defined as

$$u(t) = (F_1, F_2) \tag{4-2}$$

For simplicity the forces $F_1, F_2$, are transformed into a thrust vector, $F$, acting on the Center of Mass (CoM) and a moment, $M$, in the CoM, with $w$ the width of the quadrotor

$$\begin{aligned} F &= F_1 + F_2 \\ M &= F_1\frac{w}{2} - F_2\frac{w}{2} \end{aligned} \tag{4-3}$$

The equations of motion can derived as follows

$$\begin{aligned} \ddot{x} &= \frac{F}{m}\sin(\theta) \\ \ddot{z} &= \frac{F}{m}\cos(\theta) - g \\ \ddot{\theta} &= \frac{M}{I} \end{aligned} \tag{4-4}$$

with the moment of inertia $I$ defined as

$$I = \frac{m}{12}(w^2 + h^2) \tag{4-5}$$

Now the state $q$ and derivative $\dot{q}$ can be defined as

$$q = \begin{bmatrix} x \\ z \\ \theta \\ \dot{x} \\ \dot{z} \\ \dot{\theta} \end{bmatrix} \qquad \dot{q} = \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{z} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ \frac{F}{m}\sin(\theta) \\ \frac{F}{m}\cos(\theta) - g \\ \frac{M}{I} \end{bmatrix} \tag{4-6}$$

This results in the nonlinear system, where the output for simulation purposes corresponds to the state.

$$\begin{aligned} \dot{q}(t) &= f(q(t), u(t)) \\ y_{sim}(t) &= q(t) \end{aligned} \tag{4-7}$$

The control module however won't have access to the state. Instead it uses the attitude angle estimated by the AHRS, the simulated vision consisting of the ventral flow and the divergence. As described in Section 4-1 noise is added to the signals and a ZOH is applied to the simulated vision to simulate the effective FPS.

## 4-3 Control

In this section the different control loops will be presented. First the inner loop, which steers the quadrotor to the desired attitude by determining the desired moment. Followed by the outer loop control, which can in turn be divided into two separate loops, the horizontal and the vertical loop. The horizontal outer loop will determine the desired attitude, the vertical outer loop will determine the desired thrust. The desired thrust and moment will then be used to determine the forces $F_1, F_2$.

### 4-3-1 Inner loop control

The inner loop controls the quadrotor towards a desired attitude, and generally it is a PID controller using the desired attitude and the current attitude. The current attitude can be taken from the estimated $\theta_{AHRS}$ or from the attitude provided by the simulator, $\theta_{sim}$ depending on the goal of the simulation. Autopilots such as Paparazzi already contain an inner loop and therefore it will be outside the scope of this thesis. However, it is still needed to implement this in the simulation. To do this a simple PD controller will be chosen

$$M = K_p \cdot (\theta_{desired} - \theta) - K_d \cdot \dot{\theta} \tag{4-8}$$

## 4-3-2 Outer loop control —Horizontal loop

The horizontal loop will have different options for the control scheme, depending on the type of test.

#### Ground truth control

The first option is to use the simulation output $y_{sim}$, representing the a ground truth system such as OptiTrack. This case is used to hover the quadrotor in one position, $\theta_{waypoint}$, for example to verify the simulation is working and the quadrotor can fly from point to point. It can also be used to keep the quadrotor steady in the horizontal loop when testing the vertical loop. A PD controller is chosen for this.

$$\theta_{desired} = K_p \cdot (\theta_{waypoint} - \theta_{sim}) - K_d \cdot \dot{\theta}_{sim} \tag{4-9}$$

#### Ventral flow control

The second option is to use the ventral flow, $\omega$, as a measure of horizontal movement. In this case ventral flow corresponds with a lateral movement, thus the lateral movement can be controlled indirectly by changing $\theta_{desired}$ with a PID controller. A desired ventral flow, $\omega_{desired}$, can be chosen, but for stabilization this will be taken as 0. Also note that the desired pitch angle is bounded using the saturation function as defined in Equation 4-10. Next to a P gain, an I gain is added to add some position recovery and overcome small drifts. The $K_p$ gain will be used in a similar fashion as the algorithm in Section 3-3.

$$\text{sat}(min, x, max) = \begin{cases} min & \text{for } x \leq min \\ max & \text{for } x \geq max \\ x & \text{otherwise} \end{cases} \tag{4-10}$$

$$\theta_{desired} = \text{sat}(-\theta_{max}, \quad K_p \cdot e + K_i \cdot e_i, \quad \theta_{max}) \tag{4-11}$$

With

$$e = \omega_{desired} - \omega$$
$$e_i = \sum e$$

$\theta_{desired}$ is then fed into the inner loop control.

## 4-3-3 Outer loop control - Vertical loop

Similar to the horizontal outer loop, the vertical loop can also be divided into multiple options, including one using the simulation output and a vision based one using the divergence $Div$. However, there is an additional option, being the feed forward takeoff to simulate takeoff and stabilization behavior.

### Feed forward takeoff control

The parameters for this maneuver are the duration, and the maximum thrust. In the simulations the feed forward thrust is taken as the following formula, with simulation time $t$, feed forward time $t_{ff}$ equilibrium force $F_e$ and max thrust $F_{max}$.

$$F_{ff}(t) = \begin{cases} F_{max} - (F_{max} - F_e) \cdot \dfrac{t}{t_{ff}} & \text{for } 0 < t <= t_{ff} \\ F_e & \text{for } t_{ff} < t <= 2 \cdot t_{ff} \end{cases} \tag{4-12}$$

A value of 0.3 seconds is taken for the feed forward time, $t_{ff}$ with an $F_{max}$ of $2 \cdot F_e$, to the profile as can be seen in Figure 4-2.

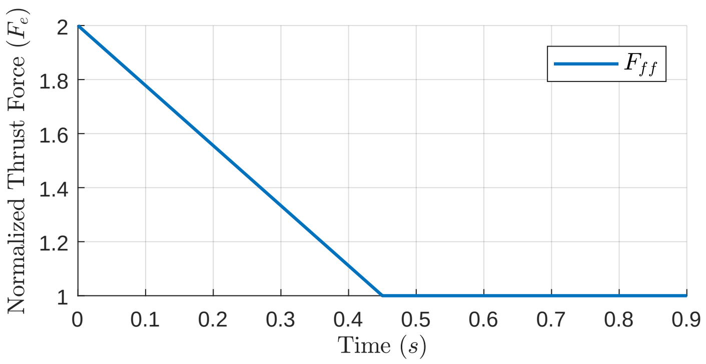


**Figure 4-2:** The feed forward thrust $F_{ff}$ decreases linearly from $F_{max}$ to the $F_e$ in $0.3$ seconds, followed by $0.3$ seconds of $F_e$.

The quadrotor is now at a certain height with a certain speed and should slow down. This is done using a P controller on the divergence $Div$ as follows, note that the output is bounded between $\frac{F_e}{4}$ and $F_{max}$, these bounds were chosen to ensure a gradual deceleration.

$$F = \text{sat}(\frac{F_e}{4}, \quad K_p \cdot e, \quad F_{max}) \tag{4-13}$$

With

$$e = Div_{desired} - Div$$

This control is used until the divergence is lower than a certain threshold, upon which the control scheme is switched to either the ground truth or the vision based divergence control.

### Ground truth control

Similar to the horizontal case, the ground truth from a system such as OptiTrack can be used in the vertical loop to keep the quadrotor in one place with a PD controller for testing purposes. Note however that there is an additional term, $F_e$, which compensates for gravity. This thrust is scaled so that when it doesn't point straight up due to the quadrotor pitching, the vertical component of the force still cancels out gravity.

$$F = \frac{F_e}{\cos\theta} + K_p \cdot (Z_{waypoint} - Z_{sim}) - K_d \cdot \dot{Z}_{sim} \tag{4-14}$$

### Divergence control

Divergence control can be used with a non zero divergence setpoint to simulate a landing with proportional decreasing speed with respect to the height, or a hover using a zero divergence setpoint. The following PI controller is used, with the same gravity compensation as in Equation 4-14

$$F = \frac{F_e}{\cos\theta} + K_p \cdot e + K_i \cdot e_i \tag{4-15}$$

With

$$e = Div_{desired} - Div$$
$$e_i = \sum e$$

The $K_p$ gain will be subjected to the adaptive gain strategy as presented in Section 3-3.

## 4-3-4 Motor mixing

With $F$ and $M$ determined by the different controllers, they can be transformed into the two different thrust forces representing the motors, $F_1, F_2$. When doing this it is important to keep into account a motor can only deliver a thrust within a range from 0 to a maximum thrust of $F_{max}$. This thrust is only positive due to the underlying assumption that non reversible propellers are used.

When limiting the thrust a choice has to be made between the priority of the different actions, thrust, pitch, roll and yaw. Normally yaw would be sacrificed first as it is not essential in the horizontal nor the vertical loop. Next would be thrust, as a horizontally stable but slowly descending quadrotor is safer than an unstable quadrotor. In the 2D simulation however there is no yaw, nor roll, as the quadrotor can only translate in X and Z direction and pitch, rotate around the Y axis which is perpendicular to the simulation. Therefore the highest priority is the pitch, followed by the thrust.

The motor mixing can thus be described as follows, first the desired moment is bounded, positive or negative, by the maximum moment that can be achieved by having one motor provide 0 thrust and the other maximal thrust, $F_{max}$.

$$M_{mixing} = \text{sat}(-\frac{w}{2} \cdot F_{max}, \quad M, \quad \frac{w}{2} \cdot F_{max}) \tag{4-16}$$

This moment is divided equally over both motors with an arm of $\frac{w}{2}$ such that the forces become

$$
\begin{aligned}
F_1 &= \frac{M_{mixing}}{2} \cdot \frac{2}{w} = \frac{M_{mixing}}{w} \\
F_2 &= -\frac{M_{mixing}}{2} \cdot \frac{2}{w} = -\frac{M_{mixing}}{w}
\end{aligned}
\tag{4-17}
$$

Now the thrust is determined from the desired thrust bounded by a minimum and a maximum that ensure each motor does not supply less than 0 or more than $F_{max}$. The minimum total thrust $F_{mixing}$ therefore is $2 \cdot \frac{M_{mixing}}{w}$ to compensate for the negative $F_i$ and the maximum is $2 \cdot (F_{max} - \frac{M_{mixing}}{w})$, the leftover thrust that can be added before the motor reaches its maximum thrust.

$$
F_{mixing} = \mathrm{sat}(2 \cdot \frac{M_{mixing}}{w}, \quad F, \quad 2 \cdot (F_{max} - \frac{M_{mixing}}{w}))
\tag{4-18}
$$

Finally the thrust is also equally divided over both motors such that

$$
\begin{aligned}
F_1 &= \frac{F_{mixing}}{2} + \frac{M_{mixing}}{w} \\
F_2 &= \frac{F_{mixing}}{2} - \frac{M_{mixing}}{w}
\end{aligned}
\tag{4-19}
$$

## 4-4 Scaling

As briefly mentioned in Section 1-2 when scaling quadrotors up or down it is not evident that the behavior is similar, due to scaling laws. In this section the influence of scaling laws on the simulation will be described, following the assumptions made in Section 4-1.

With uniform scaling all three dimensions are scaled with the same factor, however physical quantities derived from these dimensions do not necessarily scale in a similar fashion. When an object is scaled by a factor $L$, lengths $x, y, z$ scale with a factor $L$ resulting in $x \cdot L$ and $y \cdot L$ and $z \cdot L$. Surfaces, which are the product of two lengths, $x \cdot y$, scale with a factor $L^2$, because $x \cdot L \cdot y \cdot L = x \cdot y \cdot L^2$.

Volume scales with a factor $L^3$. Mass therefore, assuming constant density, scales with $L^3$. Moment of inertia however scales with a factor $L^5$, as $I = \int r^2 dm$ with $r \propto L$ and $m \propto L^3$.

Next the thrust that is generated by the motors also has to be scaled, although the underlying mechanics are not modeled for simplicity as mentioned in Section 4-1. According to momentum theory or blade element theory a thrust, $F$, can be approximated during hover in the following form

$$
F = 2\rho \cdot A \cdot v^2
\tag{4-20}
$$

For scaling with a factor $L$, density of air is $\rho \propto 1$, surface of the propeller gives $A \propto L^2$. The rotor tip velocity $v$ however scales differently depending on the assumption of compressibility of the flow[16].

Before making assumptions regarding rotor tip velocity the effect on forces and moments thus far can be noted as

$$\begin{aligned} F &\propto L^2 \cdot v^2 \\ M &\propto F \cdot L \propto L^3 \cdot v^2 \end{aligned} \tag{4-21}$$

In the case of Froude scaling incompressible flow is assumed and a constant Froude number.

$$Fr = \frac{v}{\sqrt{g \cdot L}} \tag{4-22}$$

If the Froude number is constant, then $v \propto \sqrt{L}$ because $g \propto 1$. Therefore the thrust and moment should be scaled as follows

$$\begin{aligned} F &\propto L^3 \\ M &\propto L^4 \end{aligned} \tag{4-23}$$

In the case Mach scaling the flow is assumed to be compressible and the rotor tip velocity to be constant, $v \propto 1$. This leads to the following scaling

$$\begin{aligned} F &\propto L^2 \\ M &\propto L^3 \end{aligned} \tag{4-24}$$

Now the effect of scaling on linear acceleration, $\ddot{x}, \ddot{z}$ with $F = m \cdot a$ and angular acceleration, $\ddot{\theta}$ with $M = I \cdot \alpha$ can be seen

$$\begin{aligned} \ddot{x}, \ddot{z} &\propto \frac{L^2}{L^3} \propto L^{-1} \\ \ddot{\theta} &\propto \frac{L^3}{L^5} \propto L^{-2} \end{aligned} \tag{4-25}$$

This analysis is also roughly covered in [17], however I believe they have made an error in the final Mach scaling for linear acceleration, where they forgot the minus sign in $L^-1$. For this MSc thesis the Mach scaling is selected due to the compressible nature of air. The scaling is summarized in Table 4-1.

Now Equation 4-6 can be updated in the following way

$$q = \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ \frac{F \cdot L^2}{m \cdot L^3} \sin(\theta) \\ \frac{F \cdot L^2}{m \cdot L^3} \cos(\theta) - g \\ \frac{M \cdot L^3}{I \cdot L^5} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ \frac{1}{L} \cdot \frac{F}{m} \sin(\theta) \\ \frac{1}{L} \cdot \frac{F}{m} \cos(\theta) - g \\ \frac{1}{L^2} \cdot \frac{M}{I} \end{bmatrix} \tag{4-26}$$

**Table 4-1:** Scaling for physical quantities

| Physical quantity | Symbol | Scaling |
|---|---|---|
| Lengths | $L$ | $L^1$ |
| Surfaces | $A$ | $L^2$ |
| Volume, Mass | $V, m$ | $L^3$ |
| Moment of Inertia | $I$ | $L^5$ |
| Forces | $F$ | $L^2$ |
| Moments | $M$ | $L^3$ |

It can be concluded that smaller quadrotors have significantly faster dynamics in both linearly and especially angular accelerations.

### 4-4-1 Scaling compensation

To prevent the scaling effects from changing the behavior of the quadrotor, the control parameters can be scaled in such a way that they cancel the scaling effects where possible. Therefore the thrust $F$ and moment $M$ should be inversely scaled in comparison to Equation 4-24.

To achieve this the parameters $K_p, K_i, K_d$ should be scaled with $L^2$ and $L$ respectively, looking at Equations 4-8 for the moment and 4-13 till 4-15 for the thrust. Note that the equation for the desired pitch angle $\theta_{desired}$ remains the same.

The effects of scaling, with and without gain scaling compensation will be shown in Section 6-1-3.

## 4-5 Implementation

In this section the design of the Simulink model is presented. In Figure 4-3 the global scheme can be seen. The link to the Github repository containing this file and the accompanying MATLAB code can be found in Appendix B.

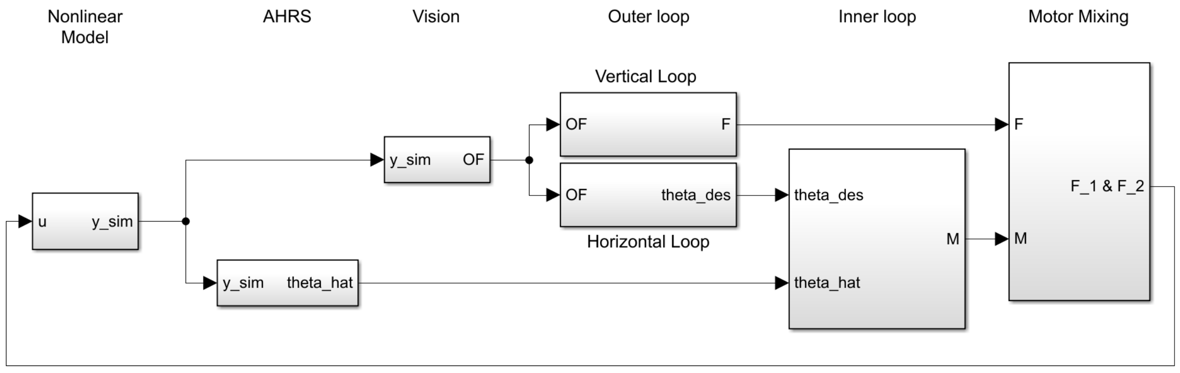


**Figure 4-3:** The core functions from left to right: the nonlinear model, the AHRS, the vision, the vertical and horizontal outer loop, the inner loop, and finally the motor mixing block.

Starting at the nonlinear model itself, this block takes the motor commands $u$ as an input, containing the individual thrust forces $F_1, F_2$, and outputs the state $q$ of the quadrotor as in Equation 4-6 as output $y_{sim}$. The internals of the block can be seen in Figure 4-4. This block represents set of equations 4-7, where the Interpreted MATLAB Function block calls a function which computes $\dot{q}$, the derivative of the state, as in Equation 4-6. The integrator has initial condition $q_0$ and gain $C$ is an identity matrix.

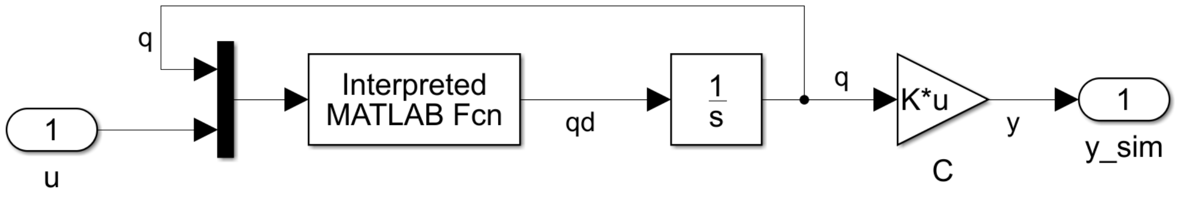


**Figure 4-4:** The nonlinear model block

This output $y_{sim}$ is used among others in the AHRS block, which simulates the AHRS implemented in Paparazzi. As can be seen in Figure 4-5, this is done by adding white noise to the $\theta$ signal, assuming the AHRS in Paparazzi can estimate the true value of $\theta$ up to a noisy error. Finally before being outputted as $\hat{\theta}$, the signal is put through a ZOH, at 1/512 seconds to simulate the Paparazzi main loop.

