Faculty of Aerospace Engineering

Delfly Freeflight

Autonomous flight of the Delfly in the wind tunnel using low-cost sensors

J.A. Koopmans

June 15, 2012

Delft University of Technology

Challenge the future

Delfly Freeflight

Autonomous flight of the Delfly in the wind tunnel using low-cost sensors

Master of Science Thesis

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

J.A. Koopmans

June 15, 2012

Faculty of Aerospace Engineering · Delft University of Technology

Delft University Of Technology
Department Of
Control and Simulation

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **“Delfly Freeflight”** by **J.A. Koopmans** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: June 15, 2012

Readers:

Prof.dr.ir. J.A. Mulder

Dr.ir. P. Chu

Dr.ir. E. van Kampen

Dr.ir. B.W. van Oudheusden

Ir. B. Remes

Summary

The Delfly is subject of great interest from the aerodynamics department at the TU Delft. Current wind tunnel measurements are performed with a dual high speed camera setup that detect particles injected in the wind stream. The difference between two subsequent images provides information on the flow field around the wings of the Delfly. These measurements are always performed with the Delfly fixed on a support. Although this method produces a lot of useful data, the restrictions that the support introduces makes it not a true representation of the free flight conditions. This thesis goal therefore, was to design, build and test a system that would enable the Delfly to fly freely in the wind tunnel. This would allow the same measurements to be performed without a support, providing insight in the influence of the support on the aerodynamic properties of the Delfly.

A low-cost, high performance tracking system using two Wiimotes was developed, providing 3D position information with an accuracy of 0.8 mm and a tracking rate up to 80 Hz. A custom auto pilot module was designed, containing a 3-axis gyro and an infrared camera. A small Bluetooth module provided two way communication between the Delfly and the ground station, allowing the position information to be sent up to the Delfly and can log the information from the on-board sensors.

Using the tracking system and a LED in the middle of the wind tunnel to provide the camera with a heading reference, a PI controller was implemented on-board. The controller could successfully keep the Delfly within ± 1.7 cm in forward and vertical direction, and within ± 3.5 cm in lateral direction of the reference point. It is the first time in the world that a flapping wing micro aerial vehicle was flown autonomously in the wind tunnel.

The achieved precision is sufficient for the aerodynamic measurements to be performed, which could shed more light on the way the wind tunnel support influences the properties of the Delfly. Further more, for the first time, good quality data has been gathered on the dynamic behavior of the Delfly. This can serve as a starting point for future projects, such as the design of more advanced controllers that cope with the observed non-linearities or provide a reference for future research on the dynamics of the Delfly.

Acronyms

ADC	Analog-to-Digital Converter
c.g.	center of gravity
C&S	Control & Simulation
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
DoF	Degrees of Freedom
DUT	Delft University of Technology
FWMAV	Flapping Wing Micro Aerial Vehicle
GUI	Graphical User Interface
I2C	Inter-Integrated Circuit
IR	Infra-Red
kbs	kilo-bit per second
LE	leading edge
LED	Light Emitting Diode
MAV	Micro Aerial Vehicle
MAVLAB	Micro Aerial Vehicle Laboratory
MEMS	Microelectromechanical systems
MIPS	million instructions per second
MSB	most-significant-byte
OJF	Open Jet Facility
PCB	Printed Circuit Board
PIV	Particle Imaging Velocimetry
SPP	Serial-Port-Profile
SRS	SIMONA Research Simulator
TE	trailing edge
UART	Universal Asynchronous Receiver/Transmitter
UAV	Unmanned Aerial Vehicle

List of Symbols

Greek Symbols

$\delta_{r,e,th}$	rudder, elevator, thrust command
Δ_T	time step size
ϕ	roll angle
ψ	yaw angle
θ	pitch angle

Reference frames subscripts

C_1	tracking camera 1
C_2	tracking camera 2
C	on-board camera
D	Delfly body
$\hat{w}$	uncalibrated wind tunnel
w	calibrated wind tunnel

Roman Symbols

$\mathcal{F}$	reference frame
f	frequency
$\mathbf{I}_3$	3×3 identity matrix
$K^I_{x,y,z}$	integrator gain for resp. x, y, z
$K_{k,m,n}$	rudder, elevator, thrust manual trim value
K_μ	heading error gain
$K_{p,q,r}$	roll, pitch, yaw rate gain

M	point in homogeneous world coordinates
m	projection point in homogeneous image coordinates
N	intrinsic camera matrix
$\mathcal{O}$	origin of reference frame
P	camera matrix
p	roll rate
q	pitch rate
R	3×3 rotation matrix
r	yaw rate
c	center pixel
s	pixel pitch
$\mathbf{T}$	3×1 translation vector
$\mathbf{u}$	pixel coordinate pair in homogeneous coordinates
u	horizontal pixel coordinate
v	vertical pixel coordinate

Contents

"Indubitably, life's lustrous unknowns mandate inquisitiveness, naivety and tantalizing inspiration."

Samuel Mark, 1322

Chapter 1

Introduction

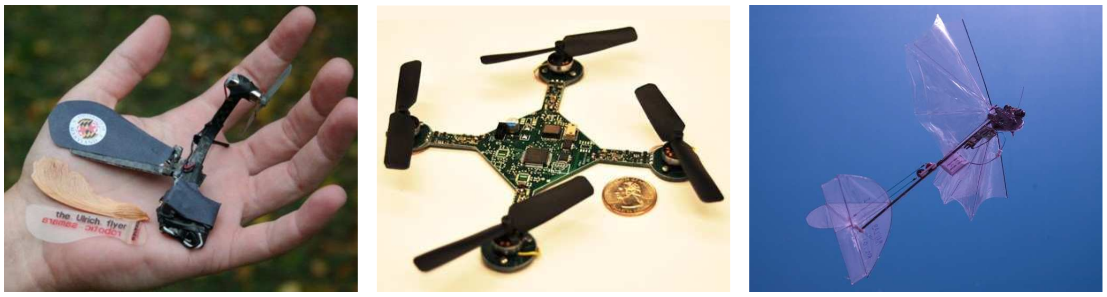
Micro Aerial Vehicle (MAV) are a type of Unmanned Aerial Vehicle (UAV) and distinguish themselves by their low weight and small dimensions (see Table 1-1). Their size makes them well suited for inconspicuous reconnaissance and spying and they are cheap to produce; they have therefore drawn the attention of both military and commercial research. A possible application would be the deployment of a network of (redundant) agents that can monitor both indoor and outdoors in otherwise unreachable locations, utilizing swarm tactics to cover large areas. Technology like this would have great potential in search and rescue missions, as well as various military applications. Defense Advanced Research Projects Agency (DARPA) for one has shown great interest in the development of such systems.

The designs of MAV generally are either of the rotary type or flapping wing, although more exotic designs have been made, like the Samara Nano Air Vehicle (figure 1-1a) which is based on a maple seed and could therefore be considered somewhere in between rotary wing and flapping wing. Rotary wing MAV's utilize propellers for lift, from single-propeller to miniature quad-copter designs like the Nano Quadcopter by the University of Pensylvania (figure 1-1b).

Flapping Wing Micro Aerial Vehicle (FWMAV) on the other hand, use flapping wings for propulsion and lift. They usually strongly resemble small birds or large insects like the hummingbird or dragonfly, and indeed designs are based more often than not on their living counterparts. Because nature has done such a good job on extremely small, agile and efficient

Specification	Requirement
Size	< 15 cm
Weight	100 g
Payload	20 g
Range	1-10 km
Endurance	60 min
Altitude	< 150 m
Speed	15 m/s

Table 1-1: MAV criteria according to Pines & Bohorquez (2006)



(a) Samara Nano **(b)** Nano Quadcopter **(c)** Delfly II

Figure 1-1: Several MAV's

flyers, many engineers seek to mimic their methods of flight. But because of their small size and complex unstable aerodynamics, development of FWMAV is faced with great challenges. Actuators, sensors and batteries need to be as light as possible. The aerodynamics of the flapping wings are still badly understood, and in current designs immature actuator technology limit the Degrees of Freedom (DoF) of the wings to just one, whereas the wings of bees and dragonflies follow complex movements, providing extreme flight performance capable of sharp turns, hovering and even backwards flight.

The TU Delft has its own FWMAV platform, the Delfly (see figure 1-1c). Development of this FWMAV began with the Delfly I, which started in 2005 as a bachelor student project. With the objective to "impress the jury of the 1st US-European Competition for MAV", the group of students managed to design a very complete package. In spite of a very small payload allowance, they were able to pack a camera, video transmitter and battery pack in the 15 gram ornithopter, even making beyond-line-of-sight flight possible. They went on to win a Technology Price in Garmisch-Partkirchen, Germany in September 2005.

The DelFly II, a continuation of the DelFly project seeking to improve robustness and decrease size and weight, was the next step. Since then many improvements have been made to the DelFly, adding or changing sensors and instruments for different applications. An even smaller version was also developed, the Delfy Micro - weighing only 3 grams and measuring 10 cm from wingtip to wingtip - but it is the Delfly II that is the subject of this thesis.

1-1 Project motivation

One of the main topics of interest regarding FWMAV is the still relatively badly understood aerodynamics that drive the flight of flapping wing aerial vehicles. The usual approach to understanding the flight dynamics of a flying vehicle is through a combination of theoretical and experimental techniques. The latter includes putting a (scale) model of the vehicle in the wind tunnel and measure the lift and drag forces generated on it and possibly visualize the airflow over the vehicle, using various techniques like surface flow visualization, particle tracer methods or optical methods. These techniques are readily applied to fixed wing aircraft and FWMAV. But where for a fixed wing it is possible to reach a pseudo static equilibrium, where the test object does not experience any movement, for a FWMAV this is generally not the case. The flapping wings cause a periodic motion of the center of gravity during normal flight, and when such a vehicle is put on a support in the wind tunnel, this restricted form of flight will not show the same aerodynamics as the unrestricted case.

But how much exactly does the support influence the dynamics and aerodynamics? Do the static measurements still correctly represent the behavior of a free flying Delfly? Questions like these have raised the curiosity of engineers of whether or not it would be possible to have the Delfly fly unsupported in the wind tunnel. With non-intrusive measurement techniques such as Particle Imaging Velocimetry (PIV) we would then be able to measure the true aerodynamic properties, which would allow aerodynamicists to compare the flow generated in both cases, and confirm or reject the validity of the static measurements.

Therefore, the goal of this thesis is

> *Devise a system to allow the Delfly fly autonomously in the wind tunnel so as to make is possible to perform free flight PIV measurements on it, using low cost sensors*

The last part of the thesis statement is added due to limited budget, restricting the total cost of the sensors to a few hundred euroś maximum.

Additional goals

The Delfly is designed for slow forward flight, with velocities up to 1 meter per second, and the primary focus will be on achieving the stated thesis goal for this flight regime. Nevertheless, the whole project was undertaken with the faster forward flight regime in mind as well, and the design was made such that forward flight should be possible with the same soft- and hardware. The project will be considered a success if the thesis goal is achieved for the slower flight regime, and all additional findings for controlling the Delfly at higher velocities are added benefit.

Besides providing us with the tools to observe the Delfly in free flight, this thesis project will also be a great opportunity to study the dynamics of the Delfly in detail. Never before has the behavior of the Delfly been monitored with high precision live during flight. Such a capability would provide a great opportunity to gain valuable knowledge about the Delfly's system dynamics, and possibly to perform system identification.

1-2 Previous research

Not many wind tunnel experiments conducted without a support have been found in the literature. Although one paper has been found where the researchers set out to study unsupported flight on a fixed wing (Nowack & Alles, 2008), and free flight (albeit tethered) flapping motion has been studied several times on biological subjects like pigeons (Biesel & Nachtigall, 1987), insects (Bomphrey et al., 2006) and bats (Hedenström et al., 2009), no literature has been found on free flight wind tunnel experiments with flapping-wing mechanical models. The reasons for this are thought to be two-fold. First of all, the support in the wind tunnel is not just a hindrance that interferes with the airflow around the model. It is actually the primary measurement tool for flight test engineers as it incorporates an intricate balance system, allowing lift and drag forces as well as the aerodynamic moments to be accurately measured. There needs to be other, non intrusive measurements techniques to make it worthwhile to pursue free flight in the wind tunnel. This is actually the case with the current research done at the TU Delft, where the focus is on visualizing the airflow around the Delfly with the use of PIV techniques.

Another, more important reason for the absence of free flight research on FWMAV is the relative infancy of these platforms and difficulties in flying it with the precision required to keep it in the center of the wind tunnel. FWMAV are a relatively new field of serious research, and the Delfly has not yet received a large deal of attention from the control department in identifying the underlying dynamic characteristics or developing advanced controllers. Moreover, the sensors required to achieve autonomous flight, such as those typically employed in larger UAV, are usually too large/heavy for the limited payload capacity of the Delfly.

Although unsupported flight of FWMAV in wind tunnels has not met any scientific attention, a lot of research on flapping-wing aerodynamics has been done, using fixed models. The basics of the aerodynamics driving flapping-wing flight are generally understood, but lots of phenomena are still left unexplained. We hope that the efforts made in this thesis project, will contribute to the ongoing research of flapping wing aerodynamics by providing a platform that allows the flapping wing aerodynamics to be appreciated in their full unrestricted freedom.

Delfly

The DelFly II is build from micron thick Mylar and thin carbon fiber rods. The whole DelFly II weighs in at 17 grams, including its on-board camera.

The Delfly was designed through a top-down approach. Instead of designing and optimizing all the subsystems separately, a total working system is designed, which is then gradually optimized. The benefit is that there is always a working platform where all the elements can be tested when optimizing. At first the Delfly was not very efficient, but continual improvements to the motor, drive train and wings brought the Delfly at the level it is now at, able to fly for roughly a quarter of an hour on a single charge. It is capable of 7 m/s forward flight, hovering and can fly even backwards at -1 m/s (Croon et al., 2009).

Many different versions of the Delfly exist, reflecting perpetual improvements on each design or modifications for special purpouses, but the general elements have remained the same. The wings are made from thin Mylar with carbon rods for the shape and strength. This allows for a very lightweight and flexible wing. The flexibility is very important for the aerodynamic characteristics, as it needs to be able to deform during the flapping movement to achieve the right aerodynamic shape. The deformation involves variable angle of attack, favorable camber, location of the axis of rotation, span wise bending and torsion, leading edge heave motion, wing-wing interaction and wing flexing during rotation (De Clercq et al., 2009).

The tail used to be also made from Mylar foil and carbon rods, but the complexity in manufacturing and the fragility have led to a redesign with a different material. Polyethylene, which is more impact resistant and easier to work with (albeit slightly heavier) is now used. Combined with advancements in servo technology, the new design allows the rudder- and elevator-servos to be embedded in the horizontal and vertical tail plane. The servos are connected to their respective control surfaces via small carbon rods. This resulted in a robust

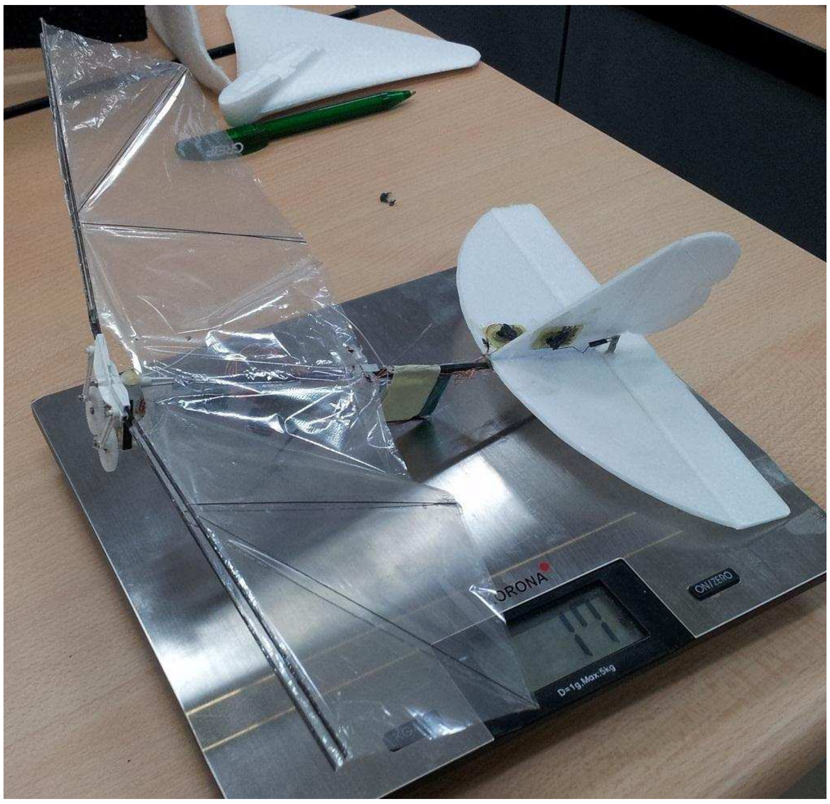


Figure 1-2: The Delfly II on a scale.

and compact solution, an improvement over previous designs where the servos were mounted on the tail-beam and more exposed during crashes.

The main components are located in between the wings. It houses the custom-made brushless motor and gear train, and a motor controller is situated right behind the motor. A battery is placed more aft. In between sits a radio receiver that controls the servos and motor.

Controllers

Besides flying FWMAV manually, a few efforts have been undertaken to design controllers to make possible various degrees of autonomy.

Several autonomous flights have been made at the MAVLAB, TU Delft with the Delfly II. Three experiments are described in (Croon et al., 2009). Firstly, using an external camera, the team successfully achieved automatic height control. The experiment was set up as follows. The Delfly was made to fly in a circle by a human operator to keep it in view of an external stationary camera. After trimming the vehicle, the control algorithm would subtract two subsequent gray scale images taken by the external camera to detect motion, the Delfly being the only moving object in view. The median of the y-coordinates of all the detected moving pixels was taken as the estimated height. This was fed back through a proportional gain controller, which would adjust the throttle to maintain altitude.