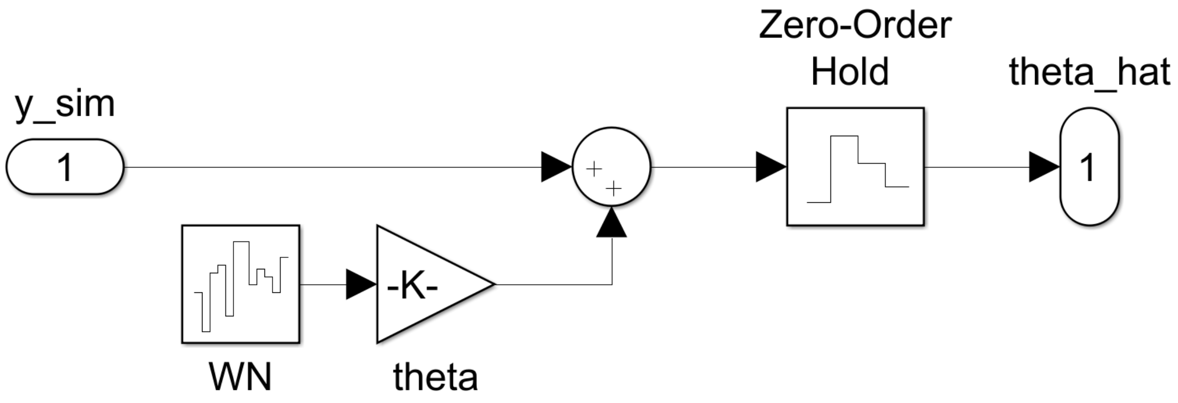


**Figure 4-5:** The AHRS block

Figure 4-6 shows the vision block which takes $y_{sim}$ as an input to simulate the OF output from the camera sensor on the quadrotor. As described in Section 4-1-2, the ventral flow is computed by dividing the horizontal speed by the height, $\omega = \dot{x}/z$ and similarly the divergence $Div = \dot{z}/z$. Next white noise is added and the signals are put through a ZOH at 1/30 seconds for example, to simulate 30 effective FPS. Finally the signals are delayed by a unit delay to make sure the results are only available after a single computation step.

Next are the horizontal and vertical loops, who form the outer loop together as seen in Figure 4-3. These blocks take the ventral flow and divergence as an input and both call a Interpreted MATLAB Function with the MATLAB code for the respective loop. This code contains the control as described in Sections 4-3-2 and 4-3-3. Furthermore, it contains the adaptive gain strategy explained in Section 3-3, which is turned on or off depending on the test that is being performed. The outputs of the horizontal and vertical loop are the desired pitch angle $\theta_{des}$ and the thrust force $F$ respectively.

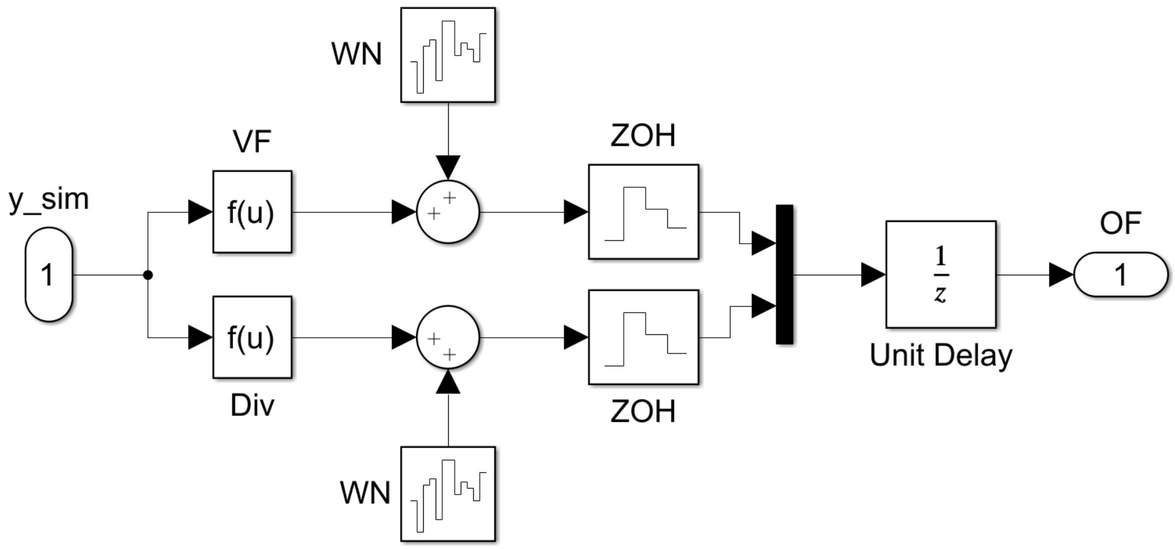


**Figure 4-6:** The vision block

The inner loop block in figure 4-3 just contains Interpreted MATLAB Function call to the inner loop as described in Section 4-3-1. It takes the desired pitch angle $\theta_{des}$ from the horizontal loop and the estimated current pitch angle $\hat{\theta}$ from the AHRS and outputs the stabilizing moment $M$.

Finally the motor mixing block, Figure 4-7 which uses another Interpreted MATLAB Function call to combine the thrust force and stabilizing moment inputs, $F, M$ and implements the motor mixing rules as explained in Section 4-3-4 followed by a ZOH at 1/512 to simulate the Paparazzi main loop. Before the signals are outputted, ZMWN is added to each individual thrust forces, resulting in outputs $F_1, F_2$.

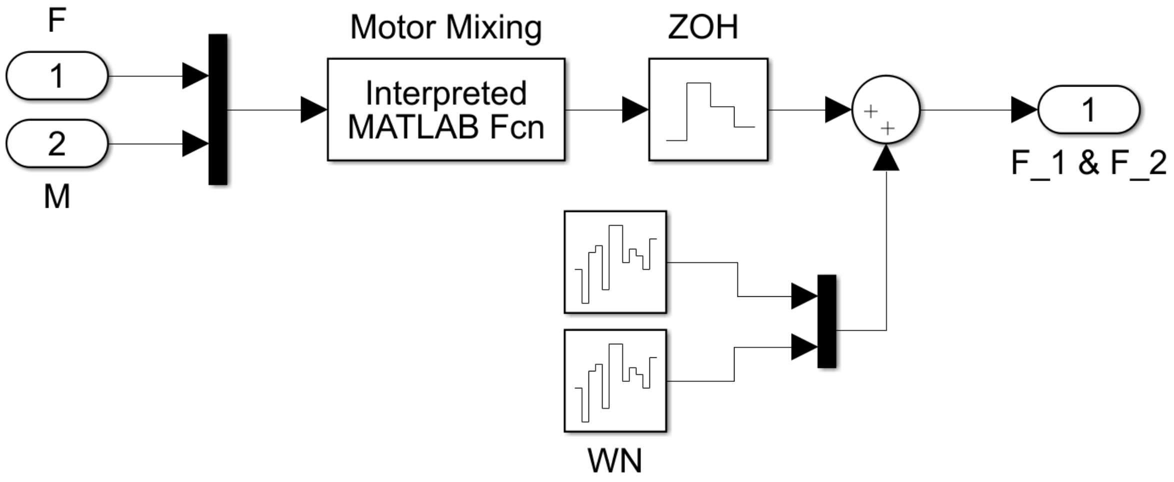


**Figure 4-7:** The motor mixing block

Note that the design covered here in this section is not actually used, instead optimizations in favor of computation efficiency are made. The Interpreted MATLAB Function calls from the motor mixing, inner loop and both outer loops are combined in one control Interpreted

MATLAB Function call. Furthermore, output blocks are inserted to log data for plotting purposes. Finally the use of scopes allows the observation of signals while the simulation is still running to aid in design and tuning.

# Chapter 5

# Experiments

Next to building a simulator experiments on quadrotors will be performed. As mentioned in Section 3-2 these experiments will be performed on an ARDrone 2 running Paparazzi, flying in the Cyber Zoo. Section 5-1 will explain what parts of Paparazzi can be used and in Section 5-2 the environment where the experiments are performed will be presented.

## 5-1 Paparazzi

Paparazzi is the open source autopilot that is being used in the MAVlab. It is designed for both fixed wing aircraft and rotorcraft such as helicopters and quadrotors. Next to the open source hardware it also works on commercial (toy) quadrotors, the Parrot ARDrone 2, Bebop 1 and 2 are examples of this.

For these quadrotors Paparazzi already interfaces with all the hardware, and it contains modules for various tasks such as telemetry, different types of AHRS, Integrated Navigation System (INS), inner-loop control and vision libraries. With these modules in place Paparazzi is a great tool to start developing and testing on a flying system. However, with all these existing modules it is necessary to make sure define which can be used, and which have to be turned off. The following subsections will explain what each file or module does, and why it can be used or has to be excluded.

#### Telemetry, radio control and GPS

The telemetry module defines how the quadrotor communicates with the ground station, it is used for sending commands like to start or stop tests and it also allows for the logging of flight data.

The Radio Control module sets the way the quadrotor will receive remote control commands from an R/C transceiver. During this thesis a transceiver will be be attached to the ground station laptop, and in the event that manual flight is required, these commands will be send using the datalink telemetry.

The information the OptiTrack system computes can be transformed into a very accurate fake GPS signal, which is sent over the datalink telemetry. This fake GPS can then be used to control the quadrotor using the ground truth obtained by the OptiTrack system.

These modules will be used purely to aid in the process of testing and comparison to ground truth, not to aid in control during the tests.

#### Motor mixing and actuator interfacing

These modules are used for combining the 4 different commands that can be given, pitch, roll, yaw and thrust, and sending the resulting motor speeds to the actuators. The motor mixing also prioritizes commands when a motor would be commanded to operate above its maximum command.

These modules are required to fly a quadrotor.

#### IMU and AHRS

The IMU module reads data from a triad of accelerometers and gyroscopes, measuring the acceleration in three directions and the angular velocities around the three axes respectively.

The sensor readings from the gyroscopes can be integrated to obtain the attitude angles pitch, roll and yaw. However, this is only a valid estimation for a short time, as the biases are also integrated, leading to sensor drift. On the contrary, when using an accelerometer to determine the orientation by computing the gravity vector, this estimation is only accurate when averaged over a long time, due to the high frequent noise of vibrations caused by the quadrotor. Combining these two estimations in a complementary filter, a high pass for the gyro estimate, and a low pass for the accelerometer, the AHRS module estimates the attitude and heading of the quadrotor. A complementary filter is used rather than a Kalman filter with the limited computational power available on the CPU's for micro quadrotors.

Both modules will be used as they are available on micro quadrotors and aid the control significantly.

#### Stabilization

The stabilization module in Paparazzi contains the inner loop control for a quadrotor. Given a desired attitude, determined in the outer loop control, and the estimated attitude from the AHRS, it will determine the required pitch, roll and yaw trajectories to achieve the desired attitude. It is worth noting that giving a desired pitch, roll and yaw of 0 won't stabilize the quadrotor itself, though the name might suggest that. It will simply rotate the quadrotor to the estimated 0 attitude, not taking remaining velocities in account. Furthermore, the estimated attitude from the AHRS isn't completely accurate, resulting in a slight drift.

### INS

An INS computes the current position and velocity of a quadrotor relative to a known or zero initial position by combining acquired sensor information in a filter. This sensor information can for example come from integrating IMU data, periodic updates from a GPS sensor, VO, US and other sensors.

During this research the INS won't be used, however the module will be used when the quadrotor is flying on the fake GPS provided by the OptiTrack system in order to fly to the desired waypoint.

### Battery voltage

Quadrotors often use Lithium Polymer (LiPo) batteries as they have a very high energy density and can be (dis)charged fast. LiPo's can be dangerous when punctured, bursting into flames that can't be put out with water due to the nature of lithium being an alkali metal. When the batteries are being discharged too fast, they heat up and swell, with the risk of bursting too. Therefore it is important to keep track of the battery voltage when flying, to prevent damage to the battery. This module monitors the battery voltage and passes it along for

### Vision

The vision modules contain all the code required to grab an image from the camera and compute the required divergence and ventral flow. The OF module has two algorithms to compute the ventral flows and the divergence, it can either use LK or EdgeFlow (EF). Furthermore, you can set the different parameters for the respective algorithms. The maximum FPS of the camera is 60 FPS, however the vision modules only grab a new image when computations on a previous image have been finished. This results in an effective FPS of $20 - 30$, this can depend for example on how many features are found and tracked per frame and the type of algorithm run to compute OF.

It is worth noting that the mentioned FPS in Table 3-1 are not accurate for each combination of quadrotor and algorithm. For an ARDrone 2 running the LK algorithm the average FPS is about 30, where running EF runs at 46 FPS. A Bebop flies with almost 91 FPS on LK.

### Control module

This is the module where the code for this thesis will be written, for the outer loop control. It is divided in two parts, the horizontal loop and the vertical loop, that can either be set to follow the designed control method, or the ground truth from the OptiTrack system. This decoupling aids in testing the vertical Z axis separately, directly adjusting the thrust command in order to control the height.

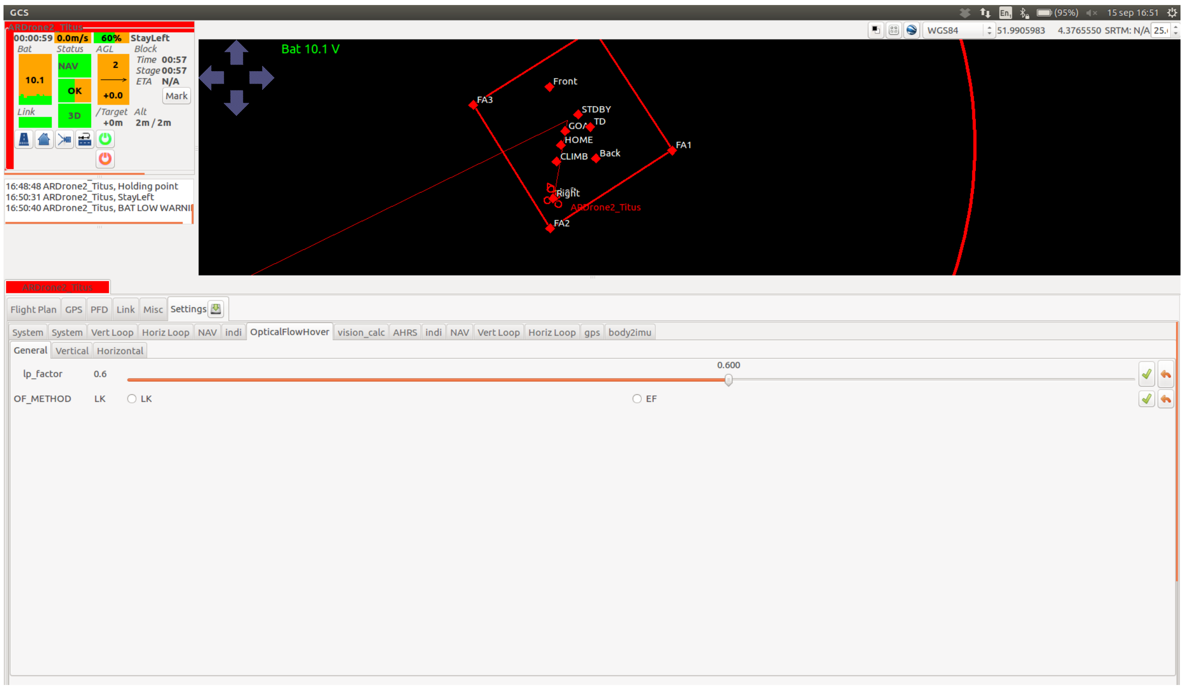
The horizontal loop however controls both the pitch and the roll commands, assuming we keep a constant heading and leave the yaw zero. To aid testing and verification from the simulation results a way should be added to use the ground truth for control in one of the two horizontal axis. This way the algorithm can be tested in both axis, X and Y separately. The pitch and roll commands are given in desired attitude angles, which the inner stabilization loop then uses as reference attitude.

### Airframe, flight plan and Ground Control Station

The airframe file configures the aircraft by telling the compiler what the firmware configuration should be. Here you can add the modules mentioned above, change settings for these modules and set calibration values for the AHRS or IMU for example. Furthermore, the gains for the inner loop are set here. Finally the mode configuration is defined so you can use a controller to take over control, or switch from one mode type to another. Paparazzi has various modes, but during these experiments only two will be used: AP_MODE_NAV and AP_MODE_MODULE. The MODULE mode will run the code that has been written for this thesis while NAV mode puts the quadrotor in control of the flight plan.

The flight plan mainly consists of waypoints and blocks. Each block corresponds to a certain state or task, such as Start Engine, which unkills the motors and allows for takeoff. I will use a block that makes the quadrotor fly towards a waypoint which is situated at the location of the experiment and keep it there. Furthermore, there are exceptions which can take over control once the battery goes low, or the link with the ground station is lost for example. In both cases the quadrotor will be commanded to start landing.

During flight the pilot can interface with the quadrotor by using the Ground Control Station (GCS). As can be seen in Figure 5-1 the GCS has a map of the experiment location with waypoints; furthermore, module settings can be adjusted on the fly to allow for easy experimentation.



```
16:48:48 ARDrone2_Titus, Holding point
16:50:31 ARDrone2_Titus, StayLeft
16:50:40 ARDrone2_Titus, BAT LOW WARNI
```

```
lp_factor    0.6
OF_METHOD    LK    LK    EF
```

**Figure 5-1:** The Ground Control Station of Paparazzi

## 5-2 Experiment environment

The experiments will be done in the Cyber Zoo at the faculty of Aerospace Engineering at Delft University of Technology, see Figure 5-2. The Cyber Zoo is a safe indoor environment for quadrotors to fly, with an OptiTrack motion capture system installed to acquire ground truth measurements. The system consists of 10 high speed cameras that track the IR light emitted from the camera units and reflected from markers added to the drone. It then computes the orientation and position of the drone with respect to the Cyber Zoo. This ground truth can also be used for control in early stages of the development, for example to stabilize the drone in the XY or YZ plane while doing tests in the Z axis or X axis respectively.

Though the grass in the Cyber Zoo has some features, it can still prove to be tough to get a good divergence or ventral flow measurement on. To aid with this, while respecting the requirements as described in Section 1-3, play mats with sufficient texture are put on the ground.

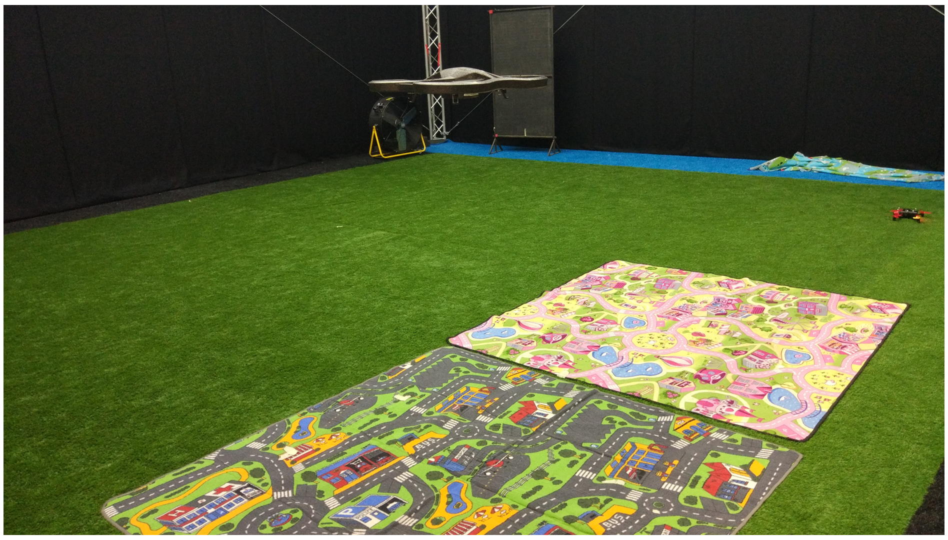


**Figure 5-2:** The ARDrone 2 flying in the Cyber Zoo, with play mats for additional features

## 5-3 Control

Similar to Section 4-3 this section what control will be used in the quadrotor in flight. The inner loop is already part of Paparazzi as a PID controller and has already been tuned for each quadrotor respectively.

As for the horizontal and vertical outer loops, they are either controlled by Paparazzi using the fake GPS signal from OptiTrack, or the module written for this thesis using the adaptive gain strategy as presented in Section 3-3. In case OptiTrack is used a PID controller will stabilize the drone in the requested axes.

Otherwise a PI controller is used similarly to the case in simulation. For the horizontal axes the desired pitch or roll angle is determined using the ventral flows while for the vertical axis the thrust is computed using a feed forward term canceling gravity, $T_e$ and the divergence.

Note that the desired pitch and roll angles are bound to ±10 degrees and the thrust is limited from $0.25T_e$ to MAX_PPRZ using a saturation function as in Equation 4-10.

$$\theta_{desired} = \text{sat}(-\theta_{max}, \quad K_p \cdot e + K_i \cdot e_i, \quad \theta_{max}) \tag{5-1}$$

With

$$\begin{aligned} \theta_{max} &= 10 \\ e &= \omega_{desired} - \omega \\ e_i &= \sum e \end{aligned}$$

and

$$T = \text{sat}(0.25T_e, \quad T_e + K_p \cdot e + K_i \cdot e_i, \quad \text{MAX\_PPRZ}) \tag{5-2}$$

With

$$\begin{aligned} e &= Div_{desired} - Div \\ e_i &= \sum e \end{aligned}$$

Note that both $\omega_{desired}$ and $Div_{desired}$ are 0 for these experiments. $T_e$ is the thrust required to hover, either determined experimentally or in case the module is started from a stable hover using OptiTrack, the last thrust command send to the motors.

Chapter 6

# Results & Discussion

In this chapter the results from both the simulation and the experiments on the drones will be presented and discussed as mentioned in Chapter 3.

## 6-1 Simulation Results

As described in Section 3-1, this section will present results in a few steps. First the working of the simulator will be verified by showing that a constant gain landing results in self-induced oscillations at a certain height. Next the control algorithm will be extended to the horizontal axis starting from hover and implemented in a control strategy taking off from the ground. Finally the effect of scaling and the influence of varying FPS will be shown.

In the following experiments the following assumptions hold, unless mentioned otherwise

- The physical properties of the quadrotor are those of an ARDrone 2, defined as $scale = 1$
- The effective FPS is set to 20 Hz
- ZMWN is added to the vision, actuator and AHRS signals
- Initial conditions are 0, except for the $x$ and $z$ positions. 1 meter is taken for $x$ and $z$ is varied between 3 meters for the constant gain landing, 1 meter for experiments starting in hover and 0.1 meters for the takeoff experiments
- For oscillation detection the covariance method will be used, taking the thrust and divergence for the vertical axis and the desired angle and ventral flow for the horizontal axis
- For an effective FPS of 20 Hz the covariance window contains 30 samples

The reason 0.1 meters is taken instead of 0 meters for the initial takeoff condition is to avoid a division by 0 in the ventral flow and divergence computation.

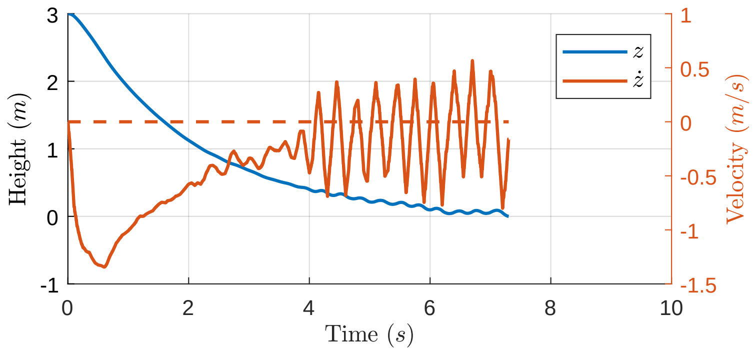
### 6-1-1 Simulator verification

Landing from a 3 meter height with a desired divergence of $-0.5$ can be done with fixed gains $K_p$ of $[10, 5, 1]$ and $K_i = 0$ as in Equation 4-15. The horizontal loop is controlled using the ground truth following Equation 4-8, as in [1].

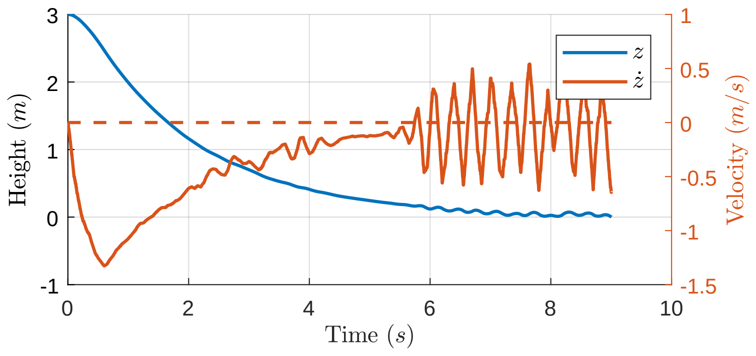
This should show a constant divergence landing until at some point the quadrotor starts oscillating. Figure 6-1 shows exactly that, where the gain influences the height at which this happens. As can be seen, a higher gain causes oscillations further from the ground. These self-induced are especially visible in the velocity axis on the right side in these graphs.

Furthermore, it can be observed that the oscillation frequency is about 3 Hz. Looking back at the covariance window size, a window of 30 samples at 20 effective FPS spans 1.5 seconds and can thus contain 4.5 oscillations. Therefore, the window contains enough information to detect these oscillations, which is supported by looking at the divergence and covariance in Figure 6-2. A threshold value can be selected for the covariance to trigger a final landing approach.

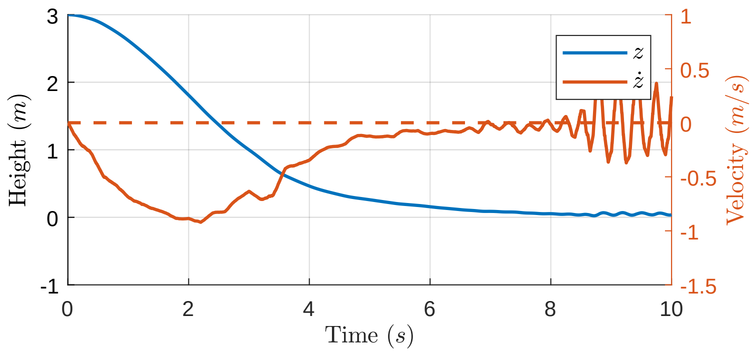
With these figures it is shown that the simulator behaves as expected. Note that the simulation in Figure 6-1b stopped when the quadrotor touched the ground to prevent a division by 0 in the vision module.



**(a)** $K_p = 10$

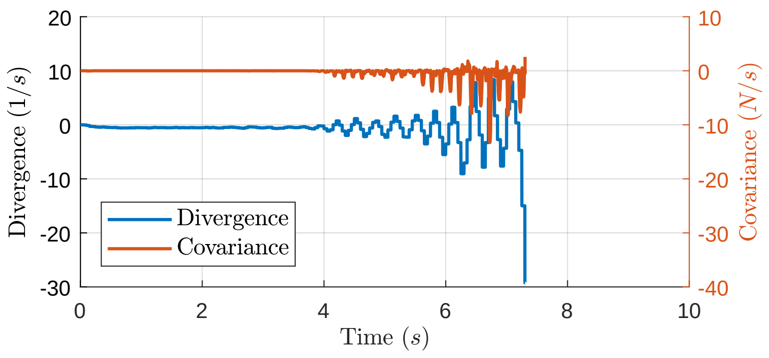


**(b)** $K_p = 5$

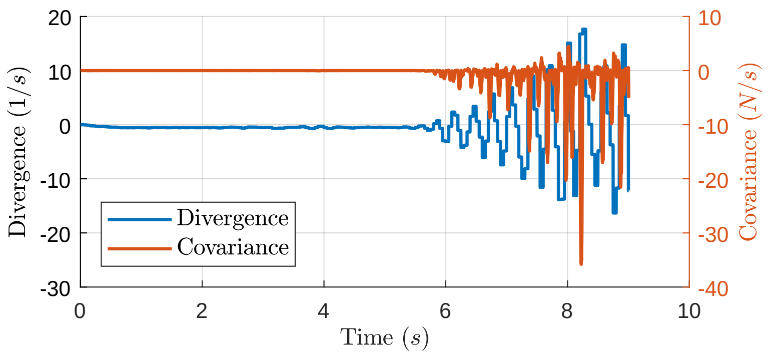


**(c)** $K_p = 1$

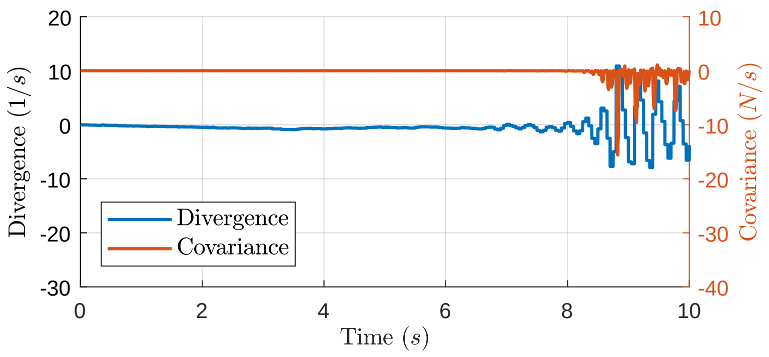
**Figure 6-1:** The height and vertical velocity of the quadrotor with $K_p = [10, 5, 1]$



**(a)** $K_p = 10$



**(b)** $K_p = 5$



**(c)** $K_p = 1$

**Figure 6-2:** The divergence and covariance of the quadrotor with $K_p = [10, 5, 1]$

### 6-1-2 Algorithm extension

In this section the algorithm is extended to the horizontal x-axis. First it is shown that when applying this algorithm starting from hover in the horizontal case it behaves similarly to the vertical axis. Next a more realistic scenario is demonstrated where the quadrotor starts with a feed forward takeoff as described in Section 4-3-3, followed by running the algorithm in both horizontal and vertical axis.

#### Hover

First to demonstrate what would happen if there was no horizontal control, the control gains $K_p, K_i$ in Equation 4-11 are set to 0. As can be seen in Figure 6-3 the quadrotor simply drifts away quite fast.

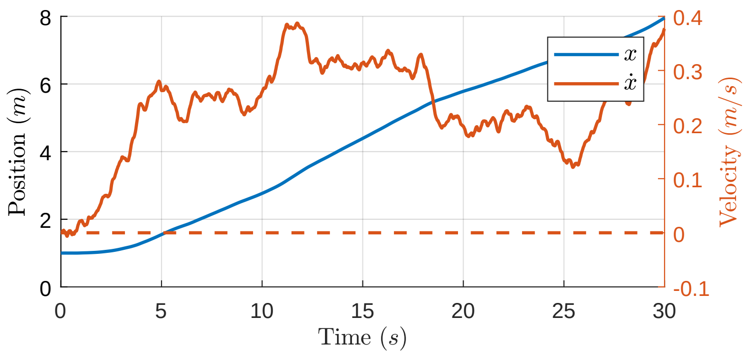
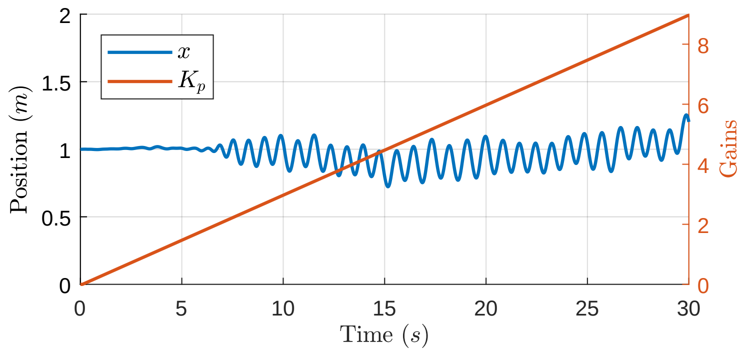
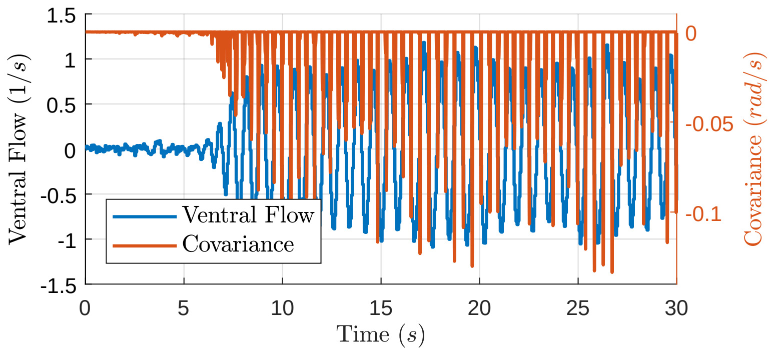


**Figure 6-3:** The quadrotor drifting away with $K_p = K_i = 0$

The following experiment shows that from hover $K_p$ is increased over time by 0.3 per second with a fixed I gain $K_i = 0.5$. Without oscillation detection the quadrotor will start to oscillate which can be seen in Figure 6-4a as the gain increases. As the oscillations begin to pick up it is clear in Figure 6-4b that a threshold can be set for the covariance to detect these oscillations.



**(a)** The position and control gain $K_p$ of the quadrotor



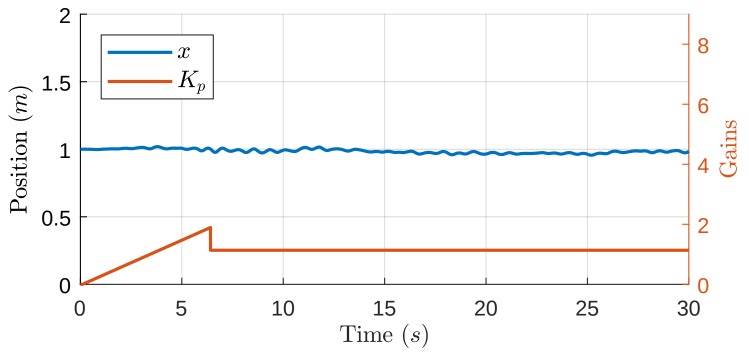
**(b)** The ventral flow and covariance of the quadrotor

**Figure 6-4:** Extension of the algorithm to the horizontal axis without oscillation detection

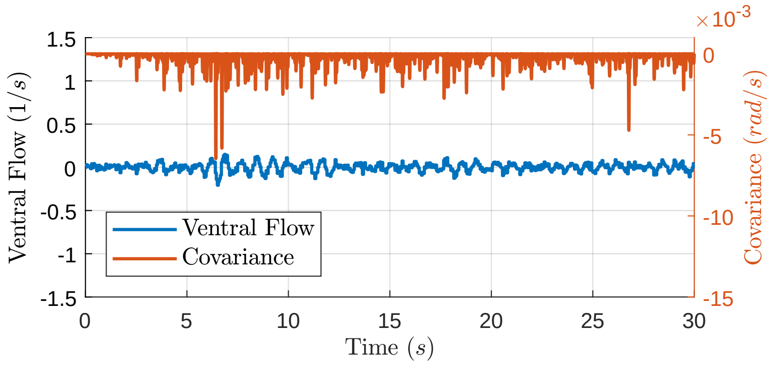
If the gain is reduced by 40% when the covariance reaches a threshold of $-6 \times 10^{-3}$, the quadrotor stabilizes. This can be seen in Figures 6-5a and 6-5b, note that the right axis of Figure 6-5b is different than that of Figure 6-4b. The values of the gains, slope, reduction and threshold are the result of tuning for a system that performs well in various cases of initial conditions.

To show that in the horizontal case the gain at which self-induced oscillations occur is also depending on the height, the same experiment is run again with a initial hover height of 2 meters instead. The results can be seen in Figures 6-5c and 6-5d. As expected it takes a higher gain for the quadrotor to begin oscillating and once it is reduced by 40% the quadrotor stabilizes.

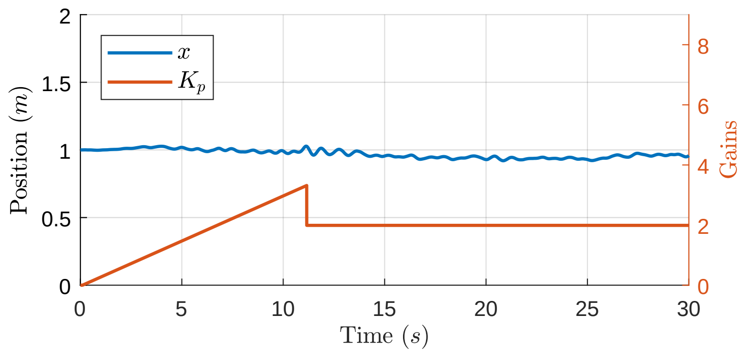
It has been shown that the algorithm can be successfully extended to the horizontal axis.



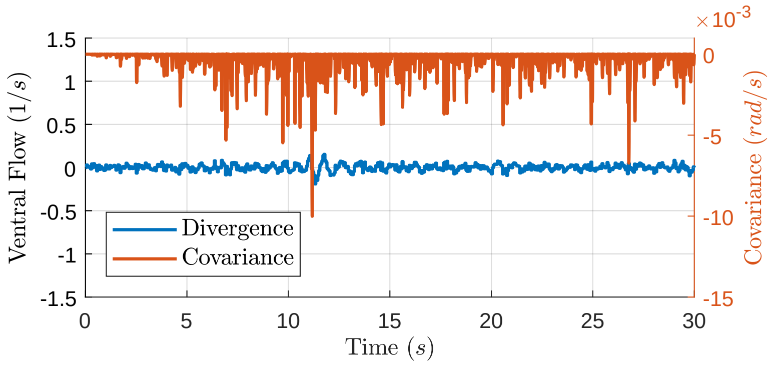
**(a)** The position and control gain $K_p$ of the quadrotor at initial height $z = 1$m



**(b)** The ventral flow and covariance of the quadrotor at initial height $z = 1$m



**(c)** The position and control gain $K_p$ of the quadrotor at initial height $z = 2$m



**(d)** The ventral flow and covariance of the quadrotor at initial height $z = 2$m

**Figure 6-5:** Extension of the algorithm to the horizontal axis with oscillation detection at initial heights $z = 1$m and $z = 2$m

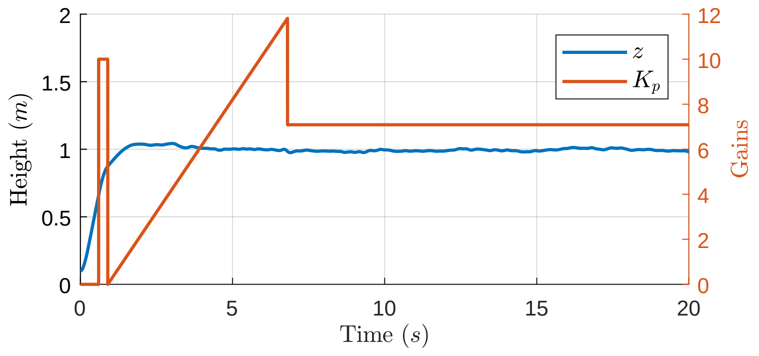
### Takeoff

Next the simulation will be altered to include the feed forward takeoff procedure and apply the algorithm to both axis at the same time. The feed forward signal will be as described in Section 4-3-3, starting from a 0.1 meter height to prevent a division by 0 in the divergence and ventral flow computations.