The second experiment was conducted using two on-board cameras. The main goal was to assess whether it was possible to achieve full autonomy by controlling the Delfly on the basis of just on-board processed images. The Delfly employed two cameras, one forward looking, and one downward looking. The forward camera was used to estimate the turn rate r by calculating the optic flow using the fast Lucas-Kanade algorithm on a set of 40 corners in a 128x128 pixel resized image. The resulting horizontal components of the flow were then averaged to obtain an estimate of r. The downward camera tracked a laid-out track of white sheets of A4 paper. From these images the height h, relative heading $\delta\psi$ and the lateral offset δx were estimated. Height information came from the amount of whiteness in view. The closer it would get to the floor, more of the image would be filled by the white paper. Utilizing edge detection the sides of the paper track were observed, of which the relative rotation gave an indication of the heading error, and the horizontal offset an indication of track error. A proportional gain on the track error gave the reference for the heading. The current heading $\delta\psi$ was subtracted of this signal, fed through a proportional gain, and compared to the current turn rate. The error of these two signals was again fed through a proportional gain and provided the control signal for the rudder. Another proportional gain relayed the height information to the motor controller controlling altitude. Successful complete autonomous flight was achieved in these experiments.

Where the previous experiment required preparation of the environment (external camera, sheets of paper on the ground), the third flight experiment for the Delfly II had the goal of controlling the height in an unprepared room. Using a single forward looking camera and employing a texton based image process technique, successful height regulation was achieved.

It should be noted that in any of the before mentioned experiments, the elevator was never used. This is because all test flight were done in slow forward flight where the thrust vector points almost completely upward, and the angle of attack is very high. All height control is done in this case by the motor controller. The lack of elevator control means the forward speed, and therefore the forward position, was not controlled.

Most recently, at the University of California, Berkeley, Baek (2011) has undertaken a project where the goal was to fly a FWMAV towards an infrared LED on the wall. Use was made of similar hardware (on-board IR camera, inertial sensors) and they have shown that it was possible to fly a straight line, recover from lost sighting of the reference LED and have a functional auto pilot with the indicated hardware.

All these experiments have lacked the capability of following a very precise reference signal though, as would be required for achieving the goal of this thesis.

Dynamics

Most research on the Delfly has focused on the aerodynamics of the flapping wings and the possibilities and/or limitations of on board cameras. The former has been mainly performed on the wings alone (thus excluding the tail plane), although just recently a study of the 3D flow field of a complete Delfly has been finished here at the TU Delft. The latter research subject considered the Delfly more or less as a capable flying platform on which to test systems and algorithm. Because of the inherent stability, these models never required detailed knowledge of the system, and therefore there has never been a thorough research on the dynamics. As of yet, no workable dynamic model has been constructed of the Delfly II. This makes it much more difficult to design a controller, especially because it is not possible to do simulations for the design and validation of the controller. In order to get the performance of the Delfly to the next level, it might be beneficial to invest time and effort in a good dynamic model.

Although there is no tested model available for the Delfly, other groups have tried to derive a theoretical model for FWMAV's. These are based on the time averaging principle to average the forces generated by multiple flap cycles, because the complex aerodynamics involved are not well understood. (Baek, 2011) derived a model based on simple kinetic relationships, assuming constant (average) thrust during the flapping cycle. The amount of thrust was derived from flight experiments and static setup measurements. Desirable as it may be to have a complete model of the Delfly, derivation hereof is far outside the scope of this thesis.

Particle Imaging Velocimetry

PIV is the flow visualization technique that is the main motivation behind this thesis. It is a non-intrusive flow measurement technique that is capable of measuring the entire volume under investigation. The basic principles of PIV will not be elaborated here, but the interested reader is referred to Willert (1997); Raffel, M., Willert & Wereley, S.T., Kompenhans (2007). PIV uses a setup of a single or dual high speed cameras, depending on whether 2D or 3D measurements are desired, and a strong laser. Small particles are then seeded into the air flow. The laser illuminates these particles for a very short duration, during which the cameras register the position of the particles in the illumination plane. This way a quick succession of snapshots can be taken of the flow from which the 2D flow field can be deduced using correlation techniques. When using two cameras the out-of plane velocity can also be determined. The technique has been used in recent years for studying the aerodynamics involved in the flight of the Delfly, see Croon et al. (2009) and Groen (2010).

In combination with a control system that keeps the Delfly in the focal point of the cameras, it would allow us to measure the airflow around the Delfly in free flight for the first time.

1-3 Thesis outline

In this chapter, Chapter 1, the project motivation is outlined, and an overview of previous research in the field of FWMAV, especially for the Delfly has been given.

In Chapter 2 an overview of the design that was made is elaborated. It will explain the approach to achieving the thesis goal, and give a detailed overview of all the hardware that was developed for this project. Chapter 3 explains the theory and practical side of estimating the state using the hardware from the preceding chapter. Chapter 4 treats the calibration that is done for all systems prior to testing. Chapter 5 elaborates on the proposed control structure.

In Chapter 6 issues regarding the real implementation are discussed. Chapter 7 shows the results of the flight tests, and whether or not it has been proven possible to achieve autonomous flight with the Delfly using the proposed system. In Chapter 8 a discussion on these results is presented, and Chapter 9 will give an overview of the most important conclusions. Finally, Chapter 10 will give recommendations for further research, based on the experiences gained during this project.

Chapter 2

Design

This chapter will give an overview of the design of the whole system. First an overview of the complete design is given, after which every element is elaborated on in more detail. During the design the following considerations were taken in account. These are derived directly from the thesis goal stated in the previous chapter.

1. The PIV measurements are done by 1 or 2 camera's which have been calibrated very precisely and don't offer a large field of view. For 3D PIV the three-dimensional space is scanned in 10 or more slices, which can not performed in rapid succession because the camera's need to focus at a new depth. So in order to perform 3D PIV measurements the Delfly need be able to consistently hover for at least a few flap cycles in the same location, preferably within a centimeter.

2. Because the experiment will be performed in the wind tunnel, there are little restrictions on the reliance on ground systems, as the design need not be flexible in terms of operating environment. What does restrict the systems is that they should not interfere with the free stream flow or any of the PIV measuring equipment. So all additional camera's, reference points or other aids should be able to be set up in such a manner as not to interfere with the PIV setup.

3. The Delfly should also not be customized too much. Because the focus is on the qualitative aerodynamic behavior rather than quantitative, it means there is still some room in customizing the on-board systems and for example shift the center of gravity a few percent, but the main wings and tail design should not be altered.

Testing the system will be done in two stages: first the hardware elements are tested on a quad copter, which is less fragile and therefore more suited for the preliminary tests, and when all systems are working everything is transfered to the Delfly. Because of this, certain trade-offs had to be made to accommodate both platforms with the same hardware.

2-1 Overview

The approach to achieving autonomous flight in the wind tunnel is as follows: a tracking system will measure the position of the Delfly. On the basis of this position information and an on-board camera that looks at a Light Emitting Diode (LED) as a heading reference placed in front of the Delfly, the (on-board) control algorithm will try minimize the position error and follow the LED by commanding the actuators appropriately.

The total system can be split up as shown in Figure 2-1. It consists of a ground section and a section on-board the Delfly, each further subdivided in their main subsystems.

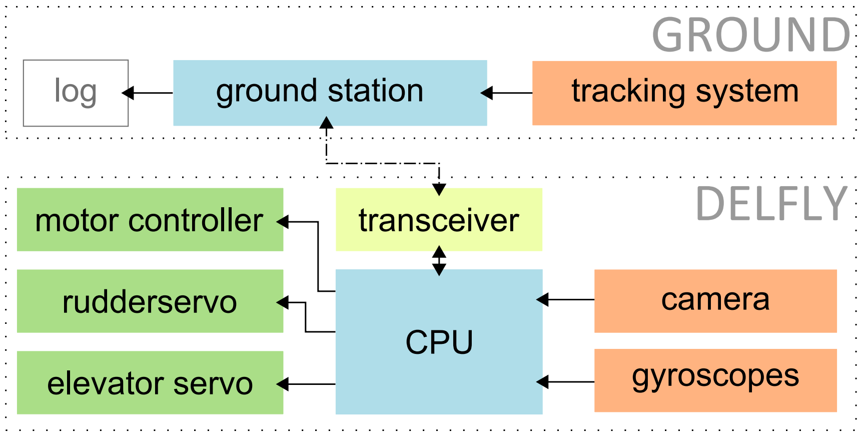


Figure 2-1: Schematic overview of all the system components and their relation

The ground station is a 2.3 Ghz dual-core laptop. It is connected by Bluetooth to the tracking system as well as to the Delfly. It sends the position and velocity information as calculated from the tracking system to the Delfly so it can be used by the on-board autopilot. All the information gathered from both the downlink (gyro and camera data) and the position and velocity data is logged for analyses.

The tracking system consists of two Wiimote cameras. These were chosen because they combine a high tracking rate (83Hz) with low cost (around 50 euro each) and because they are easy to interface with a laptop (communication is done over Bluetooth using a special Wiimote library).

All the main on-board components are visible in the side view of the Delfly, Figure 2-2. The CPU computes the actuator inputs based on sensory information. It is interfaced with a stripped down Wiimote camera, three gyroscopes and a small Bluetooth module. It is also connected to the actuators: the motor, rudder and elevator.

In order to have more flexibility in testing, a moving platform was developed to allow forward flight tests to be performed not in the wind tunnel, but a larger space like a sports hall. This approach makes it easier to test and doesn't require the wind tunnel to be turned on and off and allows for more testing hours, more flexibility and less cost. Also, all systems that are designed for the Delfly are first flight tested using a quad copter, because of the fragility of the Delfly. The quad copter will allow for more flight hours because it is robust and can survive a few crashes. The results of these tests are not discussed in the main report, but can be found in Appendix D. The final testing and proof of concept will be done with the Delfly

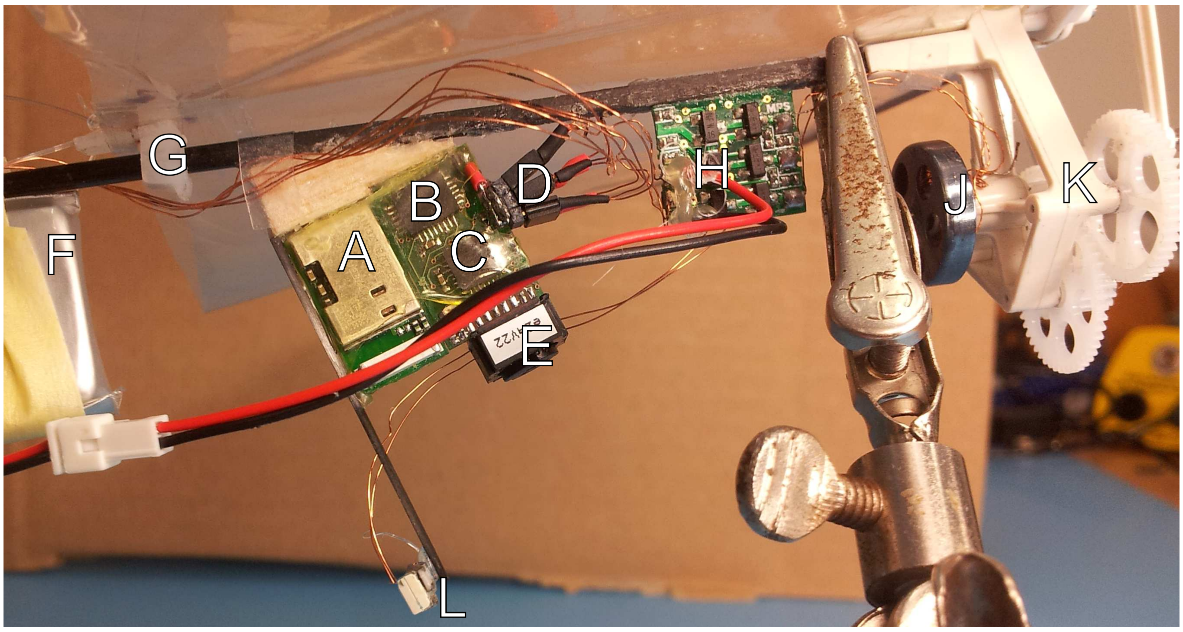


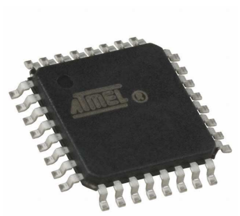
Figure 2-2: Side view of the Delfly. **(A)** Bluetooth module, **(B)** Dual gyroscopes, **(C)** CPU, **(D)** Servo and motor connectors (from top to bottom: motor controller, elevator servo, rudder servo), **(E)** WiiMote Camera, **(F)** 180mAh LiPo battery, **(G)** Trailing edge tensioner, **(H)** Motor controller, **(J)** Brushless motor, **(K)** Gear housing, **(L)** Tracking LED

in the Open Jet Facility (OJF), a large wind tunnel that allow plenty of space to test free flight of the Delfly.

2-2 On-board systems

The on board system is composed of gyro's, a Bluetooth module, a camera, and a central processing unit. All sensors are connected directly to the Central Processing Unit (CPU), using the built-in Analog-to-Digital Converter (ADC) for analog or digital pins for digital signals.

2-2-1 CPU



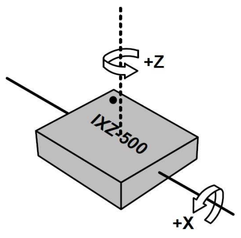
The CPU is heart of the auto-pilot system. It's a ATmega88PA capable of 20 million instructions per second (MIPS) at 20Mhz, but because it is running on a 3.3V power supply, the clock speed is turned down to 8 Mhz. It has 2 8-bit timers and one 16-bit timer. By utilizing the dual compare functionality the 8 bit timers provide, they can be used to control both servos and the motor. The 16 bit timer is only used when connected to the quad copter, and is used to time the communication with the quad copter.

The camera is connected to the CPU by the Inter-Integrated Circuit (I2C) port, while the two gyro's are connected to the ADC. The third gyro, which was later retro-fitted to the board, is also interfaced through I2C.

Because the chip only supports one Universal Asynchronous Receiver/Transmitter (UART) and both the serial connection to the quad copter and the Bluetooth module use UART a solution had to be found. By using two pin-outs as a virtual serial port, this problem was solved, although because of the extra overhead the baud rate was capped at 9600 Bd. Because the quad copter could only communicate at a baud rate of 38400 Bd, the slower virtual serial port was used for the ground link, capping data transfer to only $\frac{9600}{(8+1)} \approx 1067B/s$. This means that the total data budget for both down and up link could not exceed $\approx$ 42 bytes per loop, and in practice had to be even smaller. The data had to be packed in a tight custom package. Although this limit only strictly held for implementation on the quad copter, the same protocol remained after the switch to the Delfly, because it would require more time to rewrite the protocol and test it again. This was just one of the trade-offs that had to be made by choosing the quad copter as a testing platform.

Because a floating point operation library would take up a too large chuck of the 8kB memory available on the chip, a further restriction was posed by the CPU. The software running on-board was limited to fixed point operations, which makes every calculation a bit more complex. Round-off errors had to be considered at all times and appropriate pre-scaling was implemented where necessary. The limited memory and computing power also meant that the software had to be written with care and algorithms were kept as simple as the application would allow.

2-2-2 Gyros



The bi-directional analog gyro on board is the InvenSense IXZ-500. This MEMS-gyro packs integrated amplifiers and low-pass filters (cut-off frequency 140 Hz) and measures only $4 \times 5 \times 1.2$[mm]. It can measure up to $500°s^{-1}$ in high range mode, or up to $110°s^{-1}$ in high precision mode, with a sampling rate up to 9kHz (limited by the processors ADC). It is mounted in such a way that it provides angular rate information for rotations around the horizontal and the vertical axis of the camera.

A yaw gyro, required for yaw damping, was not considered at first because the Delfly always showed very stable yaw-behavior because of the pendulum stability of the aft center of gravity position. After some manual test flights this assumption was invalidated with the observation of heavy undamped yaw oscillations as a consequence of performing a rudder impulse input. A yaw damper was believed to be able to remedy this behavior.

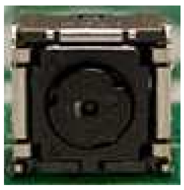
A third gyro was therefore fitted. One of the digital three gyro's on the InvenSense IMU-3000, cut out from an Aspirin board, was added to the autopilot board and interfaced through I2C. The Y-axis of this auxiliary gyro was lined up with the Z_c axis of the on-board camera of the Delfly, provided the yaw angular rate. It had a range of $\pm 500°s^{-1}$. An I2C interface was established by soldering it to the I2C connections of the on-board camera.

2-2-3 Bluetooth

First the BTM411 Bluetooh-module was used as the digital communication module. Later on this was replaced by a even smaller bluetooth

unit, the Panasonic PAN1321 for on the Delfly. The Bluetooth module is theoretically capable of 2.1 Mbit/s, although in practice there are many limiting factors such distance and other nearby sources of 2.4Ghz radiation. On the Delfly, the chip was interfaced with the CPU through the chips UART connection at a baud rate of 38400 Bd.

2-2-4 Camera



The Wiimote camera was chosen because it is cheap, fast, small, and all the preprocessing is already done by the build-in chip. The camera unit interprets the image, finds the four brightest Infra-Red (IR) point, and calculates the respective x and y coordinates, all at a frequency of 200 Hz. The preprocessing means that the only data that needs to be transferred between the camera and the CPU are the 4 coordinate pairs of the LEDs, which saves a lot of overhead for the on-board processor. It runs at a clock speed of 25Mhz, provided by a separate crystal. It is interfaced with the CPU on the I2C bus.

Parameter	magnitude	unit
Weight	0.33	gr
Dimensions (HxWxD)	8x8x4	mm
Operating voltage	3.3	V
Sample rate	max 200	Hz
Resolution	128x96, sub pixel interpolated to 1024x768	pixels
Precision	0.03	°
Nr. of tracking points	up to 4	-
Clock speed	25	Mhz
Field of view	44 horizontal, 33 vertical	°
Most sensitive to	940	nm

Table 2-1: Hardware specification of the camera from a WiiMote

As an IR source use was made of the SFH485 for ground purposes, and the much smaller and lighter OP180 for on-board of the Delfly. These are wide angle LEDs so that they are visible at a wide range of viewing angles. They are bright enough to be tracked reliably at a distance up to 2 meters, enough under nominal flight conditions in the test-setup.