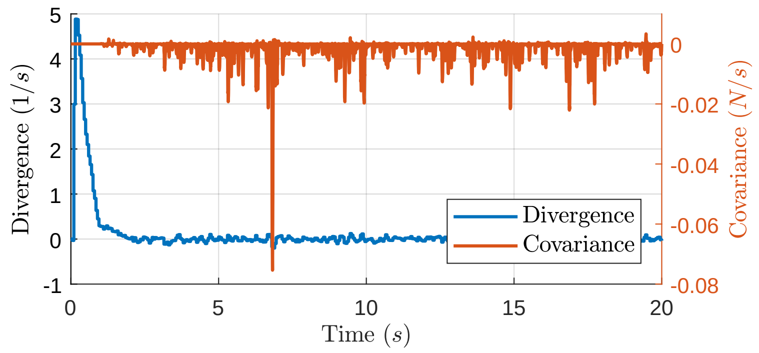
After the feed forward takeoff the quadrotor slows down to 0.5 divergence with $K_p = 10$ as in Equation 4-13. This is done because the quadrotor cannot be stabilized as the actual height is unknown, so it is slowed down to a divergence level where the controller can start without the initial drift being large. Next the algorithm is started in both the horizontal and the vertical loop. The initial slowdown gain followed by the start of the algorithm can be seen in Figure 6-6a. Note that the gain in both horizontal and vertical axis is reduced by 40% when the covariance reaches a threshold of $-6 \times 10^{-3}$ and $-4.5 \times 10^{-2}$ respectively.

Note how the height seems to stabilize at 1 meter, suggesting a form of height feedback based on an additional sensor. This is not the case, the feed forward time was simply chosen such that the quadrotor would start stabilizing around this height. The I gain then keeps it close. When setting the I gains to 0 the result shows there is no additional height sensor. Looking at the vertical axis, Figure 6-7a, and less clear in the horizontal axis, Figure 6-7c the lack of I gain allows the drifting of the quadrotor as time progresses.

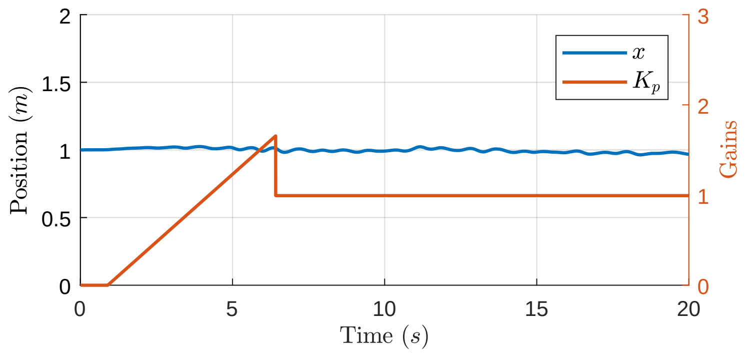
This section can be concluded now that the algorithm has been implemented in both axes following a feed forward takeoff. It shows this control method is viable for experiments on the real quadrotor.



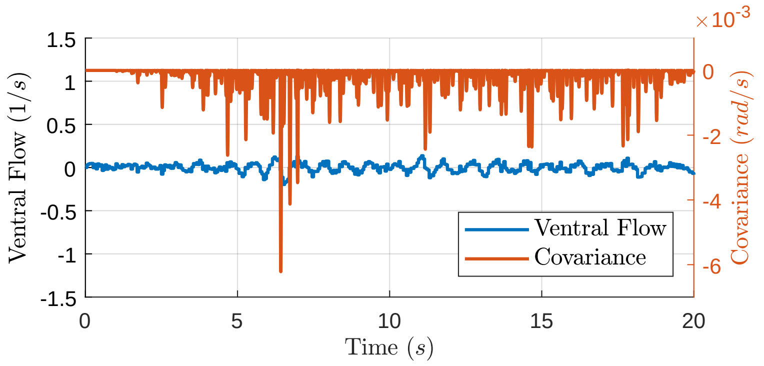
**(a)** The height and control gain $K_p$ of the quadrotor



**(b)** The divergence and covariance of the quadrotor

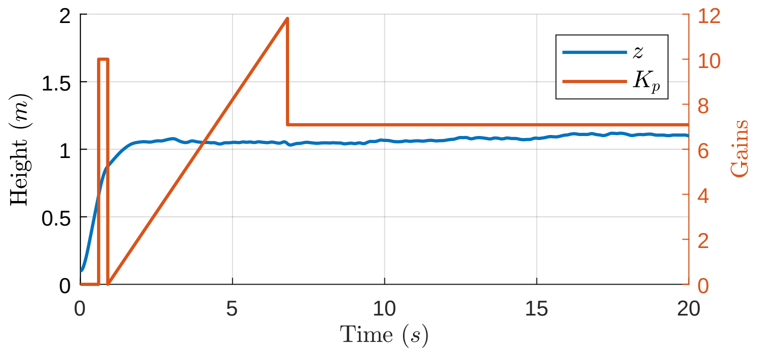


**(c)** The position and control gain $K_p$ of the quadrotor

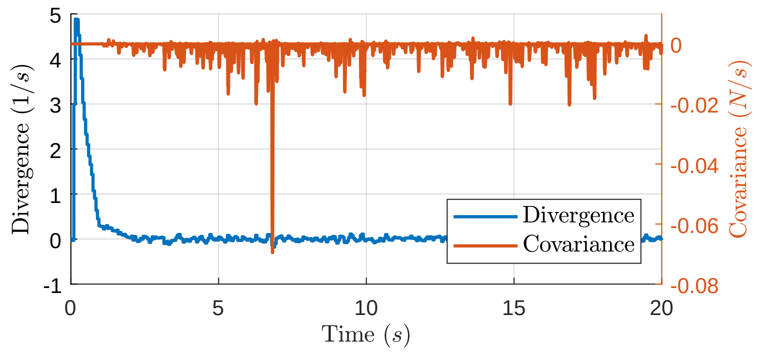


**(d)** The ventral flow and covariance of the quadrotor

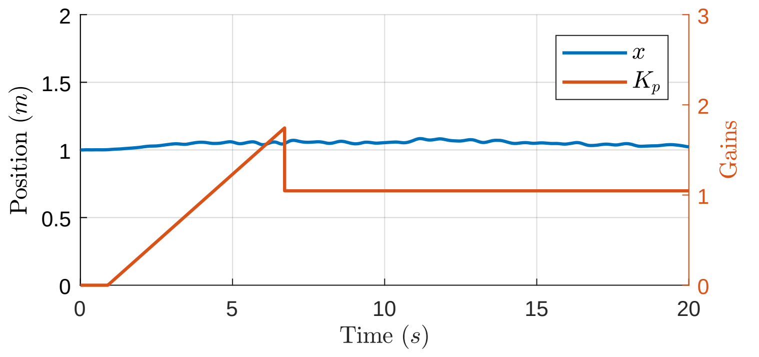
**Figure 6-6:** The quadrotor during feed forward takeoff followed by the implementation of the algorithm in both axis



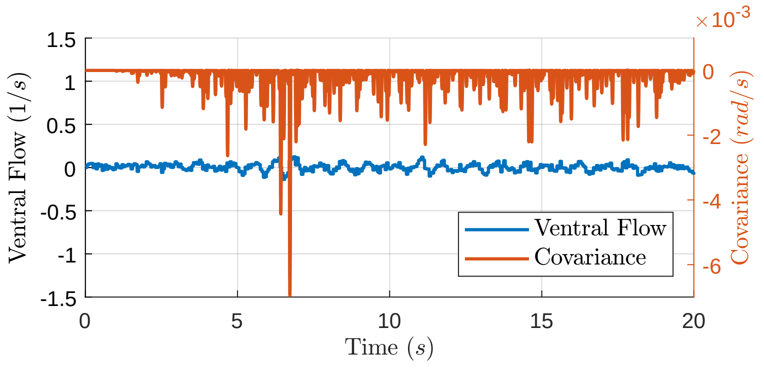
**(a)** The height and control gain $K_p$ of the quadrotor



**(b)** The divergence and covariance of the quadrotor



**(c)** The position and control gain $K_p$ of the quadrotor



**(d)** The ventral flow and covariance of the quadrotor

**Figure 6-7:** The quadrotor during feed forward takeoff followed by the implementation of the algorithm in both axis without I gains

### 6-1-3 Scaling influence

The next step is to look into the effects of scaling the properties of the quadrotor. The physical parameters width ($w$), height ($h$), mass ($m$), and the resulting moment of inertia ($I$) will be scaled according to the scaling laws as discussed in Section 4-4. Unlike the feed forward takeoff experiments in the last section the following flights will be started once again from stable hover. The algorithm will be started for both axes at the same time.

Depending on whether the compensation as mentioned in Section 4-4-1 is applied the forces and moments will be scaled differently. Table 6-1 summarizes the scaling factors, note that the uncompensated control parameters are scaled in such a way that Equation 4-26 holds. The compensated control parameters ensure the Equations of Motion (EoM) look like Equation 4-6 again.

For the moments this means the control gains $K_p, K_d$, as in Equation 4-8, will be scaled. For the forces the scaling factor is applied to $K_p, K_i, K_d$ as in Equations 4-13 till 4-15. Furthermore, the actuator noise as mentioned in Section 4-1 is also scaled, just as the slope $\alpha$ at which the vertical $K_p$ gain is increased, see Section 3-3.

Finally it must be noted that the covariance is taken of the divergence and the thrust, for the vertical direction. As the thrust is a force, the computed covariance will scale similarly. Therefore, the threshold of $-4.5 \times 10^{-2}$ should be multiplied by mass scaling factor of $L^3$.

#### Uncompensated scaling

First scaling will be applied to the quadrotor without the compensation in the control gains as mentioned in Section 4-4-1.

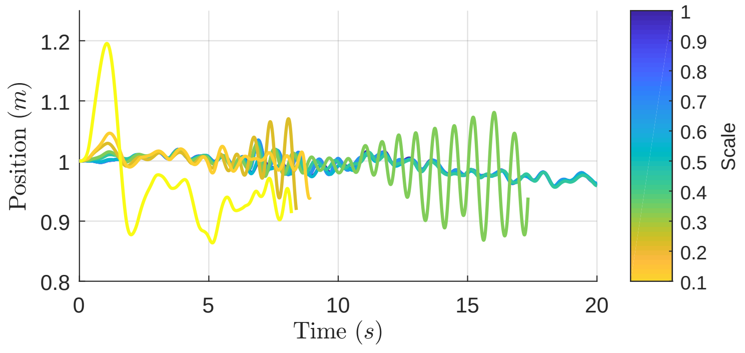
In Figure 6-8 the height and position of the quadrotor can be seen in the uncompensated case. The scale is changed from 1.0 to 0.1 in steps of 0.1 and the corresponding results are plotted in a color scale ranging from blue to yellow respectively.

Looking at the warmer lines in the vertical axis, scales 0.1 till 0.4, it shows that the quadrotor is unstable and drifts away increasingly with smaller scales, it even crashes on the ground within the selected time frame. Note that the simulation stops for a scale when the quadrotor touches the ground. The cooler colors, ranging from 0.5 to 1 show stable behavior however.

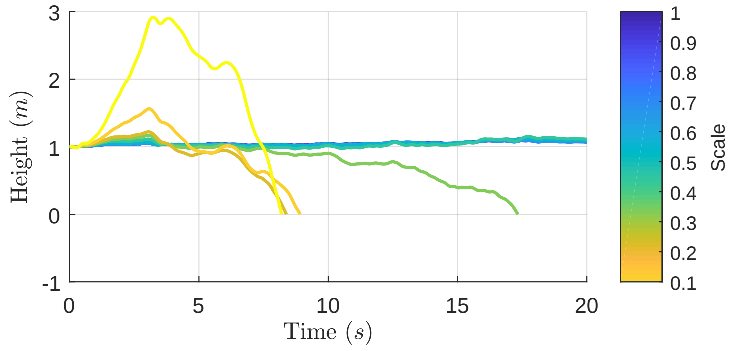
Looking at the same warmer scales in the horizontal axis, we can observe these scales start to oscillate in the x direction, whereas the cooler scales show the same stable behavior as in Section 6-1-2.

**Table 6-1:** Scaling for physical quantities

| Parameter | Symbol | Physical | Uncompensated | Compensated |
|---|---|---|---|---|
| Width, Height | $w, h$ | $L^1$ | | |
| Mass | $m$ | $L^3$ | | |
| Moment of Inertia | $I$ | $L^5$ | | |
| Forces | $F$ | | $L^2$ | $L^3$ |
| Moments | $M$ | | $L^3$ | $L^5$ |



**(a)** The position of the quadrotor for different scales

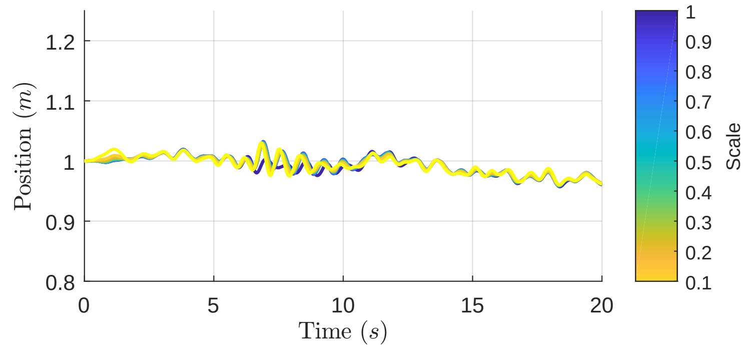


**(b)** The height of the quadrotor for different scales

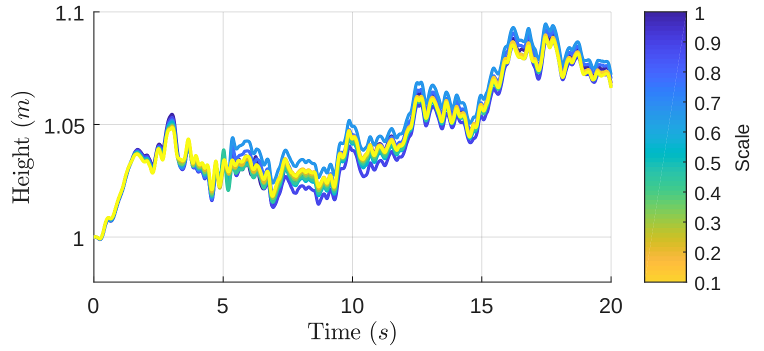
**Figure 6-8:** Examining the effects of scaling the quadrotor from scale $1$ to $0.1$ without control compensation for the scaling effects

## Compensated scaling

Now with compensation the forces and moments are scaled with $L^3$ and $L^5$ respectively as in Table 6-1, the plots in Figure 6-9 show a different result. As can be seen in both the horizontal and vertical axis, all scales are almost overlapping in stable hover.



**(a)** The position of the quadrotor for different scales



**(b)** The height of the quadrotor for different scales

**Figure 6-9:** Examining the effects of scaling the quadrotor from scale $1$ to $0.1$ with control compensation for the scaling effects

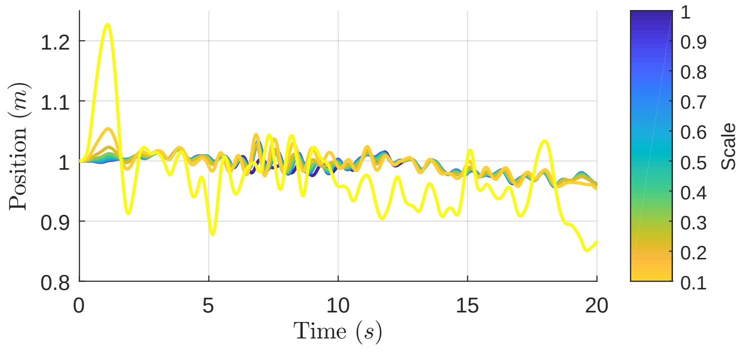
### Compensated scaling with correct noise

However, the previous results should be considered cheating as the actuator noise as assumed in Section 4-1 is not compensatable by scaling a control parameter. Scaling this parameter with $L^2$ again instead of falsely compensating by scaling it with $L^3$, shows an important difference.

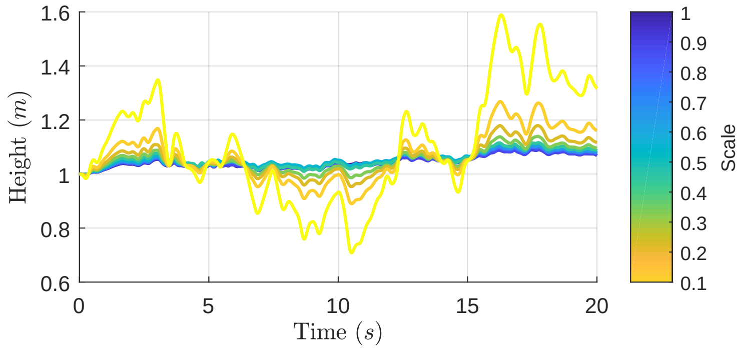
For scale 0.1 the quadrotor goes left and right uncontrollable in the horizontal axis. However, while the other scales remain stable, there is a difference with Figure 6-9 as the plots no longer overlap. Instead they all follow the same trend, with an increasing amplitude, even the 0.1 scale follows the same trend in the vertical axis.

When zooming in on the vertical axis, leaving scale 0.1 out, this effect is even more visible, see Figure 6-11. It is clear that scaling the control gains inversely to the dynamics scaling compensates for most of the effects. It can also be concluded that the noise, as can be expected, has a larger effect at the smaller scales because it cannot be compensated for by scaling the control parameters. The reason this is most clearly visible in the vertical direction is that the actuator noise works in the thrust direction of the quadrotor, which is mainly vertical while hovering.

It is also interesting to look at when the algorithm triggers in Figure 6-12. For the vertical axis the increasing gains are pictured in Figure 6-12a for the last scenario described above. Note



**(a)** The position of the quadrotor for different scales



**(b)** The height of the quadrotor for different scales

**Figure 6-10:** Examining the effects of scaling the quadrotor from scale $1$ to $0.1$ with control compensation for the scaling effects, while keeping the actuator noise uncompensated for

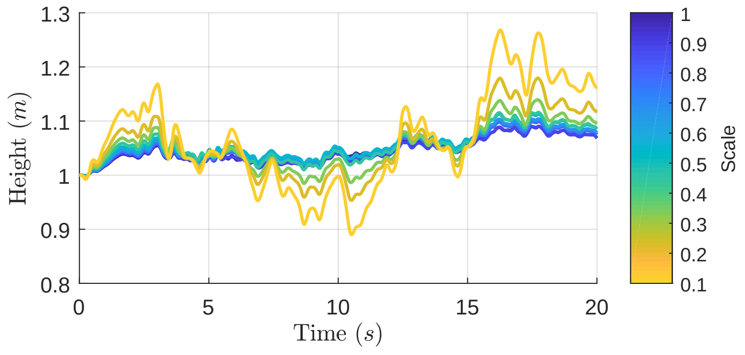
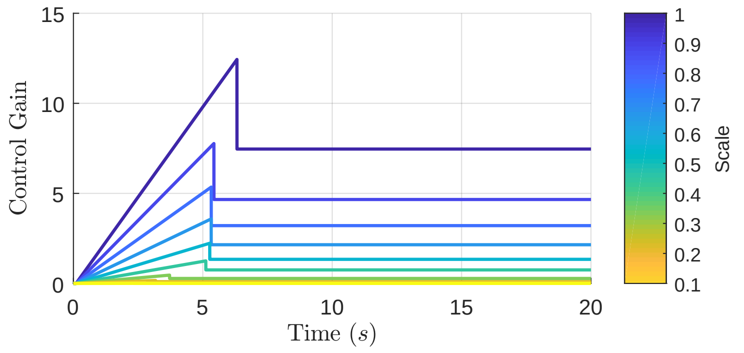


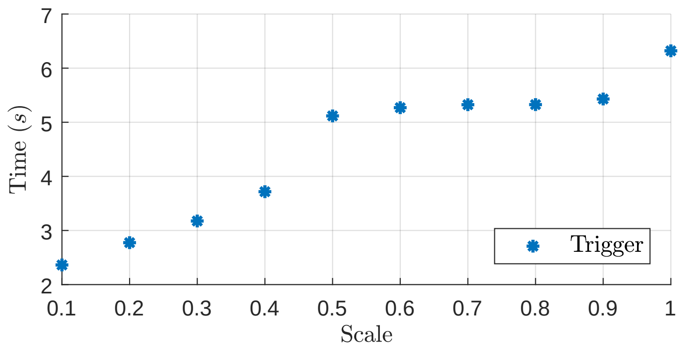
**Figure 6-11:** Closeup of Figure 6-10b without scale $0.1$ plotted

that the slope is also compensated for the scaling and thus scaled with $L^3$. The algorithm is triggered at roughly the same time for scales 0.9 through 0.5 between 5.4 to 5.1 seconds, while the gains from 0.4 to 0.1 trigger from 3.7 seconds to 2.4 seconds. Furthermore, scale 1 triggers at 6.3 seconds. Alternatively when plotting the scale against the trigger time, as Figure 6-12b shows, one could see a linear relation, with scales 0.9 to 0.5 breaking this trend. This supports the conclusion that compensating for the scaling effects with the control gains negates most of the effects, but the noise has an increasing impact on smaller scales.