The on-board IR LED, used for tracking the Delfly (referred to as the *tracking LED*) is mounted on a small carbon rod, so the battery wouldn't occlude it. The LED on the extension rod is clearly shown in Figure 2-2, indicated by an L..

2-3 Ground systems

2-3-1 Laptop

The ground segment of the hardware consists of a 2.30 Ghz Dual-Code laptop running Windows 7. It acts as the ground station, and communicates with both the tracking cameras and the Delfly. Its task is to process the sightings of the two tracking cameras, calculate from this the position and velocity of the Delfly, send this state information up and receive back log data. This is shown in the flow diagram in Figure 2-3. It runs the in-house developed smartUAV software, which has similar capabilities as for example Simulink, but with the ability to design fully integrated custom modules, and with the low level capabilities of C++ such as accessing the Windows Bluetooth stack.

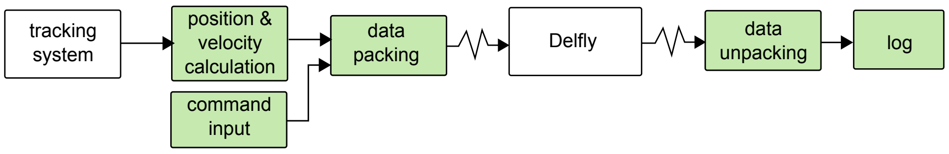


Figure 2-3: Flow diagram of ground station functionality. The green blocks represent the ground system. The waveform arrow represents the wireless Bluetooth connection to the Delfly.

2-3-2 Wiimotes

Two Wiimotes are used as the tracking cameras. They are low-cost (around 50 euro), easy to interface through Bluetooth, and have high precision. Of course there are better options than using Wiimotes for tracking. Off-the-shelve solutions like the Vicon-system offer superior tracking performance, but they can be very expensive (> $10.000).

But for 100 euro we can use two Wiimote cameras and create a system that offers sub-millimeter precision at a rate of up to 80 Hz, very near to the performance of the Vicon tracking system. The difference is that the tracking volume is a lot smaller due to the relatively small viewing angle of the Wiimotes. Also, the Wiimotes need an aactive tracking element (an IR LED in this case), instead of passive reflectors. The latter is no real issue, because a small IR LED is easily fitted on the Delfly.

The Wiimote camera specifications are the same as the on-board camera which are listed in Table 2-1, with the exception of the update rate. The update rate is lower, only 80 Hz, because of the Bluetooth layer causing additional overhead. The camera angular precision of 0.03 degrees, translates to about 0.8 mm precision at the nominal distance of 1.5 meters. Use was made of the *WiiYourself!* library by Gl.itter (2010) to interface the Wiimotes with smartUAV. This open-source code provides access to the full range of functions, although only the camera functionality is used. Being written in C++, it could fully be integrated with the triangulation algorithm that calcuulates the position as a single block in smartUAV.

Chapter 3

State estimation

This chapter provides the theoretical background in how the position, velocity, attitude and angular rate can be determined from the information available given the hardware from the previous chapter. This information will be used by the on-board controller.

The information that the employed hardware can provide is: the coordinates of the infrared tracking LED from the two tracking cameras, the coordinates of the projection of the reference LED in view of the on-board camera, and the angular rates from the triple gyro.

From the two tracking cameras we can estimate the position of the Delfly. The Linear-LS method is used to translate the two coordinate pairs of the cameras to a 3D position. By taking the derivative also the velocity can be estimated.

From the position information and the on-board camera, the attitude of the Delfly can be determined, using basic geometrical relationships. The angular rates follow directly from the gyro's, and a filter is designed to reduce the noise and improve the quality of the signal. These algorithms run on-board, which means they have to be of low complexity considering the relatively low computation power of the CPU.

3-1 Definitions

To provide a meaningful elaboration on how the state of the Delfly is measured and estimated, first a proper definition of the reference frames that are employed is needed.

The wind tunnel reference frame $\mathcal{F}_w$ is shown in Figure 3-1. This right-handed inertial reference frame is indicated by the subscript w. Its origin is roughly positioned in the middle of the the tracking box spanned by the tracking cameras. Through calibration with a calibration board it is exactly aligned with the wind tunnel. The X_w axis points into the wind, the Z_w axis downwards along the local vertical and the Y_w axis points to the right when facing the wind.

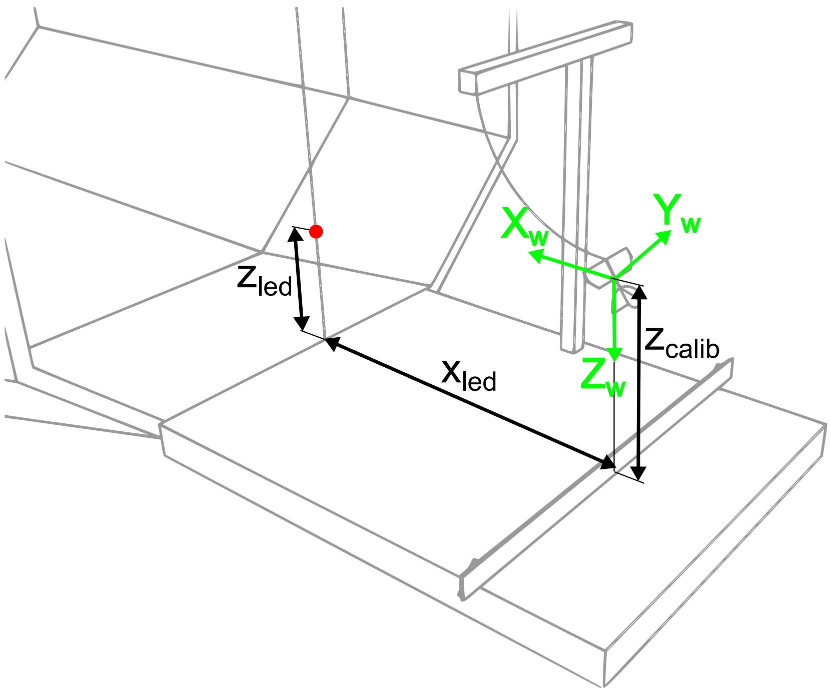


Figure 3-1: Schematic view of the OJF wind tunnel. The octogonal wind tunnel nozzle is on the left, with a LED (red) attached to a fish wire stretched over the opening. The Delfly is attached to the T-shaped beam by means of a thin wire. The tracking cameras are mounted on the horizontal beam across the platform, and look towards the Delfly.

Also indicated in the figure is the distance between the platform and the origin of the wind tunnel reference frame z_{calib}. The position of the reference LED is defined by the distances x_{led} and z_{led}. It is positioned in the X_w,Z_w plane, so y_{led} is by definition zero.

In the Figure 3-1 the two tracking camera reference frames, $\mathcal{F}_{C_1}$ and $\mathcal{F}_{C_2}$ are also indicated. The camera on the left is always indicated as camera 1, and the blue Wiimote is placed here. This is important because the cameras are individually calibrated for their internal parameters. The Z axes of both cameras are pointing approximately towards the origin of the wind tunnel reference frame $\mathcal{O}_w$.

A uncalibrated wind tunnel reference frame $\mathcal{F}_{\hat{w}}$ is defined in Figure 3-2. The uncalibrated reference system is used as an intermediate step in expressing a position in the calibrated wind tunnel frame $\mathcal{F}_w$, used out of convenience. It was defined at a certain position from one of the cameras, and is fixed to one of the cameras of the tracking system. Later on, $\mathcal{F}_{\hat{w}}$ is aligned with the orientation and position of $\mathcal{F}_w$ through calibration.

The axes of the individual Wiimote cameras are defined as in Figure 3-3. This reference frame is called $\mathcal{F}_{C_n}$, where n indicates which camera it concerns. By definition, $\mathcal{F}_{C_1}$ is the reference frame for the left camera (blue Wiimote) and $\mathcal{F}_{C_2}$ for the black Wiimote camera, as is also indicated in the wind tunnel setup overview, Figure 3-1.

For the Delfly the standard convention used in the aerospace industry is followed. The body axes are defined as in Figure 3-4, where a subscript D is used to indicate the Delfly body reference system.

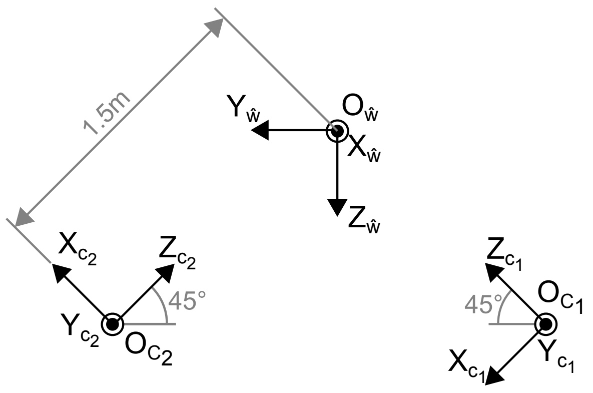


Figure 3-2: Definition of the camera reference frames $\mathcal{F}_{C_1}$ and $\mathcal{F}_{C_2}$ and the wind tunnel reference frame $\mathcal{F}_w$, as they are mounted on the platform.

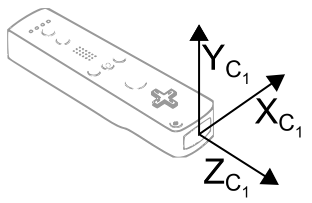


Figure 3-3: Axis definition of a Wiimote

The autopilot Printed Circuit Board (PCB) is rotated 17 degrees around the Y_D axis. The on-board camera reference frame is defined with the X-axis pointing upward, the Y-axis pointing to the negative Y_D axis, and Z-axis pointing in the flight direction, as is shown in Figure 3-4. The camera axes are indicated with a C subscript.

The gyroscopes are mounted in the on-board camera axes directions. Therefore they measure the angular rates p_c, q_c, r_c, which are the rotations around resp. the X_C, Y_C and Z_C axes.

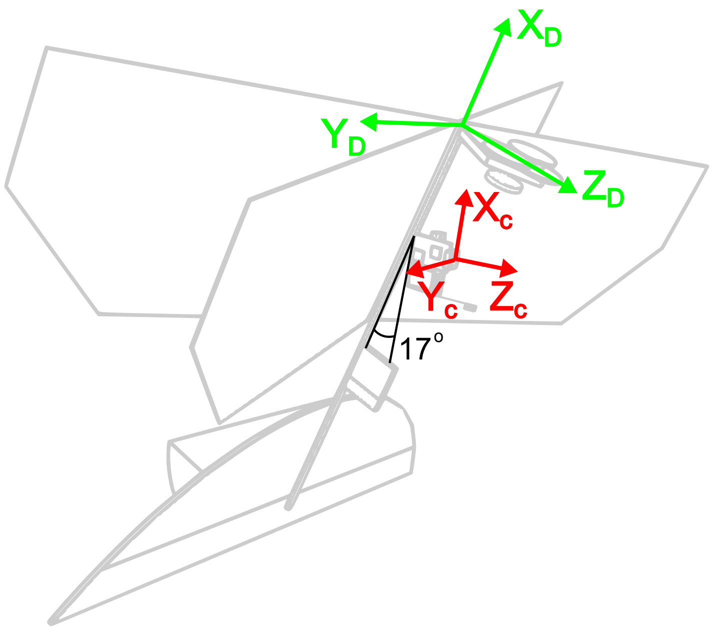


Figure 3-4: Axis definition of the Delfly body reference frame $\mathcal{F}_D$ and the on-board camera reference frame $\mathcal{F}_C$.

3-2 Position

The position of the Delfly is measured by the two Wiimotes that are placed at either side of the wind tunnel, looking up at an angle of 45° relative to the horizontal. This puts the cameras orthogonal of each other, providing the highest precision. The camera's are positioned a little over 2 meters apart, so the line-of-sights intersect at an distance of 1.5 meter. This was done to optimize the tracking volume. Because the cameras have an effective range of about 2.5 meter any farther and the LED can not be tracked reliable, any closer and the tracking volume would get too small because of the small viewing angle.

The algorithm employed for calculating the position on the basis of two 2D coordinate pairs, is a linear triangulation method called the Linear-LS method as described in Hartley & Sturm (1997). But before moving on, it is good to first introduce homogeneous coordinates as a way to express coordinates, as well as the basic camera theory necessary for implementing the algorithm.

3-2-1 Homogeneous coordinates

To represent a 2D point, normally a two element vector is used of the form $(x, y)^\top$. Likewise, a point p in 3D space is represented by $p = [\,x\ y\ z\,]^T$. In order to make these vectors invariant to arbitrary scaling factors and to represent points at infinity with finite coordinates, a third (2D) respectively fourth (3D) coordinate is added. Let's illustrate this for the 2D case, which can be readily be extend to the three dimensional case. The coordinates of a point x,y are in homogeneous coordinates represented by the ratio of the first two elements (X,Y) and the last element (Z). So x = X/Z and y = Y/Z. Because we are dealing with ratios, the vector is invariant to scaling. Also, setting the last element to zero means the point represented is at the line through x,y stretching all the way to infinity.

3-2-2 Camera model

A camera can be represented by the so-called pin-hole camera model. An 3D scene view is projected into the image plane, creating a 2D image of the scene. This can be represented by a perspective transformation of point **x** in homogeneous world coordinates into the projection point **u** in homogeneous image coordinates

$$s\mathbf{u} = N[R|t]\mathbf{x}$$

or

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \alpha & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3-1}$$

The projection $\mathbf{u} = [u\ v\ 1]^T$, where u and v are the observed point coordinates, is defined up to a scale factor s. The intrinsic parameters are represented by a 3x3 matrix **N** called the matrix of *intrinsic parameters.*

The projective transformation from world coordinates to the camera image plane also contains the rotation and translation of the camera relative to the point. This is included in the 2nd matrix and these parameters are called the *extrinsic parameters.* R is 3x3 rotation matrix and **t** the translation vector.

These two matrices combined represent the complete projective transformation and can be combined into the so-called camera matrix $P = N[R|t]$.

3-2-3 Linear-LS method

The camera observation **u** is defined by $\mathbf{u} = P\mathbf{x}$, with $\mathbf{x} = (x, y, z, w)$ the position vector and P is the camera matrix. Vector **u** is in homogeneous coordinates, i.e. $\mathbf{u} = s(u, v, 1)^\top$, where u and v are the observed point coordinates, and s is an unknown scale factor. Denoting $p_i^\top$ as the ith row of matrix P, we can rewrite $\mathbf{u} = P\mathbf{x}$ as

$$su = \mathbf{p}_1^\top\mathbf{x}, \quad sv = \mathbf{p}_2^\top\mathbf{x}, \quad s = \mathbf{p}_3^\top\mathbf{x}$$

Eliminating s using the last equation, we have

$$\begin{aligned} u\mathbf{p}_3^\top\mathbf{x} &= \mathbf{p}_1^\top\mathbf{x} \\ v\mathbf{p}_3^\top\mathbf{x} &= \mathbf{p}_2^\top\mathbf{x} \end{aligned} \tag{3-2}$$

So from two views we have 4 linear equations in the coordinates of $\mathbf{x}$, which can be written in the form $A\mathbf{x} = \mathbf{0}$ with A being a 4x4 matrix. Assuming the target is not at infinity, we set $\mathbf{x} = (x, y, z, 1)^\top$ so we reduce the set of homogeneous equations to a set of four non-homogeneous equations with three unknowns. In order to combine the measurements, we have to express $\mathbf{x}$ in the reference frame of the second camera. For this we use the rotation matrix R and translation vector t, which describe the relative position and orientation of the two cameras. They are for the moment assumed to be known. We also rewrite R and t in homogeneous coordinate format.

The camera matrix for camera 1 is

$$P_1 = N_1[\mathbf{I}_3 \;\; \mathbf{0}]$$

while camera matrix 2 has to be rotated and translated as well

$$P_2 = N_2[R \;\; T]$$

The exact determination of the intrinsic parameter matrices N_1 and N_2, as well as the rotation and translation matrices R and T, is left for the next chapter on calibration, Chapter 4.

Now using equation 3-2 and considering both measurements with the camera matrices P_1 and P_2, we can construct matrix A as

$$A = \begin{bmatrix} u_1\mathbf{p_1}^\top(3) - \mathbf{p}_1^\top(1) \\ v_1\mathbf{p_1}^\top(3) - \mathbf{p}_1^\top(2) \\ u_2\mathbf{p_2}^\top(3) - \mathbf{p}_2^\top(1) \\ v_2\mathbf{p_2}^\top(3) - \mathbf{p}_2^\top(2) \end{bmatrix} \mathbf{x}$$

where the number in brackets indicates the row vector taken from the respective matrix. Because of the assumption of the homogeneous coordinate being finite, the fourth element of $\mathbf{x}$ is 1 we can bring it to the right hand side of the equation. Writing A as $[\mathbf{a}_1 \; \mathbf{a}_2 \; \mathbf{a}_3 \; \mathbf{a}_4]$ with each vector $\mathbf{a}_n$ representing a column, we get

$$\left[\mathbf{a}_1 \; \mathbf{a}_2 \; \mathbf{a}_3\right] (x, y, z)^\top = -\mathbf{a}_4$$

This non-homogeneous equation can be solved by using the left pseudo inverse

$$\mathbf{x} = -(A^\top A)^{-1} A^\top \mathbf{a}_4$$

$\mathbf{x}$ now expressed the position of the tracking LED relative to camera 1, and can be written with subscript C_1 to indicate this fact, so $\mathbf{x}_{C_1} = \mathbf{x}$

A translation and rotation is then applied to express the coordinate in the uncalibrated wind tunnel reference frame. The translation and rotation required to express a coordinate $\mathbf{x}_{C_1}$ as a coordinate in $\mathcal{F}_{\hat{w}}$, can be seen in the side view of the tracking camera setup, Figure 3-2. We can write the transformation as

$$\mathbf{x}_{\hat{w}} = R_{C_1}^{\hat{w}}(\mathbf{x}_{C_1} + \mathbf{T}_{C_1}^{\hat{w}}) \tag{3-3}$$

where $\mathbf{x}_{\hat{w}}$ is the coordinate in $\mathcal{F}_{\hat{w}}$, $R_{C_1}^{\hat{w}}$ the rotation matrix describing the rotation from $\mathcal{F}_{C_1}$ to $\mathcal{F}_{\hat{w}}$, and where $\mathbf{T}_{C_1}^{\hat{w}}$ describes the position of the origin $\mathcal{O}_{\hat{w}}$ relative to the origin of $\mathcal{O}_{C_1}$, expressed in $\mathcal{F}_{C_1}$.