Looking at the horizontal axis in Figure 6-13 however, the triggers are all around the same time, 6.6 seconds, except for scale 0.1, which trigger at 4.8 seconds. This might again be explained by the fact that the actuator noise works in the thrust direction of the quadrotor, which is mainly vertical while hovering. The overlap of the gains is because the slope of the horizontal axis isn't scaled because it influences the desired pitch angle $\theta$, which is in radians and thus not influenced by scaling. The inner loop control gains that control the pitch angle however is scaled as it has to compensate for the faster rotational dynamics.

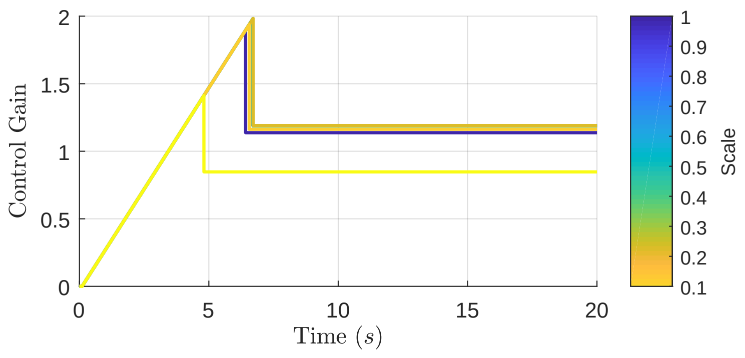


**(a)** The vertical gains of the quadrotor for different scales

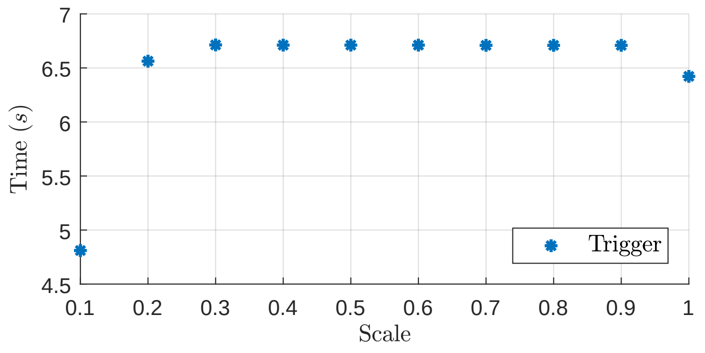


**(b)** The vertical trigger points of the quadrotor for different scales

**Figure 6-12:** Examining the effects of scaling the quadrotor from scale $1$ to $0.1$ with control compensation for the scaling effects, while keeping the actuator noise uncompensated for



**(a)** The horizontal gains of the quadrotor for different scales



**(b)** The horizontal trigger points of the quadrotor for different scales

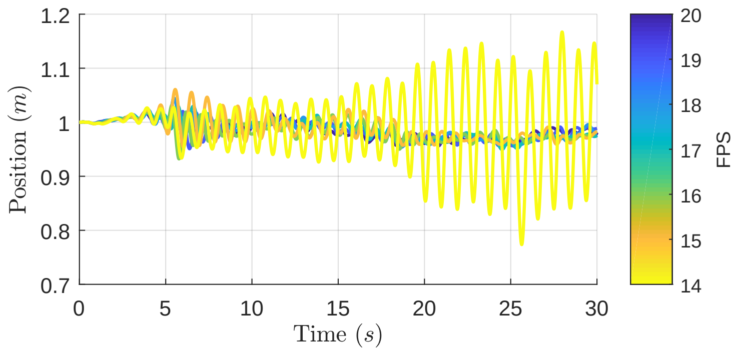
**Figure 6-13:** Examining the effects of scaling the quadrotor from scale $1$ to $0.1$ with control compensation for the scaling effects, while keeping the actuator noise uncompensated for

## 6-1-4 Influence of varying effective FPS

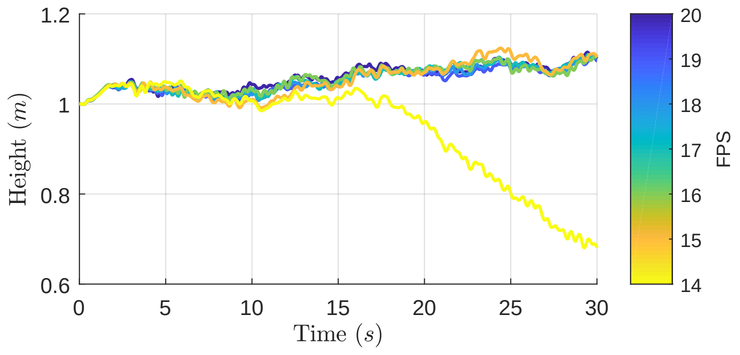
It can be interesting to look at the ARDrone 2, scale= 1 again to see what the lowest effective FPS is that will allow the quadrotor to stabilize. This would allow the quadrotor to spend computation time on other things while still remaining stable. Furthermore, continuing from the previous results, it would also be interesting to see if an increase in effective FPS would aid with stabilizing the smaller scales. It must be noted that when changing the effective FPS in MATLAB and Simulink one should also adjust the window and the delay parameters, to keep the ratio and therefore window time the same amount of seconds.

To test this the scale 1 quadrotor is set to hover at initial position and height 1. Next the adaptive gain algorithm is started in both axes for different effective FPS. Looking at Figure 6-14, where the FPS decreases from the cool colors to warm colors it is clear the quadrotor starts to oscillate in horizontal direction below 15 FPS, which probably also causes the drift in vertical direction, as the effective vertical thrust changes with the pitch angle. There is a compensation in place for this, but that is based on the estimated pitch angle by the AHRS and therefore not completely accurate.

Much more interesting is if it is possible to stabilize the quadrotor at smaller scales, as seen in Figure 6-10, with a higher effective FPS. The same experiment is performed, but for the smaller scales and higher effective FPS. Figure 6-15 shows the height of a 0.2 quadrotor, with



**(a)** The position of the quadrotor for different effective FPS



**(b)** The height of the quadrotor for different effective FPS

**Figure 6-14:** Determining the minimal FPS to fly a quadrotor of scale 1 using the adaptive gain strategy in both axis

an FPS scale from the previously used 20 FPS in yellow, to 100 FPS in blue, with steps of 8 FPS. Even though the quadrotor seems to benefit from a higher FPS, it is still not as stable as the 1 scale quadrotor was at 20 FPS. Furthermore, it seems there is an optimum, in this case 52 FPS. Figure 6-16 shows the same experiment, but without noise on the vision. The fact that it gives a better result is not surprising, but what is more interesting is that now, a higher FPS corresponds to a more stable result.

It seems that given the used actuator noise the smallest quadrotor scales can not be stabilized as well as the larger ones, even with higher FPS with the same vision noise.

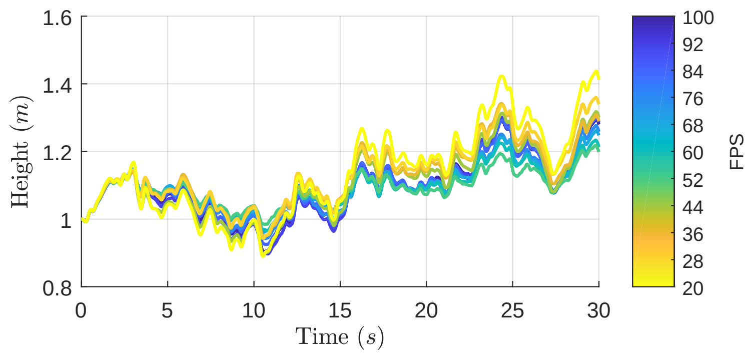


**Figure 6-15:** The height of a $0.2$ scale quadrotor with increasing FPS

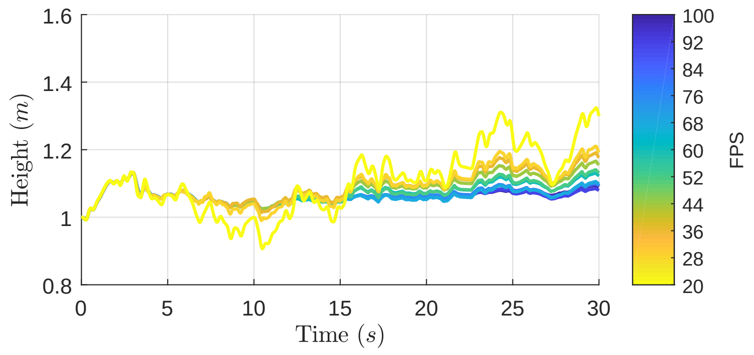


**Figure 6-16:** The height of a $0.2$ scale quadrotor with increasing FPS without noise on the vision

## 6-2 Simulation Discussion

The simulation results presented in the previous section show that the adaptive gain strategy can be used in both the horizontal axis as the vertical axis. While the results are promising, they can not be copied directly to the physical drones. Instead a few things should be noted.

- The model of a quadrotor is heavily simplified. First of all the simulation is in 2D, which eliminates an axis that could influence the other two by rotations of the quadrotor in an additional axis. Furthermore, the control is assumed to be a thrust force directly, whereas on a real quadrotor the control would have to send a Pulse Width Modulation (PWM) signal to an Electronic Speed Controller (ESC) that tries to spin the propeller at the desired RPM to generate thrust.
- The vision module, which computes the vision, only simulates the real vision outputs by adding noise and applying a ZOH at the desired FPS. On a quadrotor there are many things that could influence the result, from changes in illumination to lack of features to track. Furthermore, the rotations of the quadrotor also influence the vision, which can be solved by derotating the flow and divergence. However, this process is also not perfect. With this vision based method it is important to develop it on a quadrotor to make sure the conditions are acceptable before flying with this control method.
- ZMWN is also added to the actuator and AHRS signals. For the actuators this represents the small differences in motors and propellers, though in reality it might not have the presented influence on scaling. The AHRS signal with noise added assumes the bias is estimated properly by the AHRS leaving only the noise. In both cases the ZMWN is using the same noise seed in every experiment resulting in the same outcome every time the same experiment is run. Though this makes for consistent testing it should be considered when looking at the results.
- In Section 4-1 the scaling was assumed to be uniform, meaning that the ARDrone 2 would be scaled in such a way that a 0.1 scale quadrotor will be a mini version of the ARDrone 2. In reality, as can also be seen in Table 3-1, this is not the case. And this will most definitely influence the scaling trend found, if only by the fact that at some point certain parts can not be scaled further such as DC motors.
- The combination of the different noises prevents stabilization of the smaller quadrotors despite increases in FPS. The noise levels were meant to be chosen on the safe side, so that if they actually were lower the simulator would also work. A better selection of noise levels might have aided the conclusions in this case.

Given the mentioned points, these simulation results are only usable to detect trends and give indications whether something could work or not, not to estimate gains or give a definite answer.

## 6-3 Experimental Results

In line with Section 6-1 this section will present the results from the experiments with the quadrotors in the Cyber Zoo. As mentioned in Section 3-2, first the vertical control strategy will be implemented on the ARDrone 2 to achieve a stable hover from an unknown starting height. Next the strategy will be extended to the horizontal axis, similarly to the simulation results. Finally these will be combined to present a complete control strategy that achieves stable hover in an unknown environment using just a single camera and an IMU.

In the following experiments the following assumptions hold, unless mentioned otherwise

- An ARDrone 2 will be used with a regular battery and indoor hull
- The experiment will be started from a stable hover using the OptiTrack system
- The LK vision algorithm will be used to compute the flows and divergence
- The effective FPS will vary between 20 and 30
- The window for oscillation detection will be 100 samples given this effective FPS
- The flow and divergence inputs will be put through a low-pass filter before being used

### 6-3-1 Vertical control strategy

In this section the quadrotor will be kept stable in the two horizontal axis using the OptiTrack system while Algorithm 1 is applied to the vertical Z axis. First however, it might be interesting to see what happens when the control gains in Equation 5-2 are too low or zero.

#### Zero gains

For this the quadrotor is set to hover at 1.5 meter high on OptiTrack, before the algorithm is started in the vertical direction with zero gains. Figure 6-17 shows the quadrotor drifting up 3 meters over a period of 35 seconds. The reason the quadrotor drifts upwards is that $T_e$ might not be perfectly chosen and is slightly higher than it should. The quadrotor could also have been drifting downwards if $T_e$ was chosen too low.

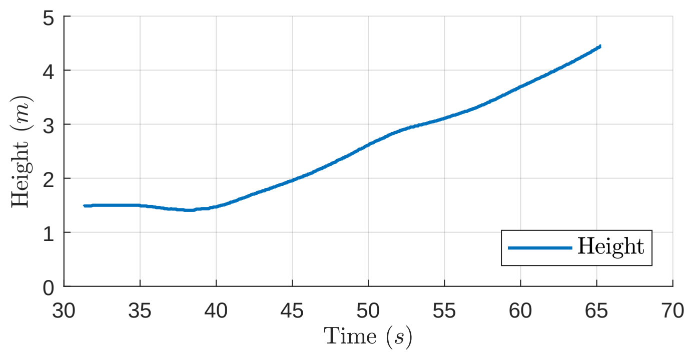
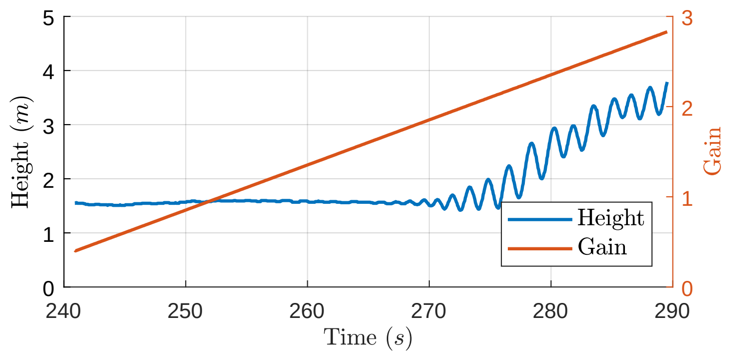


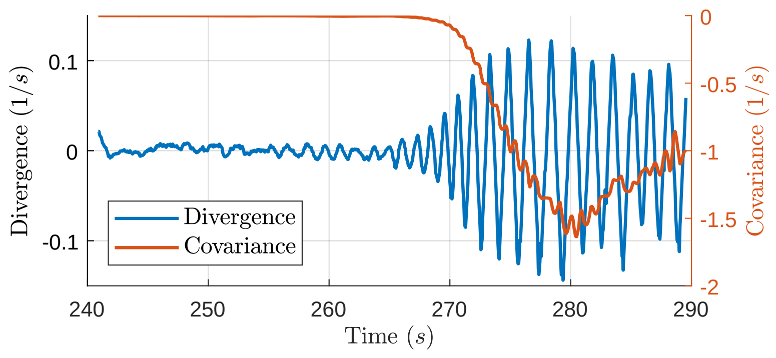
**Figure 6-17:** The quadrotor drifting away upwards with $K_p = K_i = 0$

### Increasing gain without limit

Furthermore, similarly to Section 6-1-2 it might also be useful to observe what happens if the gain keeps increasing indefinitely. Repeating the same experiment but now with a continuously increasing $K_p$ gain. Figure 6-18a shows the gain increasing and the quadrotor starting to oscillate in the vertical axis. Figure 6-18b shows the divergence, also oscillating, and the covariance with the set thrust.



**(a)** The height and control gain $K_p$ of the oscillating quadrotor



**(b)** The divergence and covariance of the oscillating quadrotor

**Figure 6-18:** Applying the algorithm to the vertical axis without oscillation detection

It is again clear that a threshold on the covariance can be used to detect the oscillations;

however to pick this value look at Figure 6-19 for a zoomed in variant of Figure 6-18b. The threshold was chosen to be 0.02.

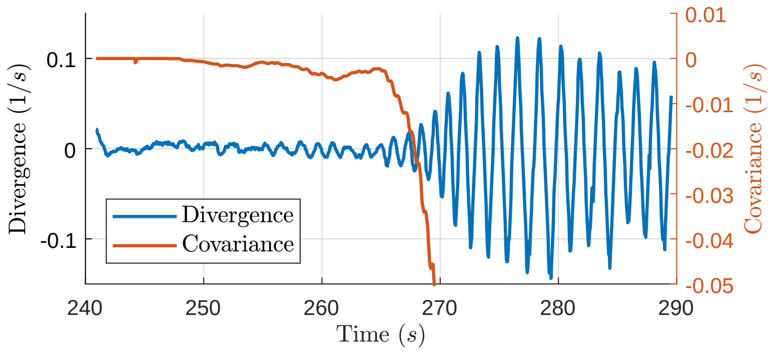


**Figure 6-19:** A close up of the covariance and divergence of the oscillating quadrotor

Furthermore, looking at the oscillation frequency it can be observed that this is roughly 0.58 Hz, a period of 1.7 seconds. With a window size of 100 and an average effective FPS of 30 the window contains about 2 period, which should suffice. This is supported by the smooth line in Figure 6-18b.

## Adaptive gain strategy

Now the full adaptive gain strategy as mentioned in Section 3-3 can be implemented. To do this the quadrotor is set to hover at 1.5 meter and the algorithm is started to increase the gain until oscillations are detected. Once they are detected the gain is reduced by multiplying with a reduction factor of 0.6, which allows the quadrotor to stabilize and continue to hover. This experiment is done multiple times until the battery ran out. In between experiments the quadrotor is reset into a stable position by OptiTrack. The result can be seen in Figure 6-20 where the gains are plotted, as well as the average gain. Note that the dots indicate the moment that the covariance became smaller than the threshold. Most triggers are around the same gain, only the second last experiment triggered a little earlier. This could be due to various factors, such as slightly different initial conditions or noise. These experiments are repeated for initial heights of 1 and 2 meter, which can be seen in Figures 6-21a and 6-21b.

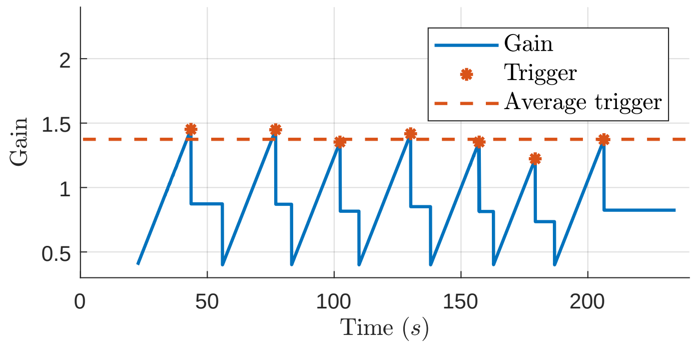
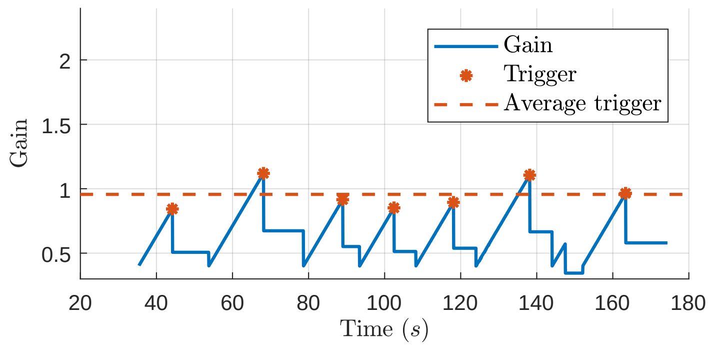
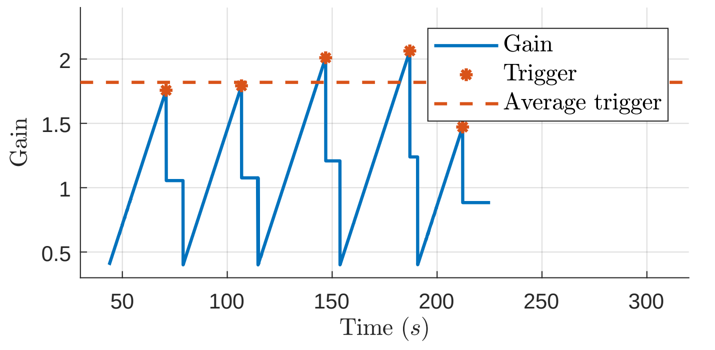


**Figure 6-20:** The gains of multiple adaptive gain experiments at 1.5 meter



**(a)** $1$ meter



**(b)** $2$ meter

**Figure 6-21:** The gains of multiple adaptive gain experiments at $1$ and $2$ meter

Note the experiment at 147.5 seconds in Figure 6-21a was aborted and therefore also not included as a trigger. Furthermore, it must be noted that the variance is slightly higher at 1 meter than at 1.5 meter, while it is even larger at 2 meter. However, the heights at which the algorithm was triggered show the same behavior. This can be seen in Figure 6-22, where the gains are plotted against the height. A line which is fitted through all datapoints is shown in dashed black.

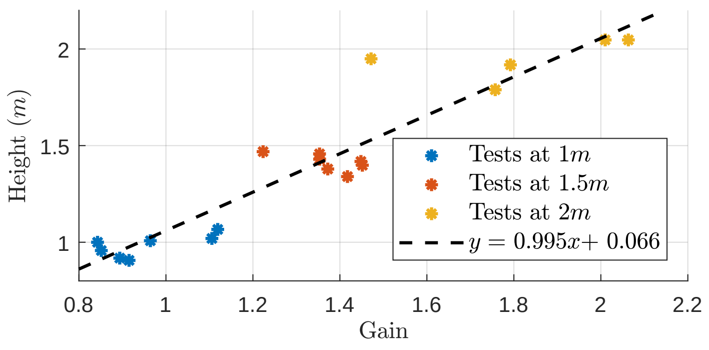


**Figure 6-22:** The gain against the height for each vertical trigger point. The dashed black line is a linear fit through the data

The fit shows almost a $1:1$ relationship for the gain and the height, at $y = 0.995x + 0.066$. This relation can now be used to estimate the height at which a quadrotor is flying when the adaptive gain algorithm is applied. Similarly it can also be used to determine the appropriate gain for each height, though to get the stable gain one should first multiply it with a reduction factor of 0.6.

## 6-3-2 Extension to horizontal axis

With the vertical axis done the next step is to extend it to the horizontal axes. However, to aid development and testing experiments are performed on a single horizontal axis at first. This is achieved by modifying the code so that OptiTrack can control the vertical axis and one of the horizontal axis, leaving the third axis for the module. In the following experiments the X and Z axis will be controlled by OptiTrack and the Y axis will be controlled by the module, therefore the desired angle will be in the roll angle $\phi$.

Repeating the order of experiments from the vertical case in Section 6-3-1, first the case with zero gains will be shown, followed by increasing gains without a trigger. Finally the adaptive gain strategy will be implemented in the horizontal axis.

### Zero gains and increasing without limit

Figure 6-23 shows the quadrotor drifting left 1 meter over a period of 13 seconds, followed by a drift to the right of 8.1 meter in 85 seconds, which clearly shows how important visual feedback is when stabilizing a quadrotor. The reason the quadrotor drifts is that the AHRS doesn't estimate the pitch and roll angles correctly.

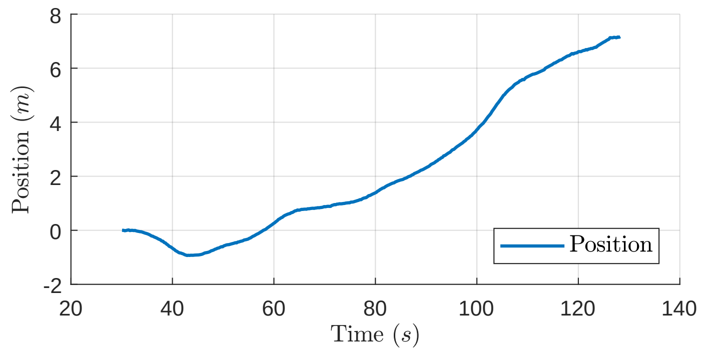
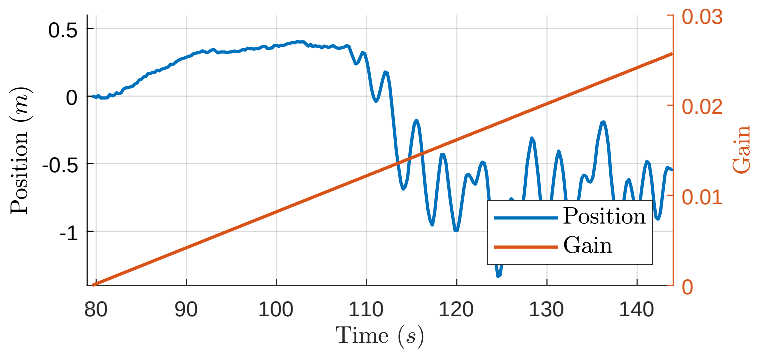
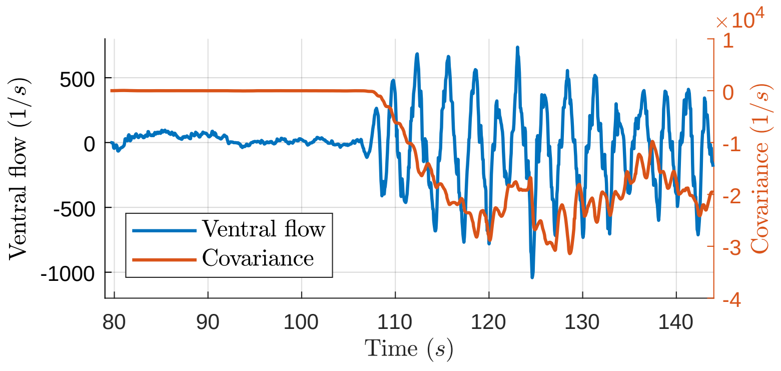


**Figure 6-23:** The quadrotor drifting away sideways with $K_p = K_i = 0$

Figure 6-24a shows what happens if the gain is increased without attempting to measure the oscillation, the quadrotor starts to oscillate. Furthermore, in Figure 6-24b the ventral flow and covariance show that the oscillation is detectable using a threshold on the covariance. The zoomed in version, Figure 6-25, shows that a value of $-2000$ would be suitable as a trigger. After triggering the gain is multiplied with a reduction factor of 0.4.



**(a)** The position and control gain $K_p$ of the oscillating quadrotor



**(b)** The ventral flow and covariance of the oscillating quadrotor

**Figure 6-24:** Applying the algorithm to the horizontal axis without oscillation detection

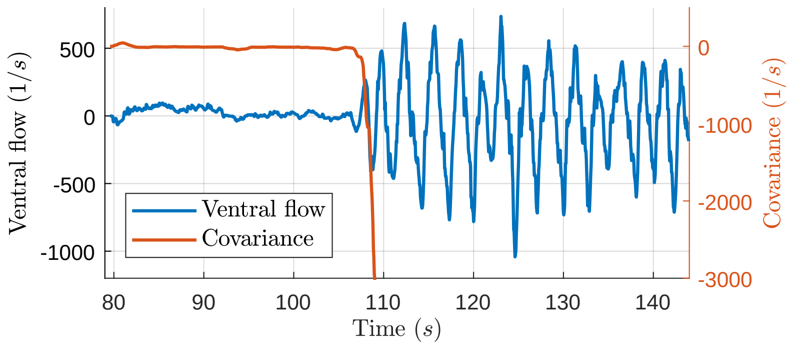
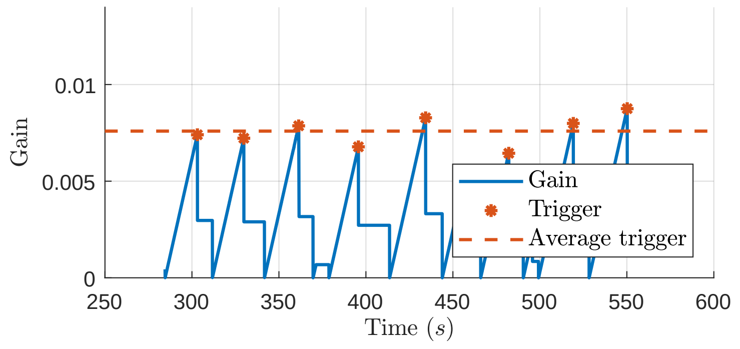


**Figure 6-25:** A close up of the covariance and ventral flow of the oscillating quadrotor

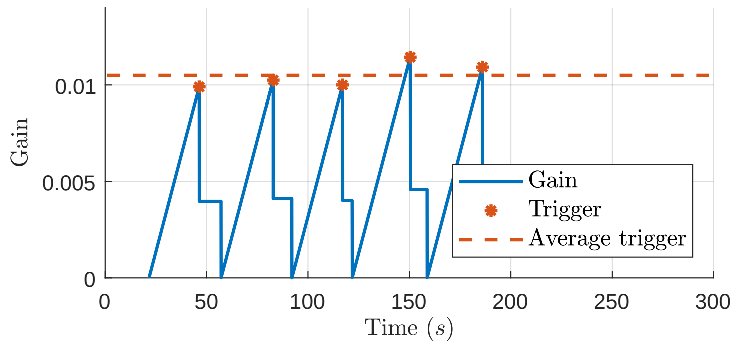
## Adaptive gain strategy

Similarly to Section 6-3-1 the quadrotor is put into a stable hover using OptiTrack before running the adaptive gain experiment in horizontal Y axis multiple times per height at 1 meter, 1.5 meter and 2 meter. A constant I gain of $K_i = 0.0001$ is also introduced. Figure 6-26 shows the resulting triggers at the different heights.

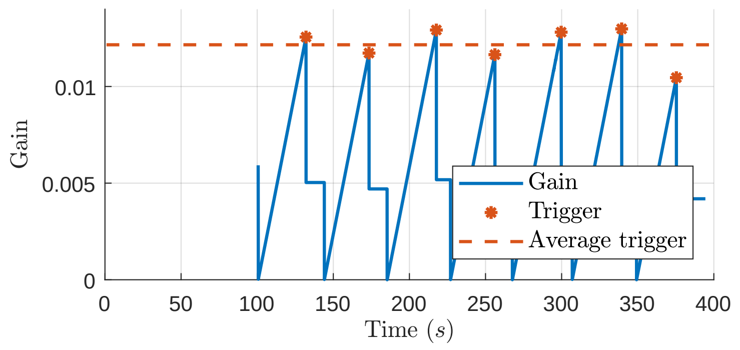
Plotting all trigger points in a gain versus height plot a line can be fitted once again. Figure 6-27 shows that the fit is worse than the vertical fit. Especially the overlap between one trigger point of 2 meter with the gains at 1.5 meter.



**(a)** 1 meter



**(b)** 1.5 meter



**(c)** 2 meter

**Figure 6-26:** The gains of multiple adaptive gain experiments at 1, 1.5 and 2 meter

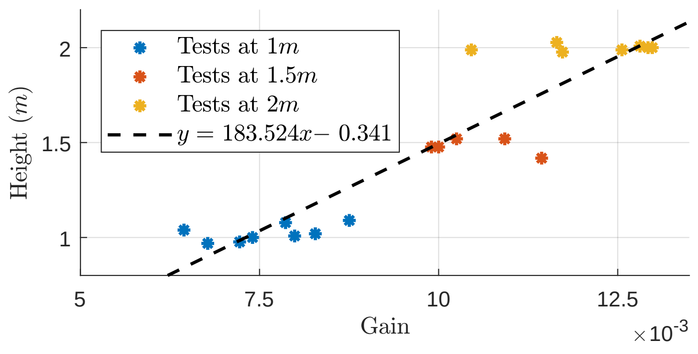


**Figure 6-27:** The gain against the height for each horizontal trigger point. The dashed black line is a linear fit through the data

## 6-3-3 Complete control strategies

In this section different control strategies will be presented to stabilize the quadrotor in all three axis. The following sections will explain these strategies:

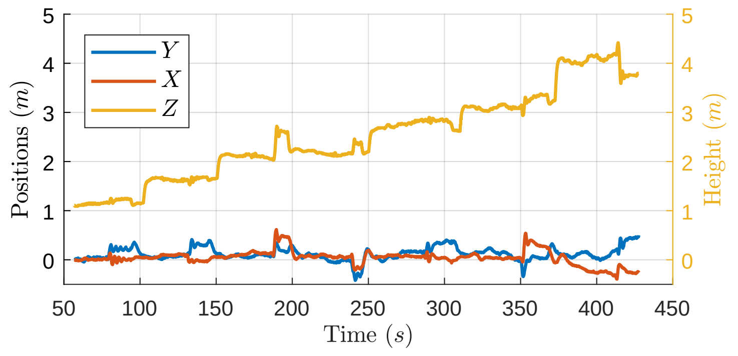
- Three axes at the same time
- Three axes subsequently
- Using the previously found height gain relationships

### Three axes at the same time

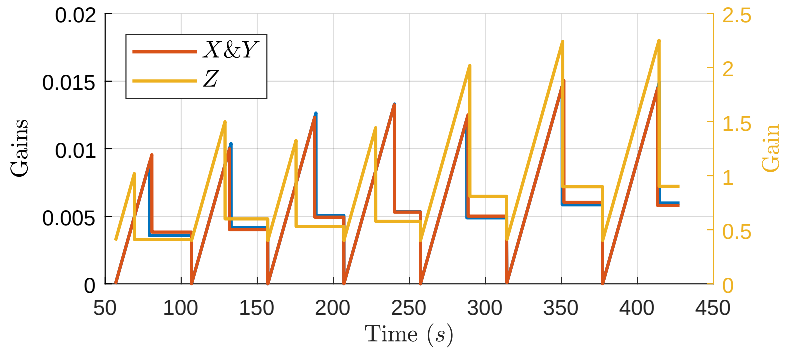
First the algorithm will simply be started in all 3 axes at the same time. This is arguably not the best approach as the quadrotor may become more unstable if all three axes happen to oscillate at the same time than previously seen in one axes. This strategy is tested at 1 to 4 meter at 0.5 meter intervals, where the quadrotor is set to hover using OptiTrack, before starting the algorithm. Figure 6-28a shows the movement of the quadrotor while Figure 6-28b shows the respective gains increasing and triggering. Though the gains don't show a perfect linear relation with the height it is clear the reduction in gain still ensures a stable system. A slight drift or oscillation can be observed when the algorithm triggered too early or late respectively. This is the case for example in the third experiment, at a 2 meter height, where the vertical gain triggers early followed by a drift upwards when the horizontal axis trigger.

Figures 6-28c and 6-28d show the ventral flows, the divergence and the height in the first plot and the covariances in the latter. It can be observed by looking at the covariances that even though the vertical axis has already been triggered by the small peaks in the first four tests, it also oscillates when the horizontal axes trigger.

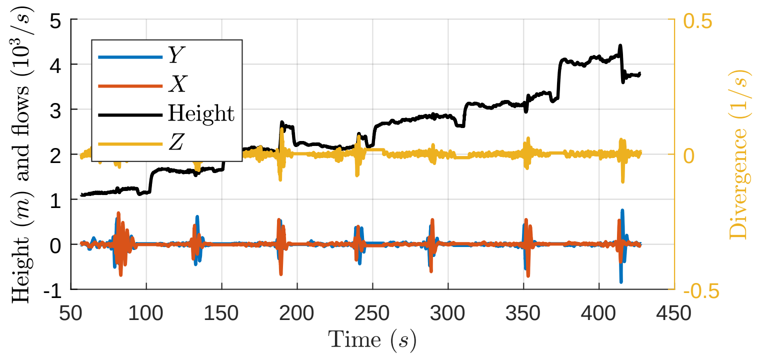
This could suggest either the found gains were not entirely correct, or the oscillation in horizontal axes causes a drop in effective upwards thrust due to the pitch and roll angles. One could compensate for this, but that would also increase the effective thrust in the horizontal axis, possibly causing an unexpected increase in oscillation there. Instead one of the other methods should be explored.