From the side view of the tracking camera setup the rotation matrix and translation vector are determined to be

$$\mathbf{T}_{C_1}^{\hat{w}} = \begin{bmatrix} 0 \\ 0 \\ 1.5 \end{bmatrix} \qquad R_{C_1}^{\hat{w}} = \begin{bmatrix} 0 & 1 & 0 \\ \cos 45^\circ & 0 & \cos 45^\circ \\ \cos 45^\circ & 0 & -\cos 45^\circ \end{bmatrix} \tag{3-4}$$

Accuracy

The accuracy of the hardware and algorithm was checked by placing the two Wii-Motes perpendicular to each other on the edge of the milling table. They were subsequently calibrated and a LED was placed on the milling head. By moving the LED in a predetermined pattern, later the pattern could be verified to correspond to the correct pattern.

The setup was tested with a dog-leg pattern. The LED was moved 300 mm from the left to the right, and then 200 mm backwards (positive z direction). The results are shown in Figure 3-5.

The standard deviation of the error is 0.3734 mm in the x-direction and 0.6378 mm in the z-direction. It can be seen that the error shows a saw-tooth behavior, probably caused by the pixel rounding of the cameras. There are two 'periods', one short period caused by pixel-jumping in the perpendicular camera, and a longer period caused by rounding in the longitudinal camera. All in all, this shows sub-millimeter precision for the position estimate.

3-3 Velocity

The velocity is obtained by taking the discrete derivative directly from the position measurements.

$$v_k = \frac{x_k - x_{k-1}}{\Delta_T}$$

where v_k is the velocity at time step k, x_k the position at time k and Δ_T the time step size.

Because all the high frequency noise on the position estimate is amplified when taking the discrete derivative, a filter needs to be developed.

The filter is composed of two parts. The first part filters out spikes in the velocity signal caused by 'holes' in the position signal. Because sometimes the LED would not register on

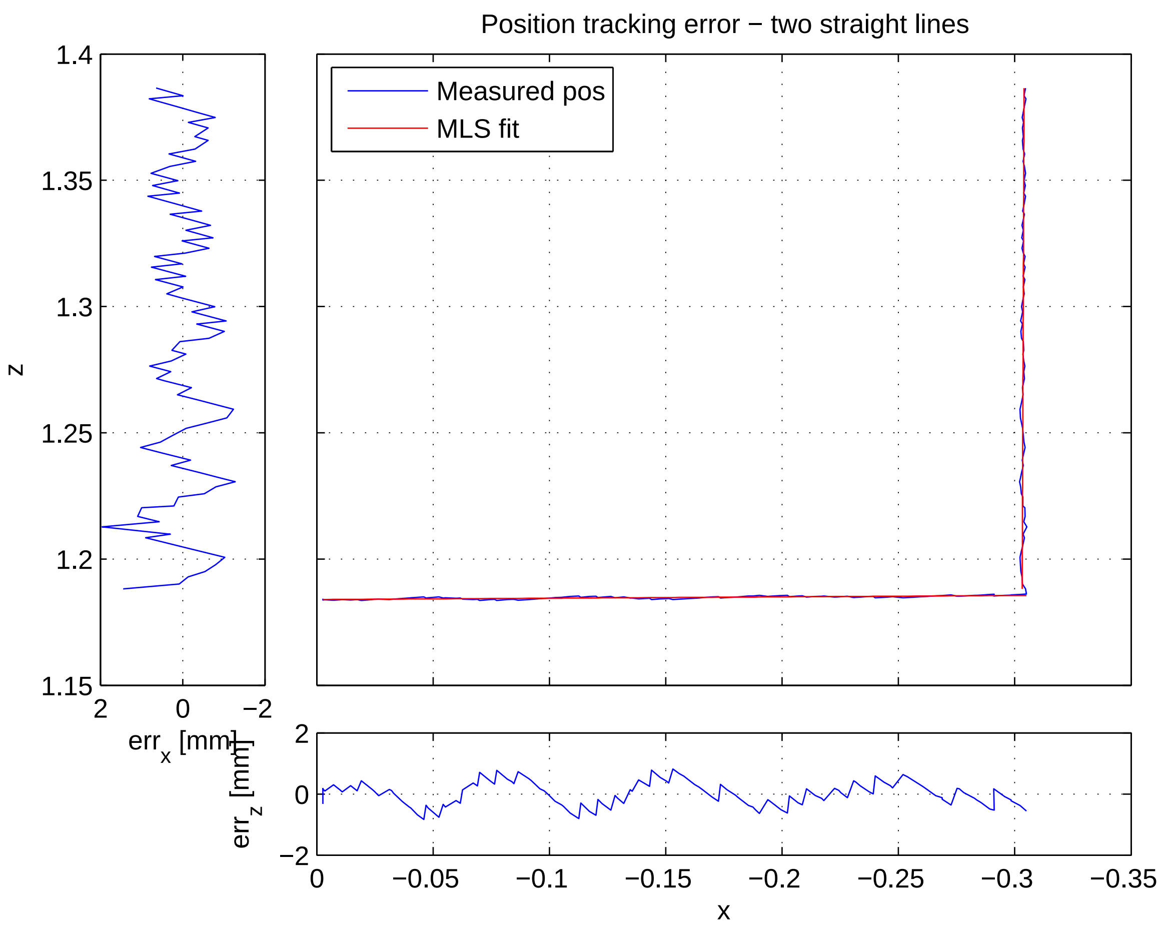


Figure 3-5: Calibration track with corresponding errors. Coordinates are expressed in the camera reference frame $\mathcal{F}_{C_1}$.

the cameras, the position signal would be zero-order-held. So when a new position registered, a spike would appear in the velocity signal because of the sudden jump in position. The first part of the filter would therefore only pass a velocity signal if it was based on two consecutive position measurements, and not if it was based on a zero order hold. A flag from the tracking system would indicate whenever a position measurement was correctly taken, therefore providing the information needed for this filter.

The second part of the filter is a Butterworth filter. The right parameters for the filter were selected by analyzing the raw measurement in MATLAB and trying different values to see which gave the best result. A result of this process can be seen in Figure 3-6.

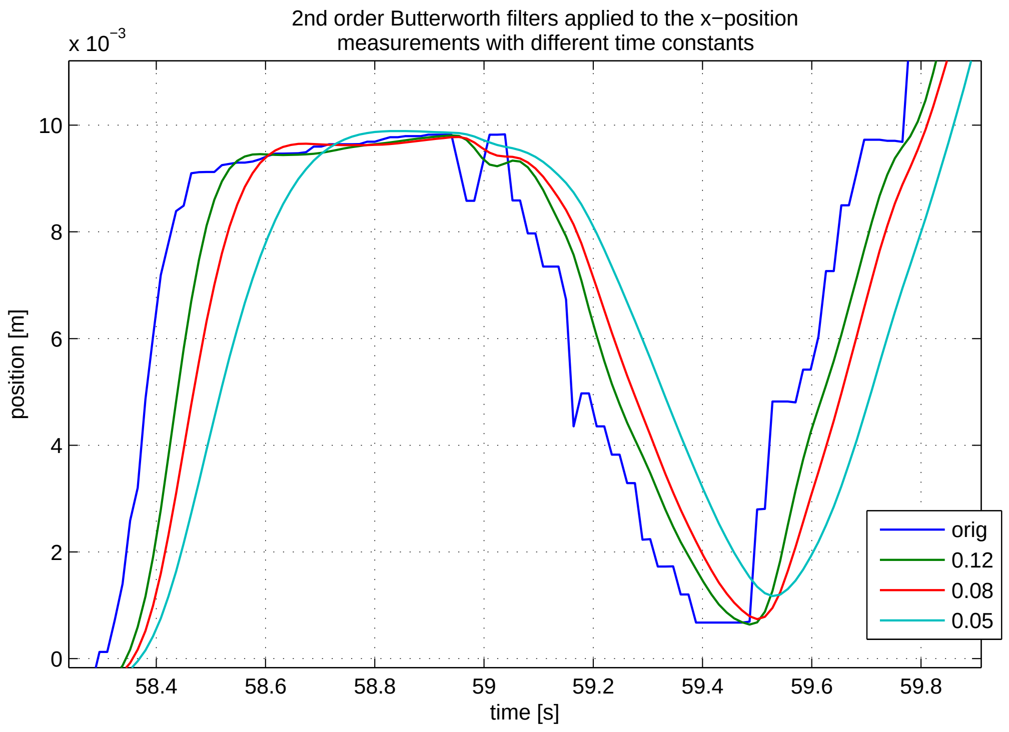


Figure 3-6: Three different 2nd order Butterworth filters applied to the raw position signal.

For the implementation of the Butterworth filter in the discrete case in the smartUAV software, use was made of the Butterworth functions from the *math/iir* library. The input parameter that represents the cut-off frequency for these functions is given in a fraction of π. This is calculated as follows:

$$f = 2\frac{f_c}{f_s}$$

where f is the input parameter between [0..1], f_c the cut-off frequency and f_s the sample frequency. So when we want to apply a cut-off frequency of 25 Hz when sampling at 100 Hz (1/4 of the sampling frequency), the filter parameter would be 0.5. In other words, the parameter given is a fraction [0..1] of the Nyquist frequency of the signal.

During the initial tests, the position signal was sampled at 71 Hz. After flight trials it was decided on using a Butterworth filter with time constant 0.08 (which meant a cut-off frequency

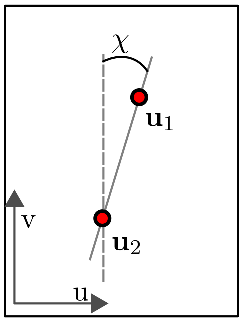


Figure 3-7: Projection of two heading reference LEDs $\mathbf{u}_1$ and $\mathbf{u}_2$ on the on-board camera. The LEDs are placed on the ground, and the camera looks downwards. The heading χ can be estimated from the horizontal u and vertical v pixel coordinates.

of 2.84 Hz) as the right mix between small latency and smoothness. A time constant of 0.05 (1.7 Hz) gives a smoother signal, but the time delay causes the system to be less stable. A value of 0.12 (4.2 Hz) was deemed to little filtering based on the observed higher frequency oscillations during test flights with the quad copter.

3-4 Heading

The Delfly needs a heading reference to know in which direction it has to fly. An on-board camera combined with an IR LED placed in the middle of the wind tunnel provided this heading reference.

In slow forward flight, the horizontal pixel coordinate u of the reference LED is a measure of the heading. This can be transformed to the heading χ in degrees by

$$\chi = s_u(u - c_u) \tag{3-5}$$

where s_u is the angle one horizontal pixel represents (also called the *pixel pitch*), c_u the center pixel that defines the mid point of the field of view in pixels.

During forward flight, where the Delfly is at a smaller pitch attitude, the on-board camera is no longer looking forward and is therefore unable to see the reference LED positioned in the middle of the wind tunnel. The heading was therefore calculated in a different way.

When the Delfly is flying at higher velocities, the camera is pointing down at an angle of about 30 degrees from the vertical. By placing two LEDs on the ground about a meter in front of the Delfly, the heading can be estimated from the difference in horizontal position. This is illustrated in Figure 3-7.

The angle defined by χ is the heading of the Delfly, and can be expressed by

$$\chi = \tan^{-1}(\tfrac{u_1 - u_2}{v_1 - v_2}) \tag{3-6}$$

where u is the horizontal pixel coordinate and v the vertical pixel coordinate of their respective pixel. Small-angle approximation is applied to save on calculation time on the on-board processor, so Equation 3-6 becomes

$$\chi = \tfrac{u_1 - u_2}{v_1 - v_2} \tag{3-7}$$

3-5 Angular rates

The gyro was low pass filtered on-board, using a digital low pass filter. Such a filter is described by

$$y_i = \alpha x_i + (1 - \alpha) y_{i-1}$$

where y_i is the filtered measurement at time i, and x_i the new measurement. α is the smoothing parameter. This can be rewritten as

$$y_i - y_{i-1} = \delta y_i = \alpha(x_i - y_{i-1})$$

So the change in value is equal to the difference between the current value and the new measurement times the smoothing parameter.

A smoothing parameters α of $\frac{1}{8}$ was selected. With a sampling rate of 100 Hz the equivalent time constant for this smoothing parameter is equal to

$$RC = \frac{1}{f_s}\frac{1-\alpha}{\alpha} = \frac{1}{100}7 = 0.07\text{seconds}$$

where f_s is the sampling frequency. This equals a cut-off frequency f_c of

$$f_c = \frac{1}{2\pi RC} \approx 44\text{Hz}$$

This cut-off frequency is low enough to reduce aliasing when the gyro data is send down at 50 Hz, and high enough to leave important content such as the frequencies around the flapping action (around 12 Hz) intact.

To transform the gyro measurements to the Delfly body axes, the following rotation matrix can be applied

$$R_C^D = \begin{bmatrix} \cos 17^\circ & 0 & -\sin 17^\circ \\ 0 & 1 & 0 \\ \sin 17^\circ & 0 & \cos 17^\circ \end{bmatrix} \tag{3-8}$$

Scaling factor

For the analog IXZ-500 the factor between the output of the gyro and the physical meaning of this value, was reportedly 9.1 mVdeg$^{-1}\,s^{-1}$ at the X4.5OUT and Z4.5OUT pins. For a range of $\pm 110°s^{-1}$ and a reference voltage of $V_{REF} = 1.35V$ this meant a voltage range of 0.35-2.35 volts. The ADC converter on the CPU scales 0 to 3.3V to 8 bit. This is scaled with three bits (factor of 8) to reduce round off errors introduced by the fixed-point calculations, increasing the gyro reading to 11 bits. The total scale factor is therefore

$$\frac{3.3}{2^{11}9.1 \cdot 10^{-3}} = 0.18°s^{-1}unit^{-1}$$

The digital IMU-3000 gyro provides a 16 bit output but only the 8 most-significant-byte (MSB) were used. For a range of $\pm 250°$ and keeping into account the fixed-point scale factor three bits, the scale factor for the Z gyro output is

$$\frac{500}{2^{11}} = 0.24°s^{-1}unit^{-1}$$

These scaling factors can be used to translate the output of the gyro to physical interpretable units.

Chapter 4

Calibration

In this chapter the various methods used to calibrate the hardware are explained. The calibration of the on-board camera and gyro's is explained. The calibration of the two tracking cameras is performed using Zhangs's method and a calibration technique is proposed for calibrating the tracking system with the wind tunnel using the TRIAD algorithm.

4-1 Tracking system

In order to employ the Linear-LS algorithm as described in the Section 3-2, we need to determine the intrinsic parameters of both cameras used in the tracking system, as well as their relative orientation and position, which are called the extrinsic parameters.

4-1-1 Intrinsic parameters

To obtain the intrinsic parameters, a program was written that collects many data points from looking at a calibration board. This calibration board consists of four infrared LED's at a known distance from each other, see Figure 4-1. From the observation of these LED's, Zhang's method for camera calibration (Zhang, 1999) was applied to calculate the intrinsic parameters of each camera, using the tool developed by Microsoft (Zhang, 2001).

This tool can also estimate the distortion parameters, which are used to first correct the camera projection coordinates for barrel and pincushion distortion. First it was tried to obtain all the intrinsic parameters and the distortion parameters in one go. Starting with a sample size of 80, it soon proved that there was still significant spread in subsequent calibrations. Therefore, the sample size was increased to 200, but still there was still significant spread between subsequent calibrations. This is because the Wiimotes can only track 4 points, and therefore the calibration pattern only contains four points. This is not enough to also estimate the distortion parameters.

Therefore a different approach was taken. First the severity of the distortion would be estimated by mounting each Wiimote on a CNC milling machine, looking downwards. A IR

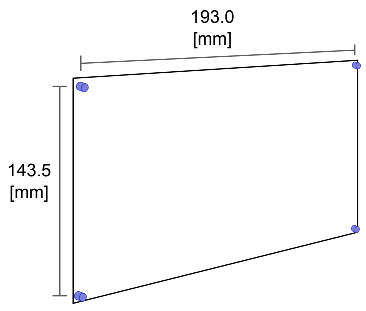


Figure 4-1: Calibration board: four infrared LEDS in a rectangular formation, used for calibration of the intrinsic and extrinsic parameters of the cameras.

LED was placed on the table below the mount and the camera was made to move in a series of straight lines. The image of what the camera captured were then visually inspected to look at amount of distortion in the images (the plots are included in Appendix A-1). This process was repeated for camera 2. From this experiment it turned out that there is almost no distortion present in both cameras. It seems that the distortion is already corrected on the chip of the camera.

Knowing that distortion would not be an issue, we could ignore these parameters and it was possible to successfully use Zhang's method to estimate the intrinsic parameters. Averaging over five runs, the resulting parameters are:

	f_x	f_y	c_x	c_y
blue wiimote	1.262	1.701	0.475	0.565
black wiimote	1.313	1.749	0.508	0.500

resulting in the following intrinsic matrices

$$N_1 = \begin{bmatrix} 1.262 & 0 & 0.475 \\ 0 & 1.701 & 0.565 \\ 0 & 0 & 1 \end{bmatrix} \qquad N_2 = \begin{bmatrix} 1.313 & 0 & 0.508 \\ 0 & 1.749 & 0.500 \\ 0 & 0 & 1 \end{bmatrix} \tag{4-1}$$

4-1-2 Extrinsic parameters

In the Linear-LS method for triangulation, the relative position and orientation of the stereoscopic camera pair is used. It describes the relationship between the two cameras, so that a point in one camera coordinate frame can be described as a point in the other. The combination of the rotation matrix and the translation vector are called the extrinsic parameters.