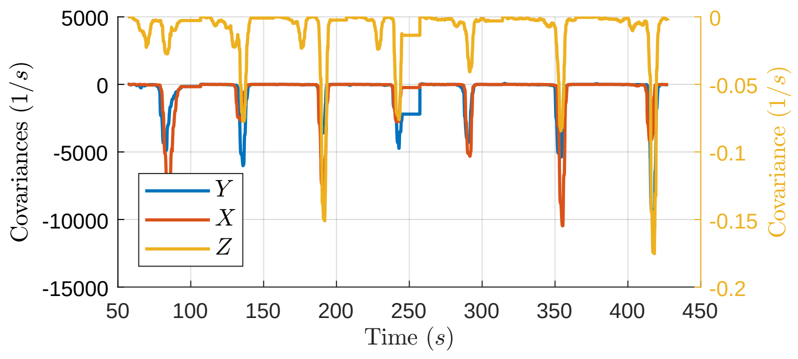
**(a)** The $X$ and $Y$ position and the height of the quadrotor



**(b)** The gains in $X, Y$ and $Z$



**(c)** The ventral flows, scaled by a factor $10^{-3}$ and the divergence



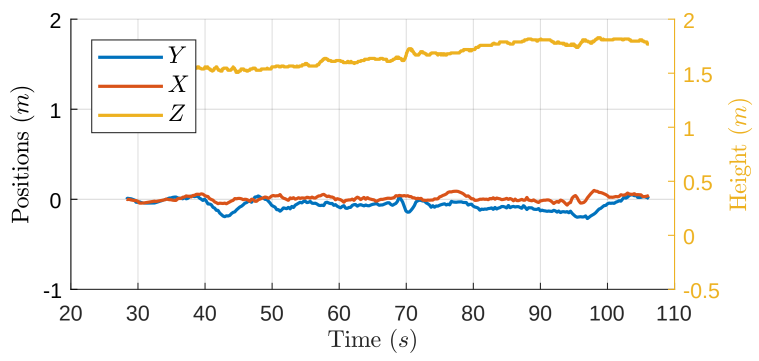
**(d)** The covariances used as a trigger

**Figure 6-28:** The quadrotor with the algorithm applied to all axes at the same time at different heights

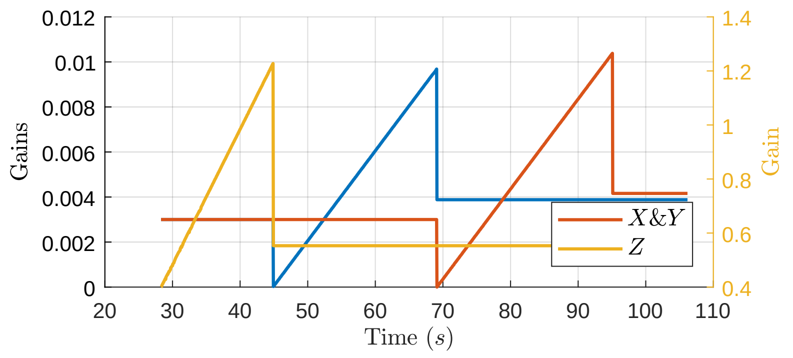
### Three axes after each other

Another solution could be to start the algorithm in one axis for example the vertical axis, followed by the Y and later X axis when the previous ones have triggered. Alternatively both horizontal axis can be set to the same gain found by applying the algorithm to the Y axis, due to the near symmetrical nature of the ARDrone 2 in terms of control.

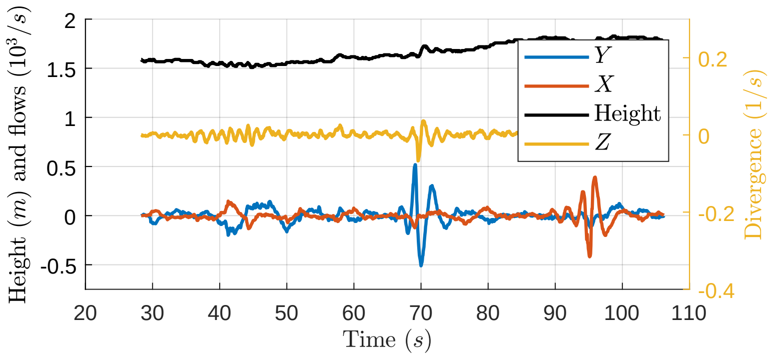
As can be seen the quadrotor oscillates in the three axes subsequently and remains stable during and after the period where the gains are increasing. It is worth noting that an oscillation in a horizontal axis still causes the vertical axis to oscillate a bit, most notable when the Y axis triggers around 70 seconds in the divergence and covariance plots in Figures 6-29c and 6-29d. To prevent this from happening the following section presents a third method, which only requires one axis to oscillate.



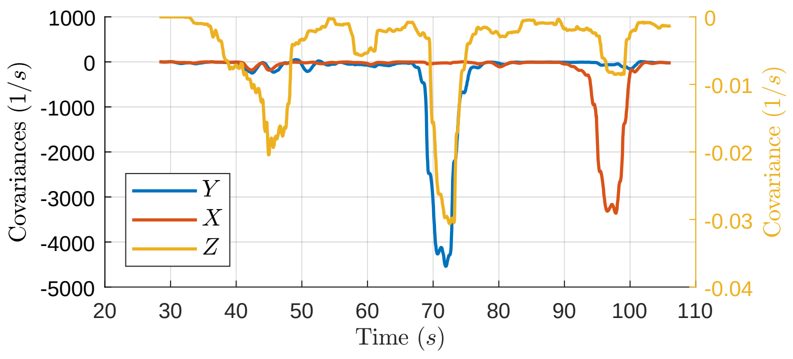
**(a)** The $X$ and $Y$ position and the height of the quadrotor



**(b)** The gains in $X, Y$ and $Z$



**(c)** The ventral flows, scaled by a factor $10^{-3}$ and the divergence



**(d)** The covariances used as a trigger

**Figure 6-29:** The quadrotor with the algorithm applied to all axes subsequently, at $1.5$m

### Using the height gain relationship

Finally the relation between height and gain at which the quadrotor starts to oscillate, as found in Sections 6-3-1 and 6-3-2, can be used by applying the algorithm only in the vertical axis until it triggers. Next this gain can be used to estimate the height at which the quadrotor is flying, which in turn can be used to set the appropriate gains in both the horizontal axes.

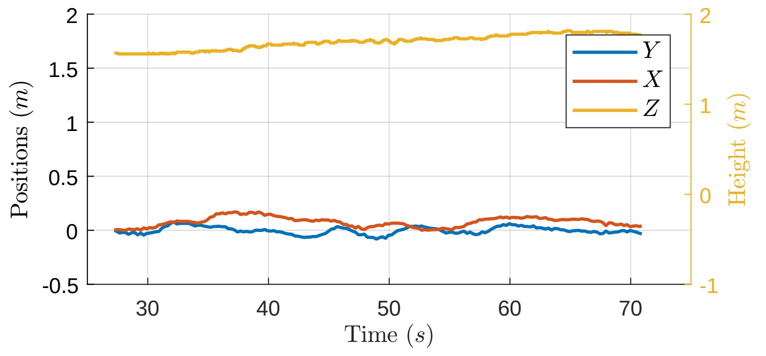
Alternatively the same could be done by starting the algorithm in one of the horizontal axes and using the estimated height to set the other horizontal axis and the vertical axis. However, using the vertical axis is preferred as that oscillation is a naturally more stable axis for a quadrotor.

Figure 6-30 shows the results in similar figures as the previous sections. From the vertical gain in Figure 6-30b the trigger gain can be determined at 1.579, using the relation found in Figure 6-22 this corresponds to an estimated height of 1.64 meter. This value is in turn used to compute the gain at which the quadrotor would become unstable in the horizontal axis using the relation found in Figure 6-27. The result, 0.0108 should however first be multiplied by the reduction factor of 0.4 before the stable gain of 0.0043 is found.

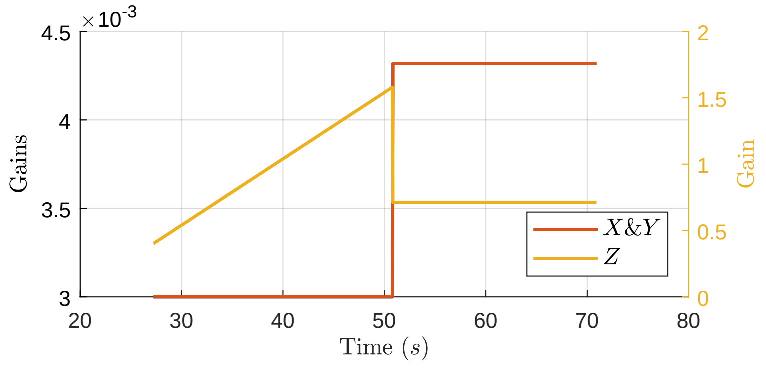
Furthermore, it is interesting to observe that the horizontal starting gain was picked relatively close already at 0.003, together with the standard I gain, $K_i = 0.0001$, resulting in little drift during the adaptive gain phase of the vertical axis. To truly see the benefit of this control strategy both the horizontal starting gains, as well as the I gains, have been set to zero before starting the experiment in Figure 6-31.

The result of the zero initial gains is that the quadrotor keeps drifting until at 45 seconds the vertical axis is triggered and the proper gains are set for all axes. Figure 6-31a shows this clearly, as the quadrotor remains in place once the algorithm is triggered instead of continuing the drift.

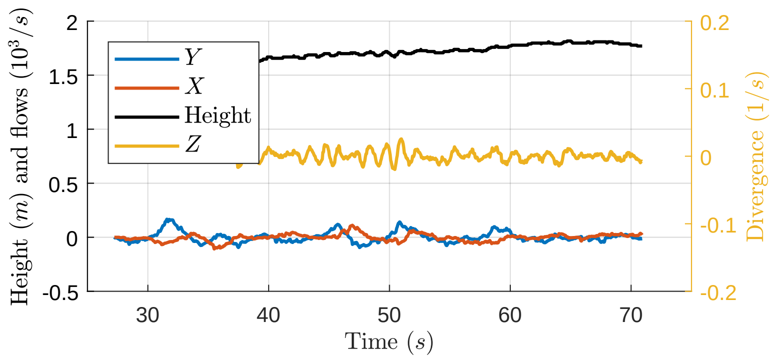
With this knowledge it can be concluded that the third method of using the previously found relationships and low but non zero starting gains for the horizontal axes is the best approach to stabilize a quadrotor with only a monocular camera and an IMU.



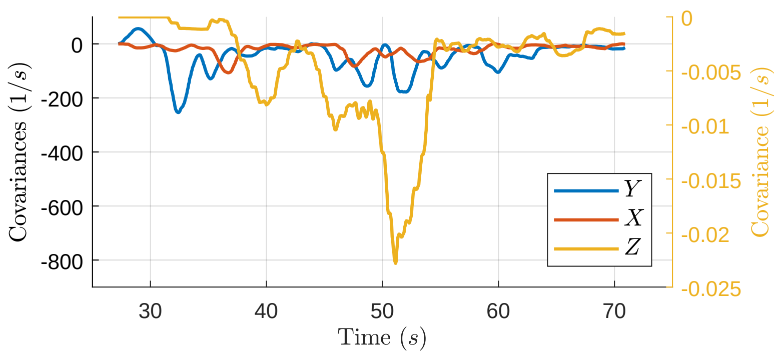
**(a)** The $X$ and $Y$ position and the height of the quadrotor



**(b)** The gains in $X, Y$ and $Z$

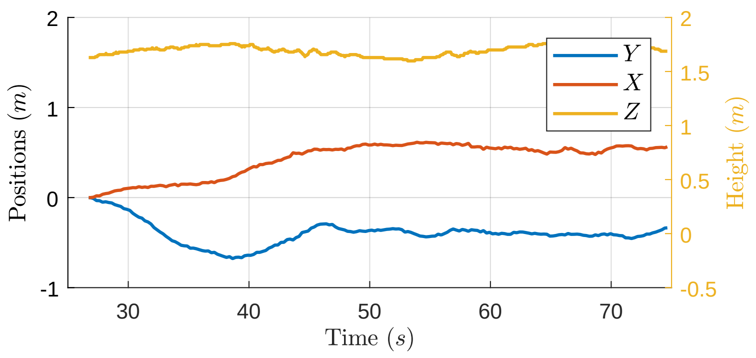


**(c)** The ventral flows, scaled by a factor $10^{-3}$ and the divergence

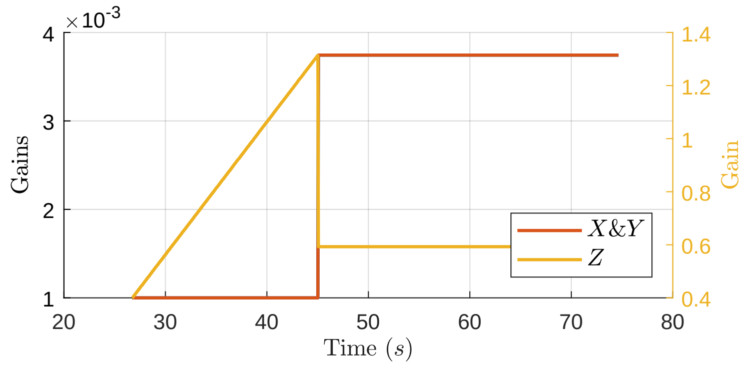


**(d)** The covariances used as a trigger

**Figure 6-30:** The quadrotor with the algorithm applied to the vertical axis, at $1.5$m, in order to set all gains using previously found relationship between height and gain



**(a)** The $X$ and $Y$ position and the height of the quadrotor



**(b)** The gains in $X, Y$ and $Z$

**Figure 6-31:** The quadrotor with the algorithm applied to the vertical axis, at $1.5$m, in order to set all gains using previously found relationship between height and gain with zero starting gains in the horizontal axes

### 6-3-4 Implementation on a Bebop quadrotor

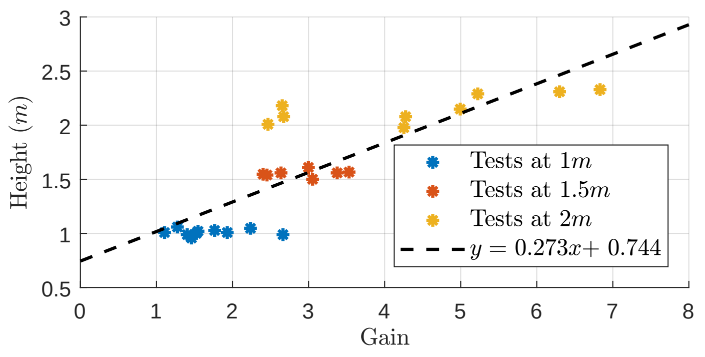
With the results achieved on the ARDrone 2 in the previous section, this section presents the results from flying with a Bebop quadrotor. As described in Section 3-2 the Bebop is smaller but does not weigh a lot less. Furthermore, it is worth noting that the Bebop actually contains a better camera in terms of resolution, and also manages to compute the flow using LK much faster than an ARDrone 2.

In the following experiments the following assumptions hold, unless mentioned otherwise

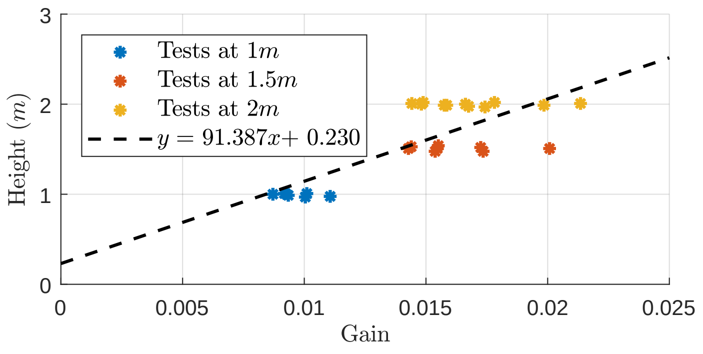
- A Bebop 1 will be used with a regular battery and indoor bumpers
- The experiment will be started from a stable hover using the OptiTrack system
- The LK vision algorithm will be used to compute the flows and divergence
- The effective FPS will be around 90
- The window for oscillation detection will 300 samples given this effective FPS
- The flow and divergence inputs will be put through a low-pass filter before being used

#### Gain height relationships

The experiments on the ARDrone 2 have been repeated for the Bebop, though for brevity only the important results are shown here. Figures 6-32a and 6-32b show the relation between gain and height for the vertical and horizontal axis respectively. The Z axis shows a clear linear relation, with some outliers. The experiments for horizontal axis at 2 meter high however seem to match those of 1.5 meter high. This the fit less accurate, as it would seem that the quadrotor cannot differentiate the height between 1.5 and 2 meter due to overlapping gains.



**(a)** The gain height relation for the vertical Z axis



**(b)** The gain height relation for the horizontal Y axis

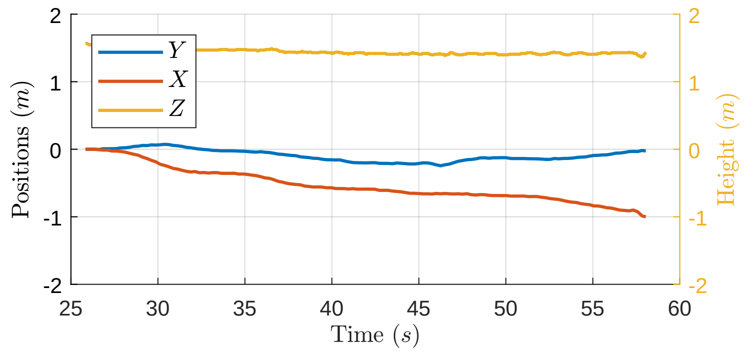
**Figure 6-32:** The gain against the height for each vertical and horizontal trigger point respectively. The dashed black line represents a linear fit through the data

## Control strategy

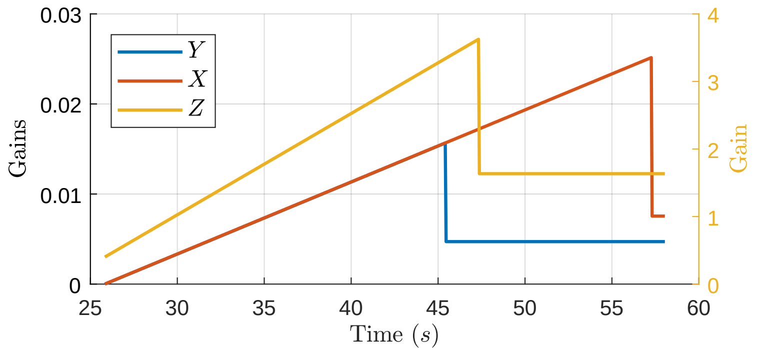
As the horizontal gain height relationship is not very accurate, the Bebop is stabilized by starting the adaptive gain algorithm in all three axes at the same time. Interestingly Figures 6-33b and 6-33d show an interesting result, the horizontal gains do not trigger at the same time. This would suggest the Bebop is not entirely symmetrical in for this controller. One explanation could be the way the Bebop body with the IMU and cameras is attached to the frame with the rotors, with four vibration dampers in the same rectangular shape as the body. They are close together in the Y axis, allowing the body to wiggle easily, and further apart in the X axis, allowing much less of a wiggle. This could cause slightly different vibrations in the horizontal axes with respect to each other, leading different camera rotation frequencies in pitch and roll. If we zoom in on Figure 6-33d, see Figure 6-34, we see that Y axis shows a higher frequency corresponding with the the roll angle $\phi$ which was easier to wiggle.

The fact that the Bebop is not symmetrical for the adaptive controller is not an issue by itself. The gain at which the X axis eventually triggers causes a very large reaction. In fact when testing this out multiple times on just the X axis, the stable gain after the 0.4 correction factor proved to be too large, maintaining oscillations. Judging from this the reduction factor can be lowered, or alternatively, the oscillation in Y axis can be used to both horizontal axis at the stable gain obtained by the Y axis. As shown in Figure 6-35, this stabilizes the quadrotor much better.

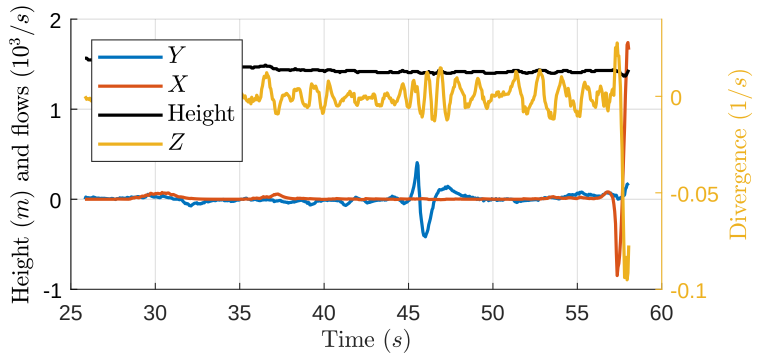
With this it is shown that a Bebop can also be stabilized using the adaptive gain strategy. However, as discussed in Section 6-4, the gain height relation in the horizontal should be improved.



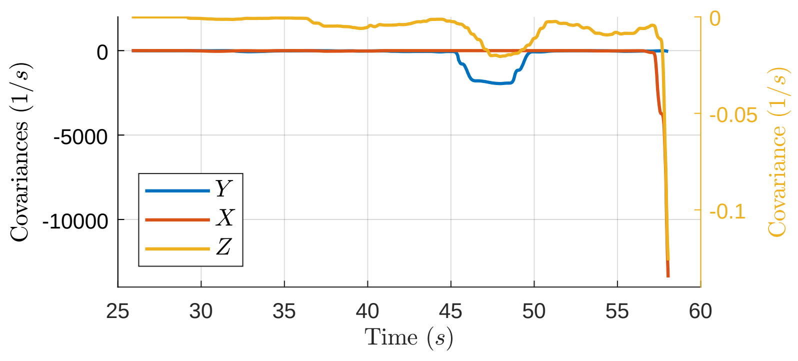
**(a)** The $X$ and $Y$ position and the height of the quadrotor



**(b)** The gains in $X, Y$ and $Z$



**(c)** The ventral flows, scaled by a factor $10^{-3}$ and the divergence



**(d)** The covariances used as a trigger

**Figure 6-33:** The quadrotor with the algorithm applied to the all axes at the same time at a height of 1.5m

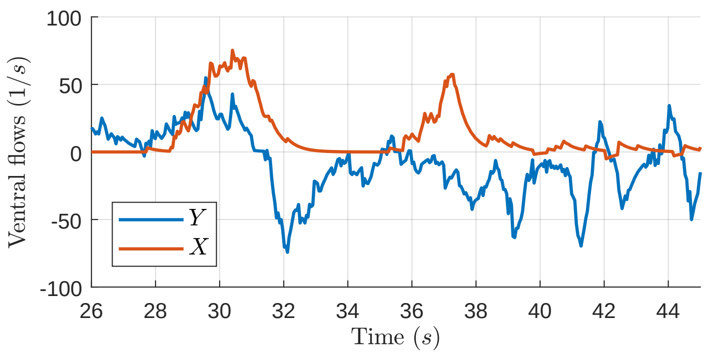
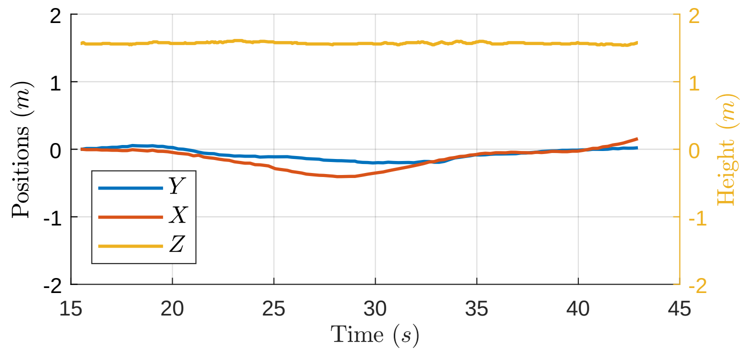
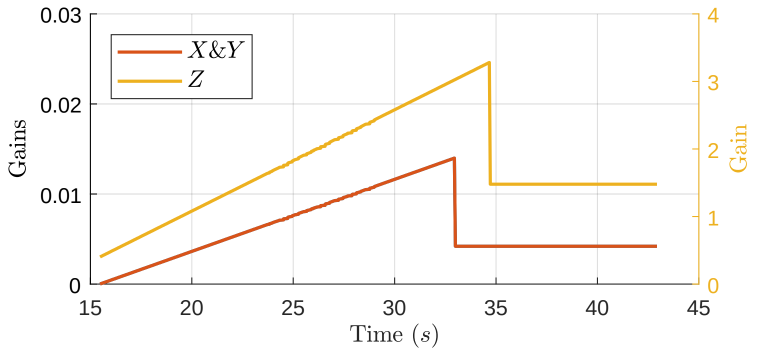


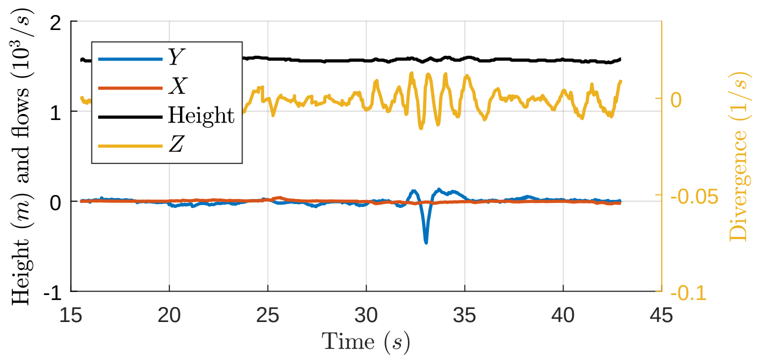
**Figure 6-34:** Zooming in on the ventral flows



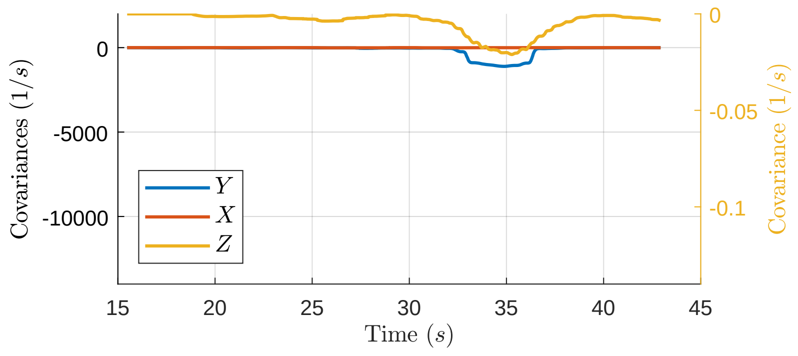
**(a)** The $X$ and $Y$ position and the height of the quadrotor



**(b)** The gains in $X, Y$ and $Z$



**(c)** The ventral flows, scaled by a factor $10^{-3}$ and the divergence



**(d)** The covariances used as a trigger

**Figure 6-35:** The quadrotor with the algorithm applied to the all axes at the same time at a height of 1.5m A trigger in the Y axis will also set the gain for the X axis

## 6-4 Experimental Discussion

The experimental results presented in the previous section

- The most important question is why the gain height relationship is not as clear as expected. On the ARDrone 2 the fit was good enough to estimate the height and use this estimation as a way to set the other gains. On the Bebop however, there was a large overlap, especially in the horizontal axis. It could simply be due to tuning, as I spend more time on the ARDrone 2 than on the Bebop. Furthermore, as previously mentioned, the Bebop quadrotor has dampers between the body containing the camera and IMU and the frame containing the motors. This was already seen as a possible reason for the fact that the two horizontal axis are not reacting symmetrical, due to different resistance to vibrations in two axis. It could be that the Bebop reacts differently to the starting oscillations at other heights because the body and thus the camera can wiggle. After all, to reach the same covariance threshold at a point further from the ground the desired angle and/or the velocity must be larger to compensate for the larger height. To verify this the dampers of the Bebop could be fixed, as is done with some Bebops in other research areas.

- The calibration of the gain height relation was done from stable hover using the OptiTrack system. Though the start of the experiments could be done from a manual flight, keeping two out of three axis stable while the algorithm is applied in the third will be tough. This limits the method to systems that can be calibrated in a controlled environment unless calibration can be done reliably in three axis at the same time. This however still leaves the problem of knowing at what height the quadrotor is performing the calibration flight. Using an additional sonar would defeat the purpose of this method given the scalability requirements.

- The quadrotors were only tested in situations where they were already in a stable flight. Unlike in the simulator no feed forward takeoff was performed. However, when testing I did start the algorithm while the quadrotor was still moving towards its setpoint. This resulted in some additional drift but the algorithm still stabilized the quadrotor, given the starting velocity was not too large.

- Implementation on the third quadrotor, the Blaze, was not achieved due to the fact that Parrot SA has not released the promised firmware yet where a Bluetooth connection can be maintained, required for telemetry. When this is released and Paparazzi runs on the Blaze, implementation should be as straight forward as tuning a few parameters including the slope of the increasing gains, the trigger thresholds and the reduction factors.

- Though one of the identified key papers is on the EF algorithm as a replacement for the more common LK algorithm it wasn't used in this thesis. On the used ARDrone 2 and Bebop drone there is enough computational power for LK, the same probably applies to the Blaze. On smaller quadrotors however, it is going to be a must. Currently EF cannot be used as a drop-in replacement for LK and was skipped due to time limitations implementation.

- The literature summary also mentioned that the snapshot approach from [8] could be useful to implement against the drift. The adaptive gain algorithm however shows no signs of long term drift with the I term added in the controller. The other contribution, the robustness against changes in illumination, was not implemented either given the controlled lightning in the Cyber Zoo. This would still be a beneficial addition to the algorithm. The same applies to other improvements in the OF algorithms.

Chapter 7

# Conclusion

Vision-based stabilization of a quadrotor in all three axes using only a single camera and an IMU is an interesting concept as it would allow for the miniaturizing of quadrotors that are limited in their sensor carrying capabilities. As described in Section 1-3 the goal of this thesis was to develop a scalable control algorithm that can autonomously stabilize a micro quadrotor in an unknown environment with all sensing and computations performed on-board.

First, a literature study was done to determine the possible options to achieve this goal. From these options the instability based adaptive gain method that was used to land a quadrotor using a single camera and IMU was selected as the best approach. A simulation was build in MATLAB to show that this method would also be able to determine the ultimate gain, similar to the Ziegler-Nichols PID tuning method, in the horizontal axis. A control strategy to stabilize the quadrotor after a feed forward takeoff was shown to perform well on the simulated quadrotor with ARDrone 2 specifications. Furthermore, the effects of downscaling this quadrotor were analyzed and it was shown that an appropriate scaling of the control gains could negate most of the effects. The simulated actuator noise however prevented the smaller scale quadrotors to properly stabilize without drifting away, even with an increased FPS from the vision module. Only by also decreasing the noise in the vision module these small scale quadrotors were stabilized. It was shown that the minimum effective FPS required to stabilize an ARDrone 2 using this method is 15 FPS.

Next experiments on two different quadrotors were performed, showing the linear relationship between gain and height at which the quadrotor becomes unstable can be determined. This relationship was later used to stabilize an ARDrone 2 after estimating its height by oscillating in the vertical axis and using this height estimation to set the appropriate gains for the horizontal axes. Furthermore it was also shown that applying this algorithm in all axes at the same time would also stabilize the quadrotor.

To answer the research questions, it is indeed possible to stabilize a quadrotor with only a single camera and an IMU while computing everything on-board. The stabilization was accurate enough that the ARDrone 2 didn't drift away from its original position, whereas without vision-based control it would drift at roughly 8 cm per second. The Bebop could also hover without drifting away. During flights in the Cyber Zoo I was even able to move the playing mat that serves as a texture rich ground surface and both quadrotors would follow the playing mat wherever I dragged it.

Further research and work is required to implement this control algorithm on even smaller quadrotors.

# Chapter 8

# Recommendations

This chapter will provide some suggestions for future work, separated in two sections, one for the simulation and one for the experiments on the quadrotors.

## 8-1 Simulation

Although the simulation has proven to be a useful proof of concept for extension of the adaptive gain algorithm to the horizontal axis, it can still be improved.

- Currently the simulator is heavily simplified, limiting the use of its predictions in reality. An extended simulator would greatly improve the use of findings. Furthermore it could potentially serve as a environment to quickly develop and test code for the quadrotors in a safe way. Simulators such as Microsoft's AirSim provide a high fedility visual and physical simulation. Alternatively Gazeboo now works with Paparazzi, thanks to Tom van Dijk, this adds all the Paparazzi code into a simulator so the same inner loop can be used for example.

- Simulating vision will always be different than the real measurements, perhaps it would be interesting to create a dataset using different cameras and maneuvers to make the simulated vision more accurate. Alternatively the above-mentioned simulators could be beneficial in this aspect too.

- Finally, there appears to be an important trade-off between the maximum FPS a vision module can supply, and the noise level of the measurements. A camera might be able to produce results faster, but more coarse. It would be valuable to investigate the optimum given a more extensive simulator, to determine the research direction for improved vision algorithms.

## 8-2 Experimental

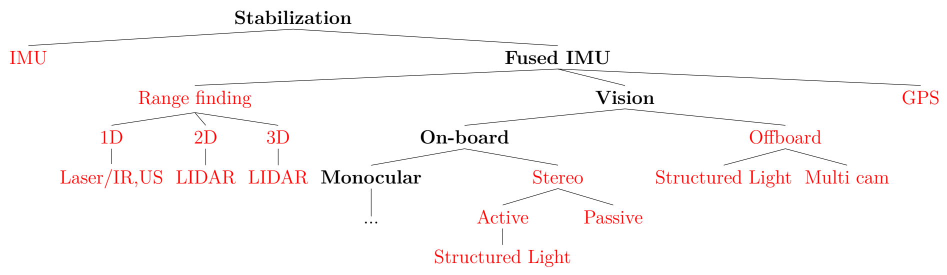
Far more work can be done on the quadrotors, from physically building them smaller to implementing the code on computational less powerful microcontrollers. The following points can be taken as a starting point for future work.

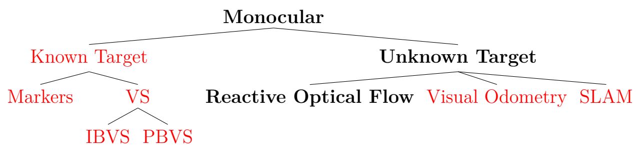
- Experiments using a Bebop with fixed dampers should be performed to determine if this indeed influences the adaptive gain strategy differently at various heights.
- Once the firmware supporting a Bluetooth connection is available for the Blaze, Paparazzi should run on this tiny quadrotor. Next implementing the presented control schemes should be a matter of calibrating some parameters and determining the gain height relationship for the Blaze.
- To reduce the computational load of the vision module EF should be used to compute the ventral flows and divergence. This either requires EF to be changed so it is a drop-in replacement of the LK algorithm, or the adaptive gain strategy to cope with EF instead. Improvements of the vision module in terms of accuracy or computation time in general are beneficial.
- The adaptive gain method is especially interesting given the gain height relationship that allows the quadrotor to estimate the height, which can in turn be used for other tasks. However, if this relationship is not as clear as was the case in the horizontal axis of the Bebop, the algorithm can still be used without the height estimation. It is important to make sure the reduction factor is on the safe side in that case, as a slightly drifting quadrotor is more desirable than an oscillating one. Furthermore, the algorithm could be altered such that after the gain is reduced, it continues to increase again with a more gradual slope. This could increase robustness as triggers that occur too early are eventually corrected and if the initial reduction factor is large enough a late trigger will also be corrected.
- In order to run the code on the small microcontrollers, instead of the Linux ARM Cortex A8 that is in the ARDrone 2 for example, the code has to be converted for fixed-point arithmetic. This will require quite some work and optimizing.

Appendix A

# Appendix A - Quadrotor stabilization

In this appendix the complete tree of quadrotor stabilization options is printed. Note that the chosen path is printed in **bold**, while the options that are not available are printed in red.





Appendix B

# Appendix B - Code

## B-1 MATLAB simulation

The MATLAB code, as well as the Simulink simulation can be found in the following repository: https://github.com/TitusBraber/Quadrotor2D

## B-2 Paparazzi code

The Paparazzi code for the ARDrone 2 and Bebop can be found in my fork of Paparazzi for now: https://github.com/TitusBraber/paparazzi/tree/xyz_OF
After my thesis I will clean up the code and create a pull request in Paparazzi master: https://github.com/paparazzi/paparazzi/

# Bibliography

[1] G. de Croon, "Distance estimation with efference copies and optical flow maneuvers : a stability-based strategy .," *Bioinspiration & Biomimetics*, vol. 11, no. 1, pp. 1–30, 2016.

[2] D. Brescianini, M. Hehn, and R. D'Andrea, "Quadrocopter pole acrobatics," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3472–3479, 2013.

[3] M. Müller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," *IEEE International Conference on Intelligent Robots and Systems*, pp. 5113–5120, 2011.

[4] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *Springer Tracts in Advanced Robotics*, vol. 79, pp. 361–373, 2014.

[5] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[6] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction of Cubic Structures with Quadrotor Teams," *Mechanical Engineering*, 2011.

[7] K. N. McGuire, G. de Croon, C. De Wagter, K. Remes B.D.W. Tuyls, and H. J. Kappen, "Local Histogram Matching for Efficient Optical Flow Computation Applied to Velocity Estimation on Pocket Drones," *In press*, pp. 1–15, 2015.

[8] P. Li, M. Garratt, and A. Lambert, "Monocular Snapshot-based Sensing and Control of Hover, Takeoff, and Landing for a Low-cost Quadrotor," *Journal of Field Robotics*, vol. 32, no. 7, pp. 984–1003, 2015.

[9] C. M. S. McMichael, James M.Francis, "Darpa's: Micro air vehicles - toward a new dimension in flight," 1997.

[10] X. Zhang, B. Xian, B. Zhao, and Y. Zhang, "Autonomous Flight Control of a Nano Quadrotor Helicopter in a GPS-Denied Environment Using On-Board Vision," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 10, pp. 6392–6403, 2015.

[11] DARPA, "Darpa nano air vehicle (nav) program," 2011.

[12] T. Ricci, "Darpa's: Micro air vehicles - toward a new dimension in flight," 2011.

[13] H. Chao, Y. Gu, and M. Napolitano, "A survey of optical flow techniques for robotics navigation applications," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 73, no. 1-4, pp. 361–372, 2014.

[14] F. van Breugel, K. Morgansen, and M. H. Dickinson, "Monocular distance estimation from optic flow during active landing maneuvers," *Bioinspiration & Biomimetics*, vol. 9, no. 2, p. 025002, 2014.

[15] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *InTech*, vol. 42, no. 6, pp. 94–100, 1995.

[16] C. H. Wolowicz and J. S. Bowman, "NASA Technical Paper 1435 Similitude Requirements and Scaling Relationships as Applied to Model Testing NASA Technical Paper 14-35 Similitude Requirements and Scaling Relationships as Applied to Model Testing," no. August, p. 65, 1979.

[17] V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles," *The International Journal of Robotics Research*, vol. 31, pp. 1279–1291, sep 2012.

# Glossary

## List of Acronyms

| | |
|---|---|
| **3mE** | Mechanical, Maritime and Materials Engineering |
| **AHRS** | Attitude and Heading Reference System |
| **CL** | Closed Loop |
| **CoM** | Center of Mass |
| **CPU** | Central Processing Unit |
| **DARPA** | Defense Advanced Research Projects Agency |
| **DCSC** | Delft Center for Systems and Control |
| **DOF** | Degrees Of Freedom |
| **EF** | EdgeFlow |
| **EoM** | Equations of Motion |
| **ESC** | Electronic Speed Controller |
| **FPS** | Frames per Second |
| **GCS** | Ground Control Station |
| **GPS** | Global Positioning System |
| **IMU** | Inertial Measurement Unit |
| **INS** | Integrated Navigation System |
| **LiPo** | Lithium Polymer |
| **LK** | Lucas-Kanade |
| **MAV** | Micro Aerial Vehicle |

| | |
|---|---|
| **MAVlab** | Micro Air Vehicle Laboratory |
| **NAV** | Nano Aerial Vehicle |
| **OF** | Optical Flow |
| **OL** | Open Loop |
| **PWM** | Pulse Width Modulation |
| **RC** | Remote Controlled |
| **SLAM** | Simultaneous Localization And Mapping |
| **UAV** | Unmanned Aerial Vehicle |
| **US** | Ultrasonic Sensor |
| **VO** | Visual Odometry |
| **ZMWN** | Zero Mean White Noise |
| **ZOH** | Zero Order Hold |