OpenCV's *stereoCalibration2* routine was used calibrate the extrinsic parameters of the two cameras. The function calculates the relative position and rotation between two camera's based on multiple samples of the cameras looking at the calibration board. This function was incorporated in a separate executable called `WiiMoteStereoCalib.exe`. It connects to the two Wiimotes, takes 40 samples, and saves the calculated rotation and translation matrices in a text file `CalibrationParametersStereo.txt`. The calibration only takes about a minute to complete, and should be performed again every time the cameras are moved.

The output of the stereo calibration routine are rotation matrix R and translation vector $\mathbf{T}$, shown in Equation 4-2

$$R = \begin{bmatrix} 0.0375 & -0.0043 & 0.9992 \\ 0.0109 & 0.9999 & 0.0039 \\ -0.9992 & 0.0108 & 0.0375 \end{bmatrix} \qquad \mathbf{T} = \begin{bmatrix} -1.4624 \\ 0.0414 \\ 1.4246 \end{bmatrix} \tag{4-2}$$

4-1-3 Inertial calibration

As explained in the section on position estimation, the result of the Linear-LS method is the position of a point relative to camera 1 which is subsequently expressed in the uncalibrated wind tunnel reference frame $\mathcal{F}_{\hat{w}}$ by the rotation matrix and translation vector defined in Equation (3-4).

In order to exactly align $\mathcal{F}_{\hat{w}}$ with $\mathcal{F}_w$, the correct translation and rotation matrix needs to be found that translate a point from the former to the latter reference frame. This is called the *inertial calibration*, because it aligns the uncalibrated, in essence arbitrary, frame fixed to the tracking system with the inertially fixed wind tunnel, whos reference frame is defined by a fixed geometry.

To this end, the calibration board from Figure 4-1 was placed on a support. One LED of the board was placed at the point that should become the origin of the new coordinate system. The board was aligned with the wind direction by looking over the board towards a reference LED in the middle of the tunnel. Furthermore, it was leveled using a spirit level. A reading with the tracking system was taken and the TRIAD algorithm was used to acquire the correction rotation matrix and translation vector.

The TRIAD algorithm determines the three-axis attitude from two vector observations. In this case, the vectors are the two axis of the calibration board, spanned by 3 of the 4 LEDs. Because the calibration board is positioned such that it defines the wind tunnel reference frame, we can use the resulting rotation matrix and translation vector to translate correct for the misalignment of $\mathcal{F}_{\hat{w}}$ with $\mathcal{F}_w$. This way, we don't need to measure the exact location and rotation of the cameras, but we can define the alignment of the wind tunnel by aligning the calibration board, and make use of the measurement precision of the tracking system to accurately estimate the calibration rotation matrix and translation vector.

The algorithm is applied as follows: The coordinates in $\mathcal{F}_{\hat{w}}$ of the four LEDs of the calibration board is calculated using the position determination algorithm as described in Section 3-2. The LED with the most negative x and y position is selected as the origin of $\mathcal{F}_w$, and its coordinates form the translation vector $\mathbf{T}_w$.

Then the TRIAD algorithm (Shuster & Oh, 1981) is applied to precisely establish the true orientation of the calibration board in $\mathcal{F}_{\hat{w}}$. The inverse of this gives us the rotation matrix to translate a point from $\mathcal{F}_{\hat{w}}$ to $\mathcal{F}_w$.

TRIAD algorithm

Given two nonparallel reference unit vectors, $\mathbf{V}_1$ and $\mathbf{V}_2$, and the corresponding two observation unit vectors $\mathbf{W}_1$ and $\mathbf{W}_2$, we want to find an orthogonal matrix A that satisfies

$$A\mathbf{V}_1 = \mathbf{W}_1 \qquad A\mathbf{V}_2 = \mathbf{W}_2 \tag{4-3}$$

Matrix A is overdetermined, and therefore we construct a manifestly orthonormal reference and observation vector by writing

$$\begin{aligned}
\mathbf{r}_1 &= \mathbf{V}_1 \\
\mathbf{r}_2 &= (\mathbf{V}_1 \times \mathbf{V}_2)/\left|\mathbf{V}_1 \times \mathbf{V}_2\right| \qquad (4\text{-}4)\\
\mathbf{r}_3 &= (\mathbf{V}_1 \times (\mathbf{V}_1 \times \mathbf{V}_2))/\left|\mathbf{V}_1 \times \mathbf{V}_2\right| \\
\mathbf{s}_1 &= \mathbf{W}_1 \\
\mathbf{s}_2 &= (\mathbf{W}_1 \times \mathbf{W}_2)/\left|\mathbf{W}_1 \times \mathbf{W}_2\right| \qquad (4\text{-}5)\\
\mathbf{s}_3 &= (\mathbf{W}_1 \times (\mathbf{W}_1 \times \mathbf{W}_2))/\left|\mathbf{W}_1 \times \mathbf{W}_2\right|
\end{aligned}$$

There exists a unique orthogonal matrix A which satisfies

$$A\mathbf{r}_i = \mathbf{s}_i \qquad (i = 1, 2, 3) \tag{4-6}$$

which is given by

$$A = \sum_{i=1}^{3} \mathbf{s}_i \mathbf{r}_i^T \tag{4-7}$$

where the superscript T denotes the matrix transpose. This is identical to

$$A = M_{obs} M_{ref}^T \tag{4-8}$$

with

$$M_{ref} = \left[\mathbf{r}_1 \vdots \mathbf{r}_2 \vdots \mathbf{r}_3\right] \qquad M_{obs} = \left[\mathbf{s}_1 \vdots \mathbf{s}_2 \vdots \mathbf{s}_3\right] \tag{4-9}$$

where the right members of the equation are 3×3 matrices labeled according to their column vectors.

Equation 4-7 and 4-8 both define the TRIAD solution to the three axis attitude determination problem, written in two different ways.

The rotation matrix that corrects for the misalignment of the uncalibrated reference frame $\mathcal{F}_{\hat{w}}$ is equal to

$$R_{\hat{w}}^{w} = A^{-1} \tag{4-10}$$

Any point that is measured in $\mathcal{F}_{\hat{w}}$ can be expressed in $\mathcal{F}_w$ by first adding $T_{\hat{w}}$ and then multiplying by $R_{\hat{w}}^{w}$

$$\mathbf{x}_w = R_{\hat{w}}^{w}(\mathbf{x}_{\hat{w}} + \mathbf{T}_{\hat{w}})$$

The result of the wind tunnel calibration is shown in Equation 4-11

$$R_{\hat{w}}^{w} = \begin{bmatrix} 0.9994 & 0.0229 & -0.0240 \\ -0.0227 & 0.9997 & 0.0079 \\ 0.0242 & -0.0073 & 0.9997 \end{bmatrix} \quad \mathbf{T}_{\hat{\mathbf{w}}} = \begin{bmatrix} -0.0365 \\ -0.0492 \\ 0.0320 \end{bmatrix} \tag{4-11}$$

4-2 Delfly on-board camera

In order to apply the attitude estimation as explained in Section C the angle measurements of the on-board camera were validated.

This was done by placing the camera at a known position and recording measurements of a LED at positioned at regular intervals along a line in front of the camera. From this the relation between the true angle and the measured pixel coordinate can be established. This setup is illustrated in Figure 4-2.

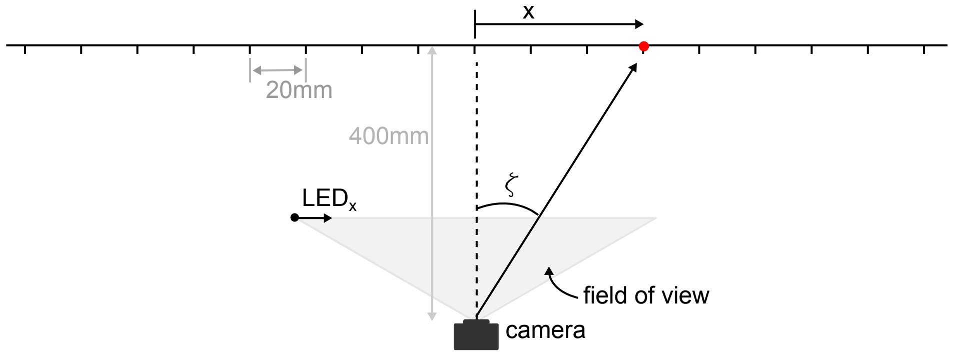


Figure 4-2: Top view: geometry of the calibration setup for the on-board camera. The red dot indicates the position of the LED presented to the camera at an angle ζ.

If we consider the vertical case, the linear relationship between the measured vertical pixel coordinate v and the actual geometric angle ζ can be expressed by

$$\zeta = s_v(v - c_v) \tag{4-12}$$

where s_v is the angle one vertical pixel represents (also called the *pixel pitch*), c_v the center pixel that defines the mid point of the field of view in pixels.

Seventeen different measurements were done over the whole field of view, see Figures 4-3. Normally a separation of 20 mm was adhered, but at the limits of the camera range the outer most points, where the camera would still just see the LED, were taken. Then, a minimum-least-square solution was calculated for the parameters s and c from Equation 4-12, resulting in a value of $c_v = 508$ pixels and $s_v = 0.0443\ ^\circ/pixel$. The camera was found to show very linear behavior, with an R^2 value of 0.9992.

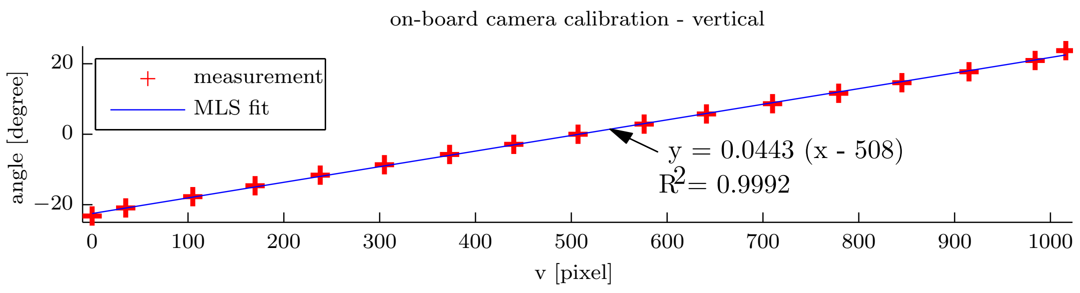


Figure 4-3: Measurements of the on-board camera calibration for vertical direction, and the results of the linear least square fit.

The same measurements were done for the horizontal direction. The results of this are shown in Figure 4-4. The resulting linear least square solution is

$$\zeta = s_u(u - c_u) = 0.0419(u - 385) \tag{4-13}$$

where the subscript u now indicates the parameters for horizontal direction, and u is the camera coordinate in horizontal direction.

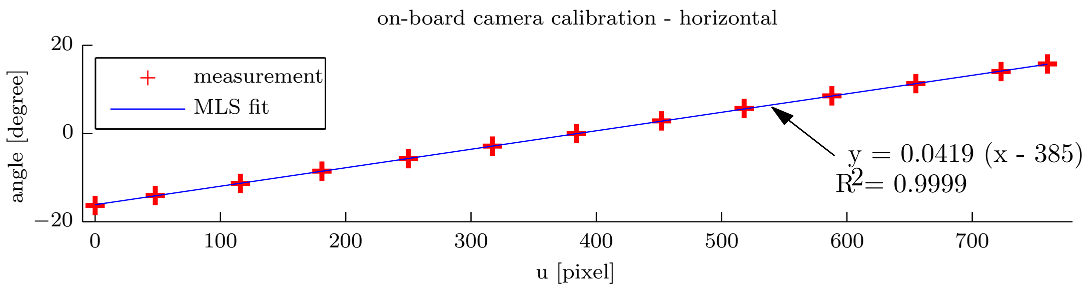


Figure 4-4: Measurements of the on-board camera calibration for horizontal direction, and the results of the minimum least square fit.

4-3 Gyroscopes

The calibration of the gyroscopes was performed before each flight to make sure the bias was removed. This was done by putting the Delfly on a fixed surface such as the T-shaped mast on the platform in the wind tunnel. The Delfly was then ordered to start the gyro calibration by sending a gyro configuration message from the control panel. The calibration value is obtained taking a heavily low passing reading of the gyro signal, after waiting two and a half seconds for the low-pass filter to stabilize.

Chapter 5

Controller

The task at hand is disturbance rejection, where the goal is to maintain a position as close to the center of the wind tunnel as possible. For this task we can control the change of pitch (elevator), change of yaw (rudder) and the total amount of thrust (throttle).

The lateral and longitudinal dynamics are considered to be separate, so two controllers, one for lateral and one for longitudinal control, are proposed. The controllers are based on the state information available as described previously. Aside from the position and velocity information, also a IR reference LED is presented to the on-board camera.

At first a more complex controller algorithm was designed, using multi loop feedback control and even energy control methods were considered, but this was later dropped for complexity reasons. A much simpler algorithm was designed, which uses the reference LED only for heading control.

5-1 Lateral control

The Delfly has only three actuators: elevator, rudder and the speed of the motor. It has no aileron or substitute for the conventional aileron, which means the Delfly is unable to directly control roll. The intrinsic stability (i.e. low center of gravity) keeps the Delfly up right, and in order to make a turn it will have to rely on its rudder.

5-1-1 Slow forward flight

Consider the case as depicted in 5-1. The Delfly has a lateral off-set of y and a heading χ. We could construct a multi-loop controller, where the lateral off-set would provide a heading reference, calculate the heading error and steer in the opposite way.

Instead, it is much simpler to try to minimize the error between the direction the Delfly is facing and the direction of the LED, angle μ. Over time, this will automatically correct for

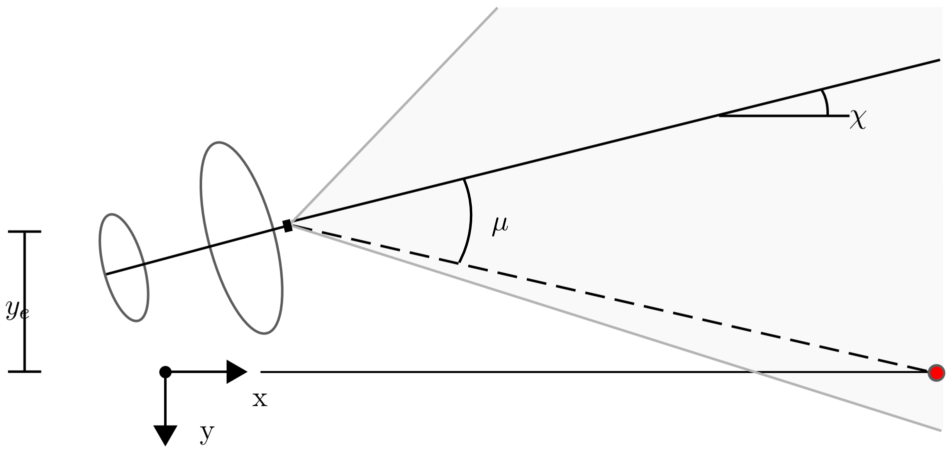


Figure 5-1: Schematic top-view of an lateral error. The red dot indicates the location of the infra red reference LED.

the lateral position off-set. This is illustrated in Figure 5-1. The controller would have the form

$$\delta_r = K_\mu \mu \tag{5-1}$$

To reduce possible oscillations we can include derivative control. A gyro is positioned in the vertical axis of the camera, X_C. It measures the rate around the camera X-axis, and we will therefore call the rate it measures p_C. We can use this information to create a derivative controller, providing damping to reduce possible oscillations, by feeding back p_C to the rudder.

$$\delta_r = K_p p_C \tag{5-2}$$

where K_p is the so-called roll gyro gain. Gain K_p is expected to have a relatively low value, because the rudder does not directly control the roll but the heading. The effect of the rudder is not direct on the heading, so a too high K_p gain can destabilize the Delfly.

The original design did not include a yaw damper, and the autopilot PCB did not include a yaw gyro, but during manual flight testing the Delfly showed severe oscillations around the vertical body axis Z_D, so a yaw gyro was fitted. This gave the possibility to include a yaw damper by feeding back the yaw rate r. The gyro was fitted orthogonally to the other two gyros, which meant its axis was aligned with the Z axis of the on-board camera, Z_C. Because the X_C and Z_C axes are rotated 17° around the Y_D axis, the true yaw rate r_D around the Z_D Delfly body axis is given by (from Equation 3-8(:

$$r_D = r_C \cos 17^\circ + p_C \sin 17^\circ$$

where r_C and p_C are the rates measured by the gyros aligned with the Z_C and X_C axes respectively.

$$\delta_r = K_r r_D \tag{5-3}$$

where K_r is the yaw damper gain.

We want to have the Delfly stay at exactly the center of the wind tunnel, so to reduce a steady state error in the y direction an integrator term can be included. In effect this will provide automatic trimming of the rudder.

$$\delta_r = K_y^I \int (y - y_{ref}) \tag{5-4}$$

where K_y^I is the integrator gain.

The total controller for slow forward flight becomes, dropping out y_{ref} which is zero for the middle of the wind tunnel

$$\delta_r = K_\mu \mu + K_p p_C + K_r r_D + K_y^I \int y + K_k \tag{5-5}$$

where K_k is a value trim to be set manually when necessary. The gains will be manually tuned during testing in the wind tunnel.

5-1-2 Forward flight

When the Delfly is flying at higher velocities, the controller remains largely the same. One important difference though was that the reference LED right in front of the Delfly is out of view, because the camera is looking downwards. Therefore, instead of steering towards the LED, the heading was used to keep the right direction. The first part of the total controller shown in Equation 5-5 becomes

$$\delta_r = K_\chi \chi \tag{5-6}$$

Because minimizing the heading will not make the Delfly minimize the lateral off-set y_e, this state needs to be fed-back also. Including the yaw damper, these three gains in effect work as three loop controller, where the outer-loop provides a heading reference based on the lateral off-set, the next loop generates a yaw rate command, and the inner loop controls the rudder to reduce the yaw rate error. This three loop control scheme is shown in Figure 5-2.

The control law that results from this three loop controller can be expressed by

$$\begin{aligned}\delta_r &= K_r r_e \\ &= K_r(K_\chi \chi_e - r) \\ &= K_r(K_\chi(K_y y_e - \chi) - r) \\ &= K_r(K_\chi(K_y(y_{ref} - y) - \chi) - r)\end{aligned} \tag{5-7}$$

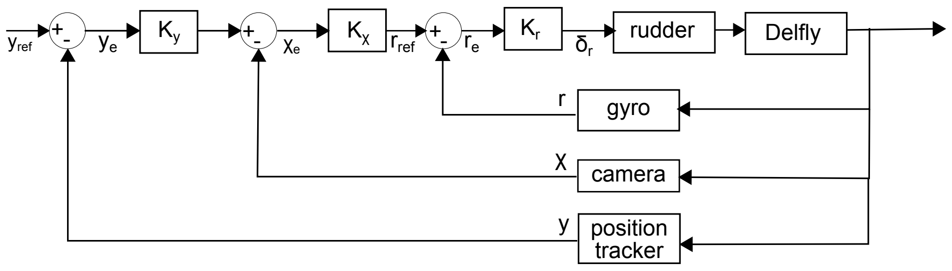


Figure 5-2: Triple loop lateral position controller.

where the roll rate r represents the rate around the Delfly body axis Z_D, i.e. $r = r_D$. Using $y_{ref} = 0$ and rewriting

$$\delta_r = -K_r K_\chi K_y y - K_r K_\chi \chi - K_r r \tag{5-8}$$

if we replace the product of multiple gains by a single gain and including the minus sign in this new gain, we can write

$$\delta_r = K'_y y + K'_\chi \chi + K'_r r \tag{5-9}$$

where K'_y, K'_χ and K'_r are the new gains. Dropping the accent, we see that the three loop control is nothing else as the addition of position, heading and yaw rate feedback.

Including an integrator term to allow the steady state error to be removed, the total controller becomes

$$\delta_r = K_y y + K_\chi \chi + K_r r + K_y^I \int y + K_k \tag{5-10}$$

With K_k a trim value that can be set manually.

The differences with the lateral controller for the slow forward flight are that it uses the calculated heading χ instead of the relative angle to the LED μ, the roll term has dropped out and the position y is fed back directly. The roll feedback is not expected to be needed, because there is no actuator to directly control the roll.

5-2 Longitudinal control

When flying at low speeds, the thrust vector is pointing almost straight down and therefore mostly controls the vertical velocity, whereas at higher velocity it controls the horizontal velocity more. The elevator acts exactly the opposite way. So when the Delfly is flying too low at low velocities, the motor should correct for this and the elevator should be used to control the horizontal velocity. The opposite should be done at higher velocities: the motor controls the horizontal velocity and the elevators make the Delfly fly up or down.

5-2-1 Slow forward flight

When flying slowly forward the pitch angle is around 70°. At this angle a small part of the thrust is used to overcome drag, but the larger part is counteracting the force of gravity. Therefore for now we assume that the motor controls the force balance in vertical direction, ignoring the effect it has on the horizontal force balance. Furthermore, any change in thrust setting causes the Delfly to accelerate or decelerate vertically, reaching after a while a new equilibrium velocity. It is assumed that for small changes in thrust settings this equilibrium is reach quickly, and in effect the motor directly controls the vertical velocity.

The elevator causes a change in pitch rate, which in effect changes the pitch. The thrust vector is tilted, and an extra force accelerates the Delfly forwards or backwards. In analogy with the thrust and the vertical acceleration, we assume that a certain elevator setting will rather quickly stabilize at a certain horizontal velocity.

Observations from flight tests have shown that the above are indeed valid assumptions. If we question a Delfly pilot and ask how he controls it, it turns out that he indeed controls the vertical velocity by making very small adjustments to the throttle. He would also indicate that normally the elevator is not used at all, and the Delfly is flown at a certain elevator trim setting for a certain forward velocity, set once during flight. This also supports the assumption that the elevator can be regarded as controlling the horizontal velocity at slow forward flight/high angle of attack.

From the above discussion we assume a completely decoupled system, where the elevator controls the horizontal velocity and the motor controls the vertical velocity. If we want to control the position, we can arrive at the very basic control laws:

For the vertical position z

$$\delta_{th} = K_z(z - z_{ref}) \tag{5-11}$$

and the horizontal position x

$$\delta_e = K_x(x - x_{ref}) \tag{5-12}$$

where δ_{th} and δ_e are resp. the thrust and elevator input, K_z and K_x gains, z and x the current horizontal and vertical position and z_{ref}, x_{ref} the reference position we want to achieve.

With the above controller, there is no compensation for a steady state error. If the actuators are not perfectly trimmed, a zero error will never be achieved. Also, during flight the battery will drain and the electric potential will drop, as shown in Figure 5-3. This means that at the same motor setting, less power can be generated and therefore there is less thrust.

This can be solved by including a feed forward term (in combination with a look-up table of the voltage output over time) in the motor controller, or by measuring the voltage over the battery and compensating accordingly. Another way would be to measure the actual RPM of the motor, and add a feedback loop to keep the RPM at the required level.

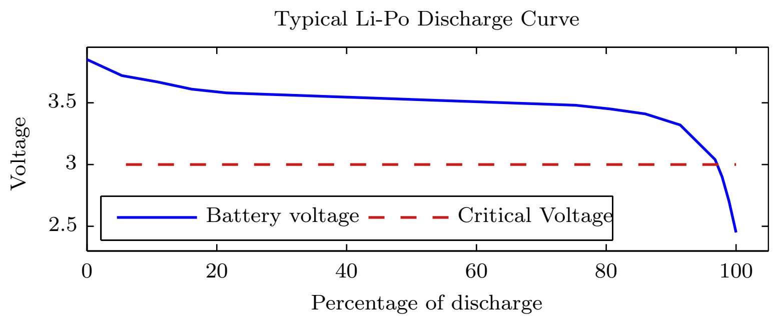


Figure 5-3: Result of the integrator term added to the controller. Over time the steady state error was resolved without any manual trimming.

The first option would require knowledge of how the battery drains over time, and is a function of load, making it very difficult to generate a look-up table. The last two options require extra state information, the battery output voltage in the former case, or the RPM of the engine in the latter, but neither of these two two measurements were available.

As a solution we can can expand the controller with an integrator term. This makes sure the actuator is increased until all steady state error has disappeared. It is of the form

$$\delta_{th} = K_z^I \int (z - z_{ref}) \tag{5-13}$$

$$\delta_e = K_x^I \int (x - x_{ref}) \tag{5-14}$$

where K_z^I and K_x^I are the gains associated with the integral terms.

A pitch damper was also employed, described by

$$\delta_e = K_q q$$

where K_q is the pitch damper gain and q the pitch rate as measured by the pitch gyro.

The total control law for the throttle setting becomes then (again dropping out the reference values, which are zero at all times)

$$\delta_{th} = K_z z + K_z^I \int z + K_n \tag{5-15}$$

and for the elevator input

$$\delta_e = K_x x + K_x^I \int x + K_q q + K_m \tag{5-16}$$

where two extra terms, K_n and K_m, have been included for manual trimming of the throttle and the elevator respectively.

5-2-2 Forward flight

When the Delfly flies at higher velocities, the pitch angle decreases to about 30° for velocities around 2 m/s and upwards. The Delfly behaves like a conventional fixed wing configuration, and the role of the actuators changes from the situation outlined above.

The input and output channels are changed so that the elevator is now controlling the altitude and the motor setting controls the forward velocity. The proposed controller therefore takes a similar form as above, but the actuators are interchanged. Also, because of the higher velocity, and hence faster dynamics, for the longitudinal control also the velocity information was used in the controller. This will be able to provide derivative control action on the position signal.

For the forward flight the vertical position z is controlled by the elevator

$$\delta_e = K_z(z - z_{ref})$$

and the horizontal position x is controlled by the thrust setting

$$\delta_{th} = K_x(x - x_{ref})$$

The term for the velocity feedback in the horizontal and vertical direction is

$$\delta_e = K_{v_z} v_z \delta_{th} \qquad = K_{v_x} v_x \tag{5-17}$$

In order to reduce the steady state error, we can add a small integrator term.

$$\delta_e = K_z^I \int (z - z_{ref}) \tag{5-18}$$

$$\delta_{th} = K_x^I \int (x - x_{ref}) \tag{5-19}$$

The pitch damper does not change

$$\delta_e = K_q q$$

where K_q is the pitch damper gain and q the pitch rate as measured by the pitch gyro.

The total control law for the throttle setting becomes then

$$\delta_{th} = K_x(x) + K_{v_x} v_x + K_x^I \int (x) + K_n \tag{5-20}$$

and for the elevator input

$$\delta_e = K_z(z) + K_{v_z} v_z + K_z^I \int (z) + K_q q + K_m \tag{5-21}$$

including the manual trim settings K_n and K_m, and ignoring the zero-values reference values x_{ref} and z_{ref}. These control laws are almost exactly the same as Eq. 5-15 and 5-16, only the axes on which control surface acts on are changed and a derivative term is added.

Chapter 6

Experiment Details

This chapter is about the experiments performed with the Delfly. The goal of this thesis was to show whether or not it is possible to fly the Delfly with high precision in the wind tunnel, and to this end the controller had to be tuned until the Delfly could fly at a predefined point with the highest accuracy possible. After this was achieved, some extra experiments were performed. These consisted of position step inputs in the vertical and longitudinal direction.

First, a description of the test environment, which is the OJF low-speed wind tunnel, is given. Secondly, the way the experiments were performed is shown. It will describe the method of testing and explain how the controller was tuned during the test flight by using the custom made Graphical User Interface (GUI).

6-1 Wind tunnel setup

The TU Delfts OJF was used for the wind tunnel experiments. It is a large open-jet tunnel, capable of generating wind speeds up to $30ms^{-1}$. This tunnel was chosen because of its large size, providing ample margin of error for the fragile Delfly.

The OJF has a movable platform that can be moved up and down. The platform was raised until it was level with the bottom of the tunnel exit. On this platform two aluminum beams were mounted, creating a support from which the Delfly could be suspended with a thin wire. By catching the Delfly with no damage when things went wrong, the wire proved to be a true life saver, allowing for continuous testing days on end without a single fatal crash. Only once the wire got caught in the gearing, but generally the stiffness of the wire made sure this didn't happened. At low velocities ($< 1ms^{-1}$) the wire was of very little influence, hanging slack when the Delfly was flying, exerting maybe a very little pitch-down moment. At higher velocities though, the wire would provide enough drag that it would float upwards, influencing the pitch of Delfly noticeably.

Although the support was in the wind stream, velocities were so low that the support did not create a very large turbulent area. Late in the experiments, at higher velocities (around

$3ms^{-1}$), it was noticed that the beams were influencing the wind stream too much. Therefore, the beams were removed and a long round iron bar was suspended from the top of the wind tunnel, extending half way the test area. From this the Delfly could be suspended and nothing would influence the free stream. Using this beam is therefore the preferred way to create a safety catch for the Delfly.

From the top to the bottom of the nozzle-exit a thin fish wire was stretched. A small infrared LED was suspended with a little tape and the LED's copper wiring was wound around the fish wire. The exact position of the LED, x_{led} and z_{led} (Figure 3-1) was noted.

The two ground cameras were placed in their wooden mount on the test platform, the camera's on the windward side, and clamped down by two F-clamps. A 1100 mm long aluminum X-bar was set up approximately in the middle of the platform, and the calibration board was placed on top, LED's facing down so all four LED's could be seen by both cameras. Using a spirit level it was made sure the board was level. With a tape measure one of the LED's of the calibration board was positioned exactly in the middle of the tunnel, and the board was rotated so the long side of the board was facing the reference LED. The calibration board now exactly defined the wind tunnel reference system, $\mathcal{O}_w$. The three principle axes are shown in Figure 3-1. Then the calibration program was run and the exact rotation matrix and translation vector of the camera's relative to the wind tunnel axes were obtained. Before removing the calibration board, the exact height of the calibration board, z_{calib} (Figure 3-1), was measured and written down.

6-2 Experiments

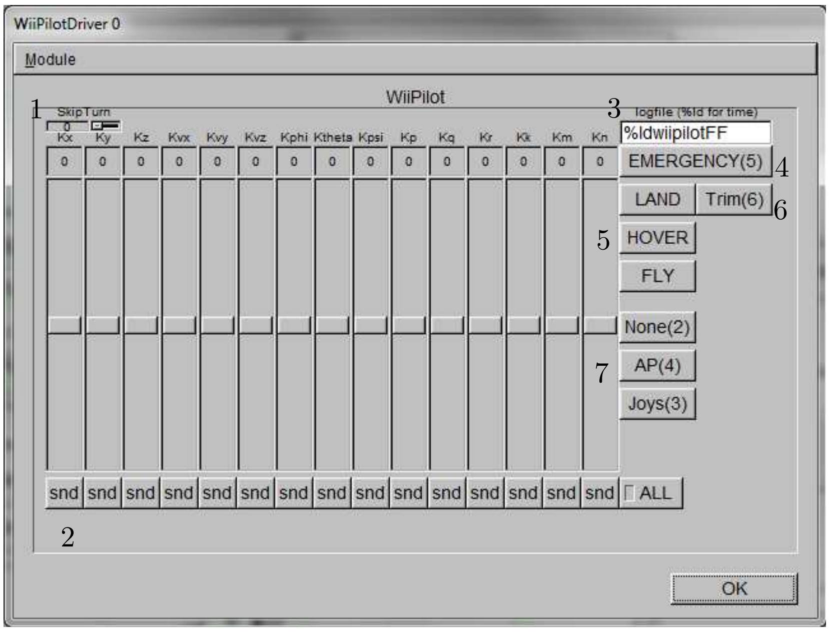
The experiments consist of three separate parts:

- Achieve autonomous flight at slow forward flight ($< 0.8ms^{-1}$)
- Achieve autonomous flight at forward flight ($> 2.0ms^{-1}$)
- Step inputs on the x- and z-position at slow forward flight

Before elaborating these three points, first a quick overview will be given of the interface that was used to control the Delfly and tune the gains in real-time.

GUI and Communication

A GUI for WiiPilotDriver was developed so that the gains could easily be changed during flight. It also featured buttons to switch modes on the Delfly, for example between manual joystick control and auto-pilot mode. The GUI is shown in Figure 6-2.



The following features were implemented:

1. **Skip turn** To fine tune the rate at which information is send up, the module could be made to run at a different rate than the main program. This would allow the tracking-system to still be polled at a high bandwidth for filtering, while at the same time unloading the communication with the Delfly. Because of the problems with the Bluetooth communication, the skip_turn was set to
2. **Gain settings** A total of 15 gains could be set by changing the value of the slider, and pressing *snd*. By pressing the *ALL* button, all the gains are send at once.
3. **Log file** The name of the log file can be entered here.
4. **Emergency** The Delfly has a emergency mode, where the engine will be shut-off immediately, and the Delfly will do nothing.
5. **FMS** The FMS, or Flight Management System, allows different modes. This was not used for the Delfly, but was implemented to allow the quad copter to lift-off and hover.
6. **Trim** This button initiated the gyro trim routine, so the gyro's would be zeroed. This needs to be performed every time the Delfly is restarted.
7. **Mode switch** Turns the Delfly in no mode (None), auto-pilot mode (AP) where the on-board controller would be in charge, and joystick mode (Joys) where manual control was engaged.

Several buttons have a number behind their name. They refer to the number of a button on the joystick. This allowed easy access during flight testing. Other short-cuts on the joystick were buttons to zero all integrator values, and buttons to order the Delfly to make a step input in x- or z-direction.

Data rate

The used Bluetooth interface was unable to perform up to its specifications, which meant the up link and down link rates had to be turned down a lot. Whenever too much data was pumped through the connection, the signal was delayed more and more. A buffer would fill up, and when sending too much position data for a longer period of time, it could take up to 15 seconds for the buffer to clear and the data to be sent.

After trying several different rates, it was settled on a 12.5 Hz up link and a 20 Hz down link rate. At these settings there were no buffer problems, and all messages arrived in time. For more details on the Bluetooth issues and latency measurements, the reader is referred to Appendix B-3.

6-2-1 Slow forward flight

The first time the Delfly was flown in the OJF the proportional gains were to be tuned to achieve stable flight. The process is outlined below.

The tunnel was switched on with the Delfly suspended by the safety wire. By keeping the wire in hand and slowly increasing thrust until the Delfly could sustain its own weight the trim thrust setting was estimated to be around 10, on a motor setting range from -127 to 127 (this value turned out to be too high during flight, and was subsequently tuned down to 1). This was done at a wind velocity between 0.5 and 0.6 m/s. The OJF is not made for such low velocities, so the wind velocity sometime changed a bit.

After trimming the motor input, K_μ was set to see whether or not the Delfly would be able to fly in the direction of the reference LED, and therefore keeping its y-position. At the same time position feedback for the x- and z-direction was added, i.e. gains K_x and K_z. The values of these gains were determined by trial and error. The gains were increased until unstable behavior was observed. This meant heavy oscillations for the heading direction, or steep climbs and descents in vertical direction. The gains were then decreased until stable flight was achieved.

In the end, the following values for the gains were established, see Table 6-1. All other gains were zero, which meant in the beginning the controller was reduced to a proportional controller only. These values are the values of the gains as used in the controller described in Section 5-2-1.

K_x	K_z	K_{mu}	K_n
49	2	-5	1

Table 6-1: Gains used in first autonomous flight

With just the proportional gains the Delfly was able to keep in approximately the center of the wind tunnel.

In order to eliminate the steady state error, the integrator term was added on all three position coordinates, x, y and z. A relatively low value was implemented. This meant the Delfly would reduce the steady state error slowly, but it also meant that overshoot and the possibility of it

K_x	K_z	K_x^I	K_y^I	K_z^I	K_{mu}
20	2	6	-6	6	-5

Table 6-2: Gains used in the slow forward flight controller.

destabilizing the system would be kept at a minimum. The complete controller now had the gain values as shown in Table 6-2.

Then several values for the gyro feedback were tried, in an effort to reduce some observed oscillations. This was done by trial and error, looking at the position signal and see whether or not it would show less oscillations at different gain values. All data was logged for post-processing purposes, although the communication problems meant that the down link was restricted to 20 Hz, which had its effect on the quality of the data, most notably on the gyro data where the flapping frequency of about 12.7 Hz could cause aliasing when sampled at 20 Hz.

6-2-2 Step inputs

Although staying in the middle of the wind tunnel was the original and principal goal of this thesis, it is by no means a thorough indication of the performance of the controller. Some step inputs on the longitudinal position signals (i.e. x- and z-direction) were performed, to see what quality of data could be extracted from the tracking system. This can give an indication of the suitability of the system for system identification purposes and to see what the qualitative dynamical properties are of the Delfly, especially the non-linearities. This can be used as a starting point for the design of more advanced, non-linear controllers.

The step in the position signal sent to the Delfly was generated at the press of a joystick button, changing the reference from -15 cm to +15 cm and visa versa in either the x or the z direction.

The exact same controller is used, with gains as shown in Table 6-1, although from the results of gyro feedback a yaw damper gain of $K_r = -40$ was included. This was thought to increase the stability in the lateral plane during the step maneuvers.

6-2-3 Forward flight

Due to circumstances, the wind tunnel was only available for one day for the forward flight testing, and of this day half was taken up by necessary repairs to the Delfly. There was not enough time to thoroughly investigate the performance of the proposed controller.

First the Delfly was tried to be flown manually in the wind tunnel at a velocities of $3ms^{-1}$. It was soon discovered that the normal configuration of the Delfly as used with the slow forward flight, was unable to fly faster than $0.8ms^{-1}$ forward. Some severe modifications needed to be made to allow it to fly faster than that. The center of gravity was way too far to the back, and needed to be shifted forward. This was done by putting a carbon rod under the nose and taping the battery to this extension. This way it was possible to put the center of gravity in between 30 and 40 percent of the chord length. In this configuration, the Delfly behaved as a conventional airplane and it was possible to fly it in forward flight at velocities of 2.5 to at least 3.5 ms^{-1}. The maximum speed of the Delfly has not been tested, but it was noticed that the Delfly needs a lot less thrust in forward flight. The throttle setting was down from around 10 in slow forward flight at $0.8ms^{-1}$ to about -50 for a velocity of 3.0 ms^{-1}.

The battery could not be shifted forward too much, as this would produce a too large destabilizing moment due to the large surface area in front of the center of gravity (c.g.). It caused the Delfly to become very unstable in yaw direction. But with the battery at about 6 cm in front of the motor gearing, it was possible to keep the Delfly in the 2 meter square area of the wind tunnel by manually controlling it, where only a yaw damper helped the pilot. It was then tried to replicate the performance of the pilot with the proposed forward flight controller as explained in Sections 5-1-2 and 5-2-2.

The lateral controller was capable of keeping the Delfly flying straight in to the wind with yaw gyro and heading feedback. The longitudinal controller was more troublesome. The safety tether seemed to be interfering with the Delfly. The drag on the wire pulled it backwards, causing a large pitch-up moment. After trying several combinations of gains, it seemed that the Delfly would be able to fly autonomously and at one point it was staying in the box for about half a minute, but it is not clear how much influence the wire had. All in all, the tests were quite unfruitful. There was unfortunately not enough time to thoroughly investigate the controller by looking at the video footage and data, and make improvements to the controller.

Chapter 7

Results

The results of the slow forward flight tracking task and the step inputs on the x- and z-position signal are presented here.

7-1 Slow forward flight

A plot of the position of the Delfly with only a proportional controller is shown in 7-1. The steady state error due to wrong trimming is clearly visible, and there is no integrator gain to remove this error, and the Delfly moves around a equilibrium different from zero. The maximum deviations from the equilibrium values are shown in Table 7-1, as well as the respective RMS error values.

7-1-1 Integrator

The integrator term in the control law resulted in a successful cancellation of the steady state error, as can be seen in Figure 7-2. It not only provides the trimming needed to reach the zero reference point, but it is also effective in canceling the effects of the draining battery providing less power over the flight duration.

direction	maximum deviation [cm]	RMS [cm]
x	2.0	1.0
y	6.3	2.7
z	4.3	1.3

Table 7-1: Maximum deviations from the equilibrium position and the RMS error values in all three dimensions.

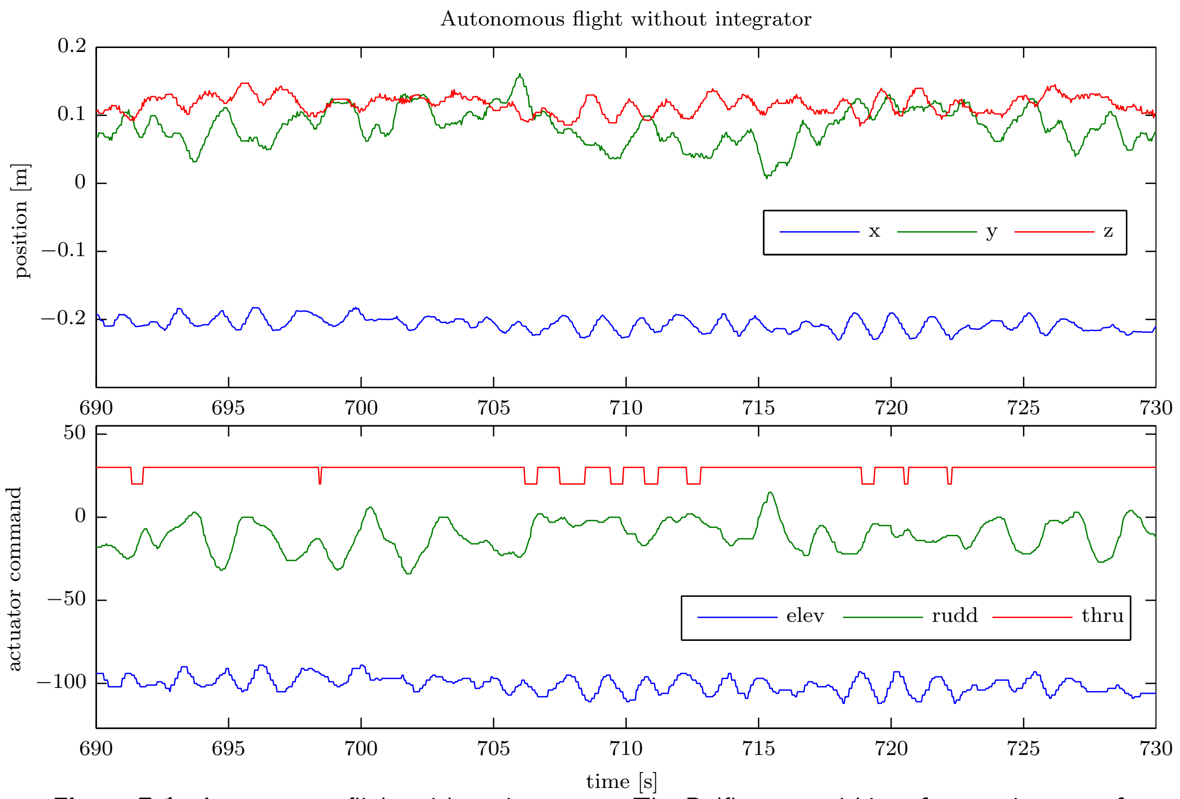


Figure 7-1: Autonomous flight without integrator. The Delfly stays within a few centimeters of its equilibirum point.

Cancellation of steady state error

It took about 20 seconds for the integrator to reach the required trim value, due to the relatively low gain on the integrating term. This was intentionally kept at a low value, so no overshoot would occur, and so that the integrator would be of little influence once the steady state command values were reached. Of course, the time it takes for the integrator to reach the proper value can be reduced by already providing a approximate trim value, based on previous flights.

Effect of voltage drop of battery

To see the effect of the integrator term on the altitude loss caused by the voltage drop of the battery, a longer flight was performed. It lasted about 14 minutes, almost one complete battery charge. The results is shown in Figure 7-3. The exact same trend as the voltage drop in Figure 5-3 is visible in the thrust command generated by the controller, but obviously it is inversed, as the motor setting needs to compensate for the reduction in potential. In the end the integrator can not keep up with the voltage drop, and the Delfly slowly sinks, at which point the engine was stopped to preserve the battery.

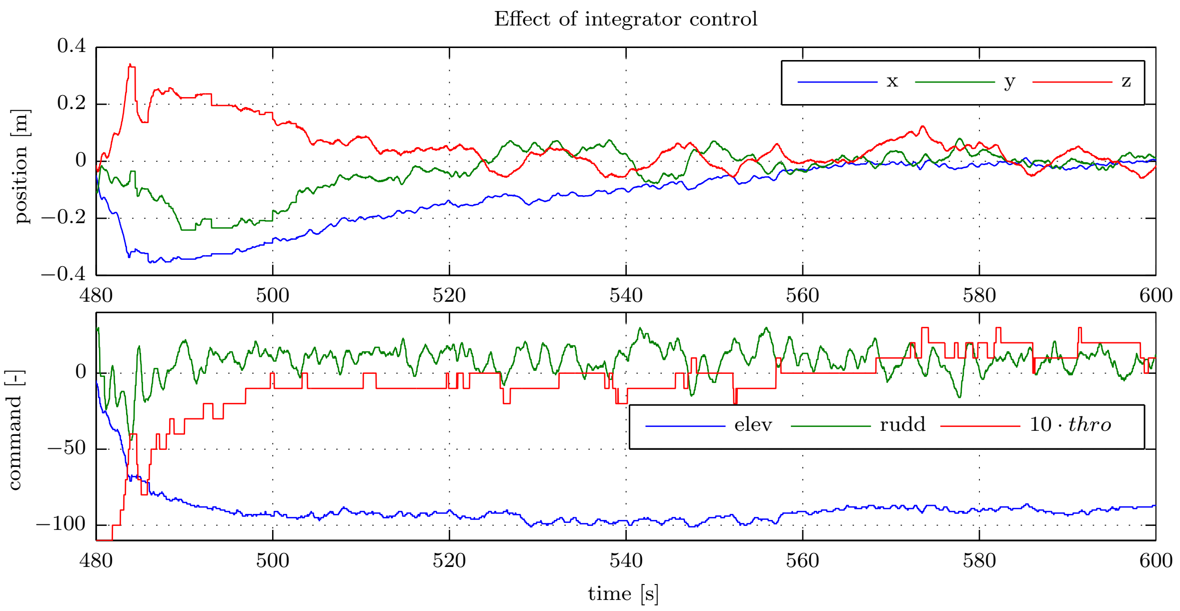


Figure 7-2: Result of the integrator term added to the controller. Over time the steady state error was resolved without any manual trimming.

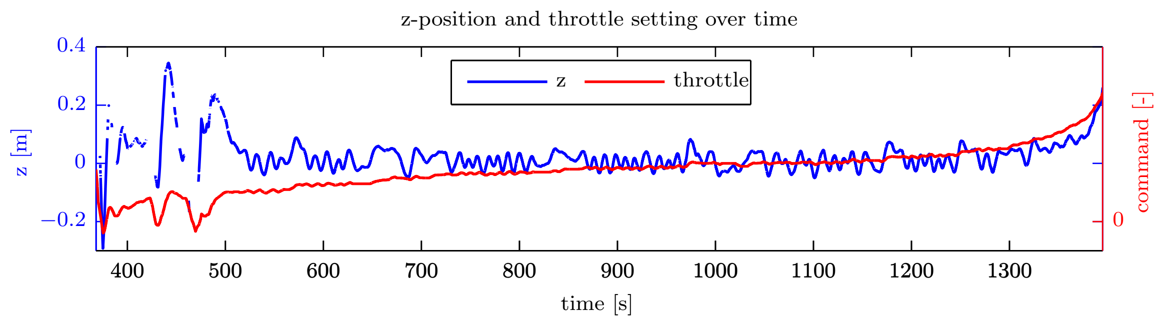


Figure 7-3: Vertical (z) position and thrust command over time. The integral action of the controller keeps the altitude constant, although a higher thrust setting is needed continuously because of the draining battery.

Motor controller imprecision

If we look a little bit closer to the position signal, we see a very a low frequency oscillation in the y and z direction, illustrated in Figure 7-4. It can be seen there that the throttle is alternating between two/three values. The Delfly descents every time the throttle is reduced one unit, and ascends when the throttle is increased (mind that the positive z axis is defined downwards). The motor controller is not precise enough to control the altitude.

Another effect due to the insufficient precision of the motor controller is illustrated in Figure 7-5. Up to $t = 470$ the motor controller is unable to reach a stable equilibrium. Because the battery drains over time, the same throttle setting will correspond to a diminishing amount of power. At $t > 470$ the power output of the battery was reduced to such an amount that the lift that is generated at $\delta_t = 13$ is exactly right to maintain an equilibrium, and the Delfly is hovering at a constant altitude. When the battery drains further, equilibrium is lost, and

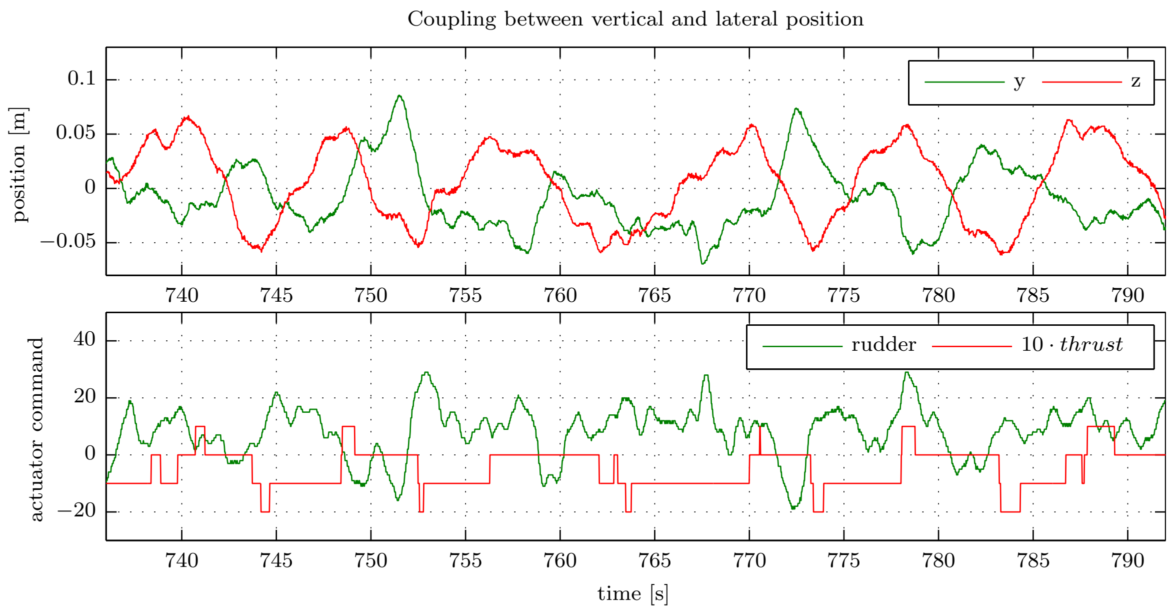


Figure 7-4: Close-up of a minute of autonomous flight. The throttle has been scaled $10\times$ for clarity. It is alternating between one or two values. The y- and z-position seem to be coupled: the Delfly moves up and right, or down and left, with a period of about 8 seconds.

direction	maximum deviation [cm]	RMS [cm]
x	1.6	0.82
y	4.3	1.8
z	1.5	0.95

Table 7-2: Maximum deviations from the equilibrium position and the RMS error values in all three dimensions, for a constant throttle setting.

at $t = 492$ the Delfly starts descending again. After a while the integrator has increased enough to increase the throttle setting one unit, and the Delfly rises, and starts oscillating again around the reference point.

The RMS error is significantly lower when throttle setting is constant than the values listed in Table 7-1. The maximum deviation and RMS values between $t = 470$ and $t = 490$ are shown in Table 7-2.

7-1-2 Gyro feedback

The effect of the gyros is investigated by looking at the RMS error of the position for different feedback gains for the roll, pitch and yaw damper. Unfortunately, after the flight testing it was discovered that the gyro correction matrix was not applied in the on-board controller to correct for the orientation of the roll and yaw gyro. This was especially concerning for the yaw damper, as it was meant to stabilize the yaw motion, and was now also measuring partly the roll rates. The roll damper on the other hand, was used as a heading damper, and was

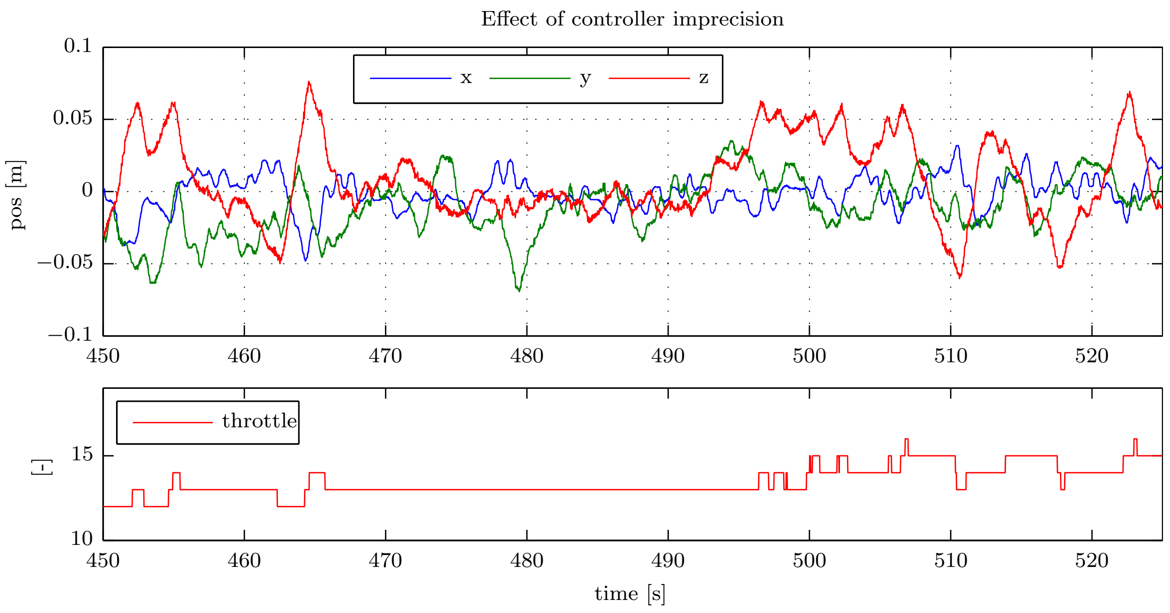


Figure 7-5: The oscillation disappears if the combined system of battery and controller is in an exact equilibrium. As soon as the battery drains a little more, the equilibrium cannot be reached due to the imprecision of the motor controller, and the Delfly starts oscillating.

therefore correctly oriented for measuring heading oscillations. All measurements of the gyro feedback were performed with the PI controller gains stated in the previous chapter.

Yaw damper

For the yaw damper, the effect on the tracking performance is shown in Figure 7-6. The RMS error in z-direction for $K_r = 30$ is a lot lower than the other gain settings. After more investigation, it turned out that the throttle setting was constant for the sample duration, causing a lot better performance due to the effect described earlier, where the imprecision of the motor controller would cause tracking performance degradation.

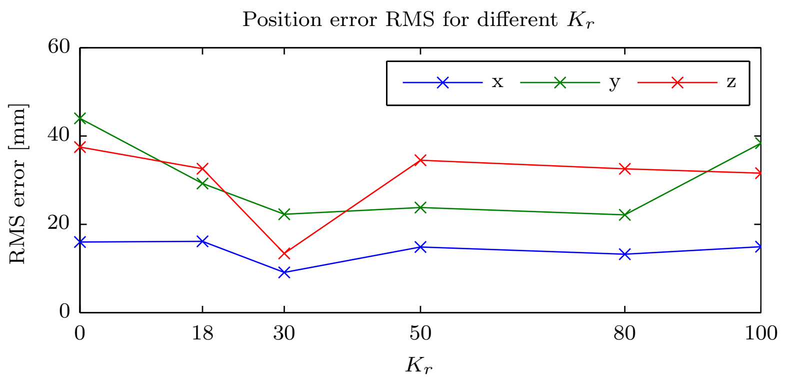


Figure 7-6: RMS error of the x,y and z position for different gain settings of the yaw damper. The sample duration was about 40 seconds for each setting.

Heading damper

The effect of the heading damper is shown in Figure 7-7. Only about 24 seconds per setting were investigated. It was observed that with only roll rate feedback, the Delfly would become unstable at a gain of $K_p = 15$. The inclusion of a K_r term would increase the point at which the heading damper would become unstable, but no data has been gathered on the combination of both gains.

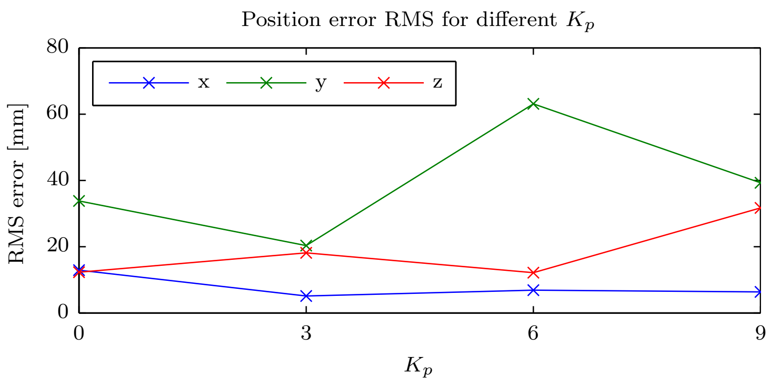


Figure 7-7: RMS error of the x,y and z position for different gain settings of the heading damper. The sample duration was about 24 seconds for each setting.

Pitch damper

The x position oscillates with an amplitude just under a centimeter, at a frequency of about 1.1 Hz. This frequency corresponds with an earlier observed eigen frequency of the Delfly during manual flight. For the effect of the pitch gyro feedback on the tracking performance, only a sample duration betwee 11 and 18 seconds were taken, and only for three gain values, but the result is shown in Figure 7-8.

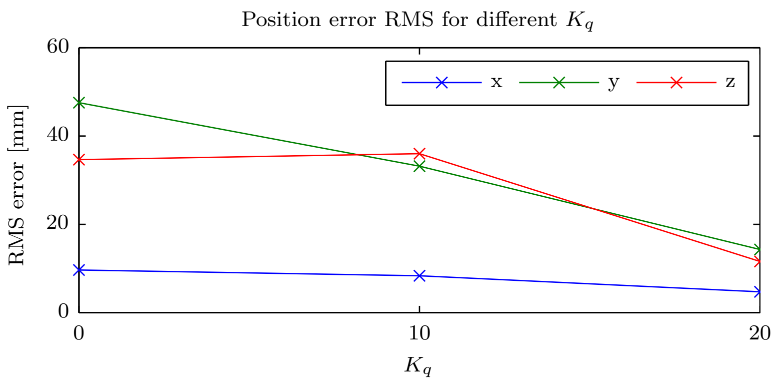


Figure 7-8: RMS error of the x,y and z position for different gain settings of the pitch damper. The sample duration was about only 11 seconds for $K_q = 10$, and 18 seconds for the other two.

7-2 Step inputs

Part of the results of these tests are shown in Figure 7-9 to 7-12. For a complete overview of all the results for different gain settings, see Appendix E-2. The data that is send down over Bluetooth, in this case the actuator commands, is time shifted by the average down link time delay of about 80 ms forwards to compensate. Even then it can still be seen that there is a delay between the change reference signal and the change in actuator input, caused by an irregular delay in either the up link or the down link.

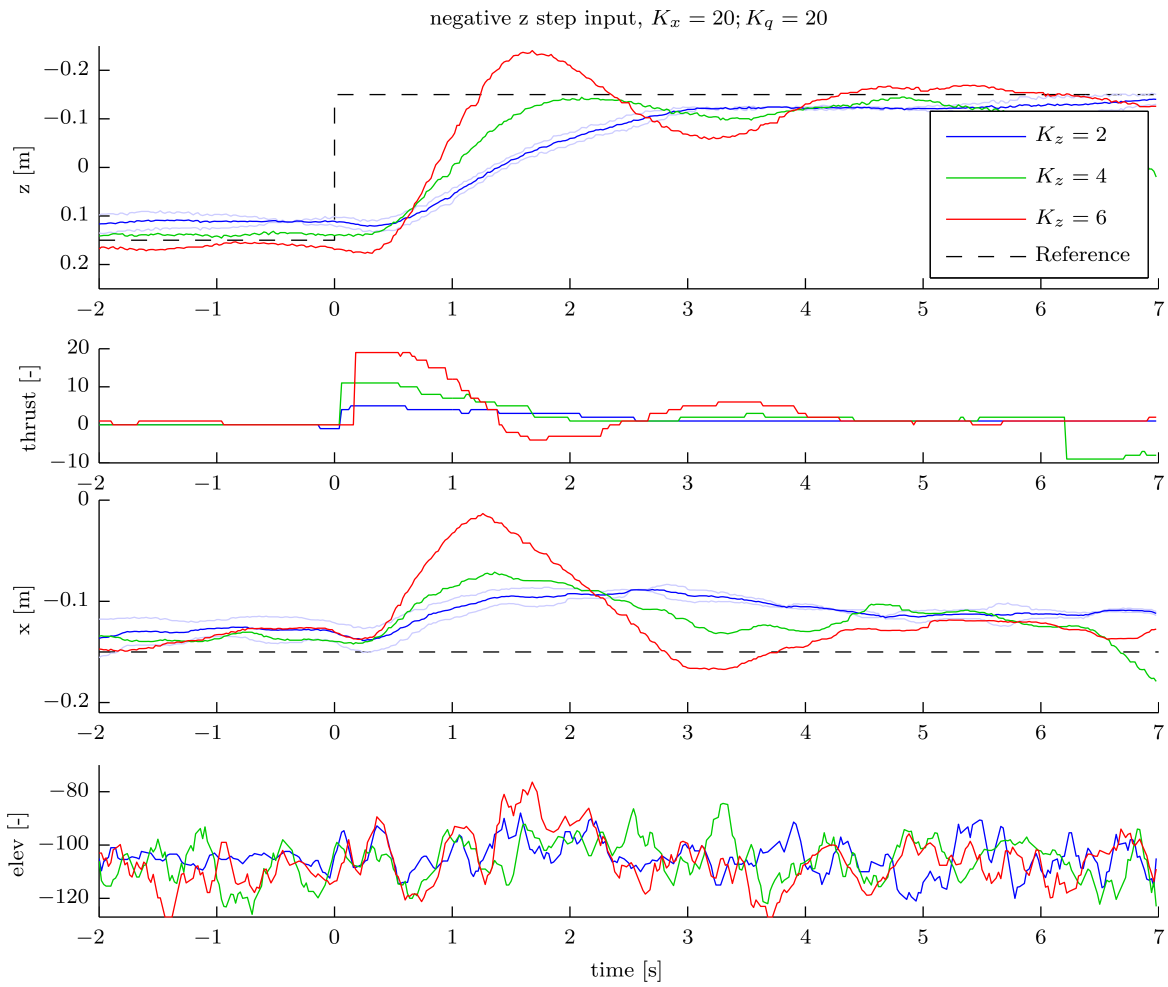


Figure 7-9: A 30 cm step input in negative z direction. The Delfly was instructed to fly at a fixed x_w reference position of -15 cm. The Delfly tries to follow the upward step by increasing thrust, therefore also increasing velocity. The resulting forward motion is clearly visible in the lower plot where the x position is shown. The single proportional gain on for the x deviation and the elevator command, is unable to counter this behavior. It can also be seen that higher gains on the thrust command make the Delfly react a lot quicker. For a gain of $K_z = 6$ the system shows under damped behavior, at K_2 over damped behavior, and at a gain of K_4 the system appears to be almost critically damped.

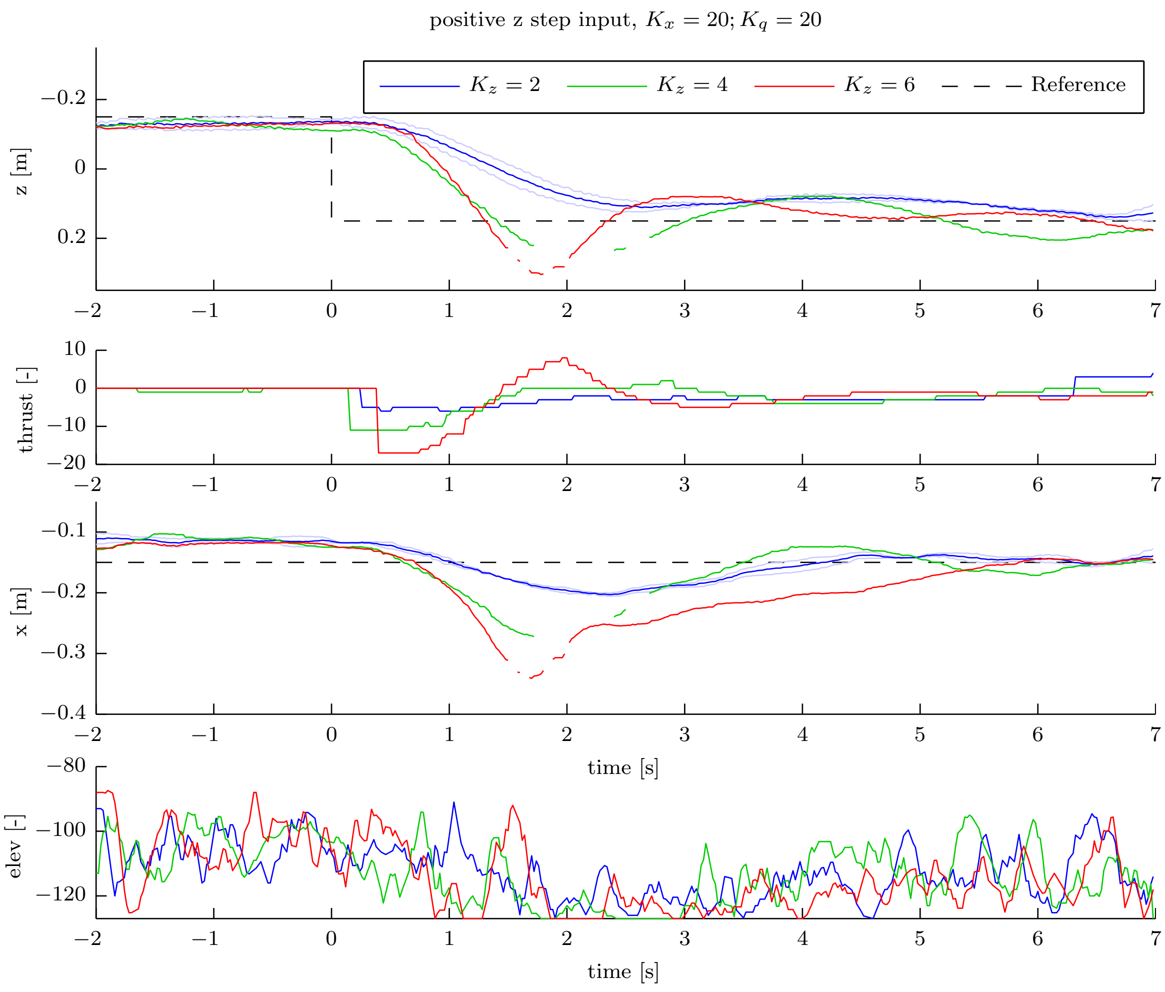


Figure 7-10: A positive step z-direction (i.e. downwards) shows that the Delfly is drifted backwards due to the lower thrust setting. The elevator is unable to respond quick enough to counter this motion in x direction. The change in x position, as well as the overshoot in the z direction, is a lot larger. Note also that with the same controller, a gain $K_z = 4$ now shows overshoot instead of the critically damped behavior, indicating non-linearities in the behavior of the Delfly.

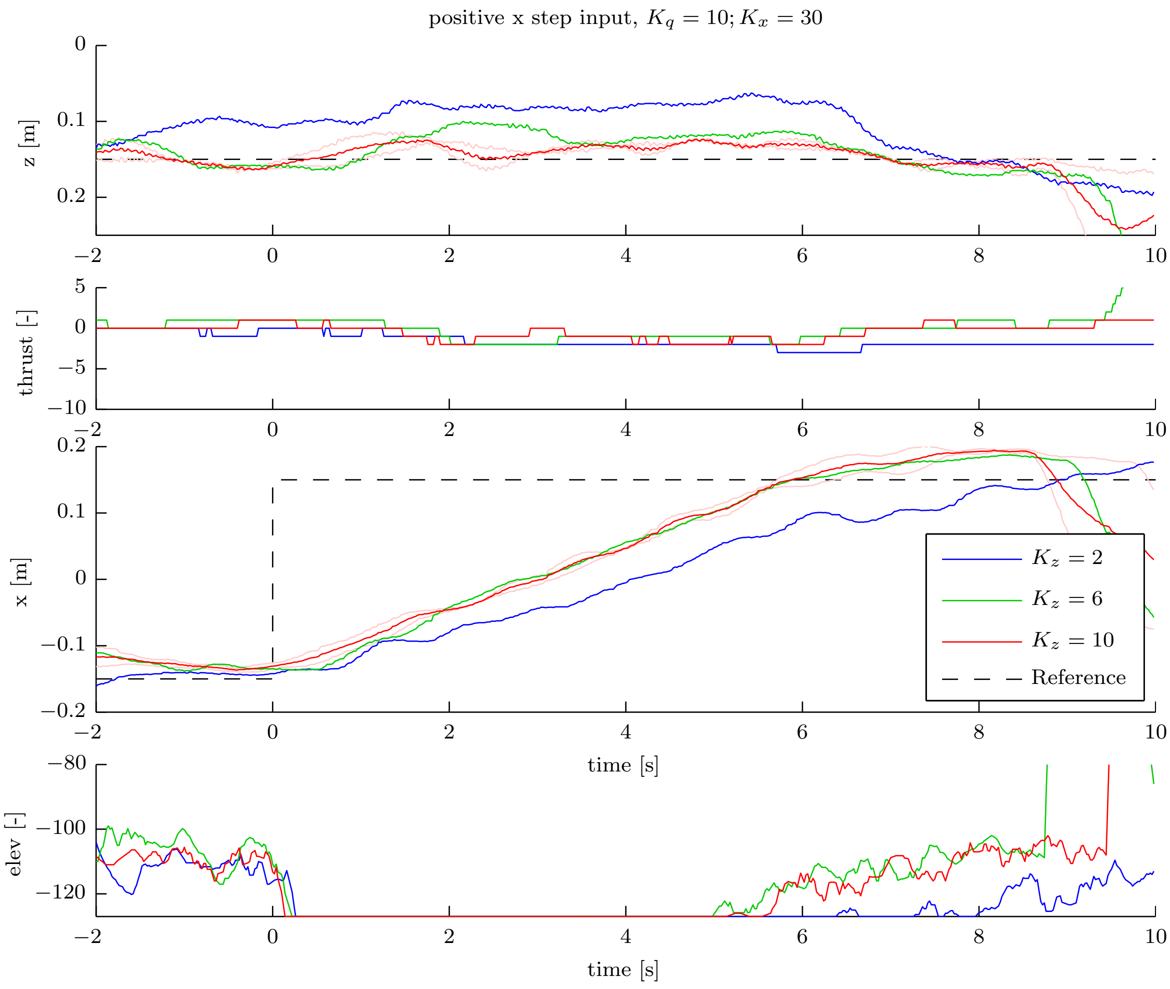


Figure 7-11: For a horizontal step forward, the Delfly reacts very slowly due to saturation of the elevator. It is not capable of pitching more forward, and the gain setting for the elevator has no real effect. During the forward motion, more lift is generated due to the higher forward velocity, and consequential there exists an error in z-direction.

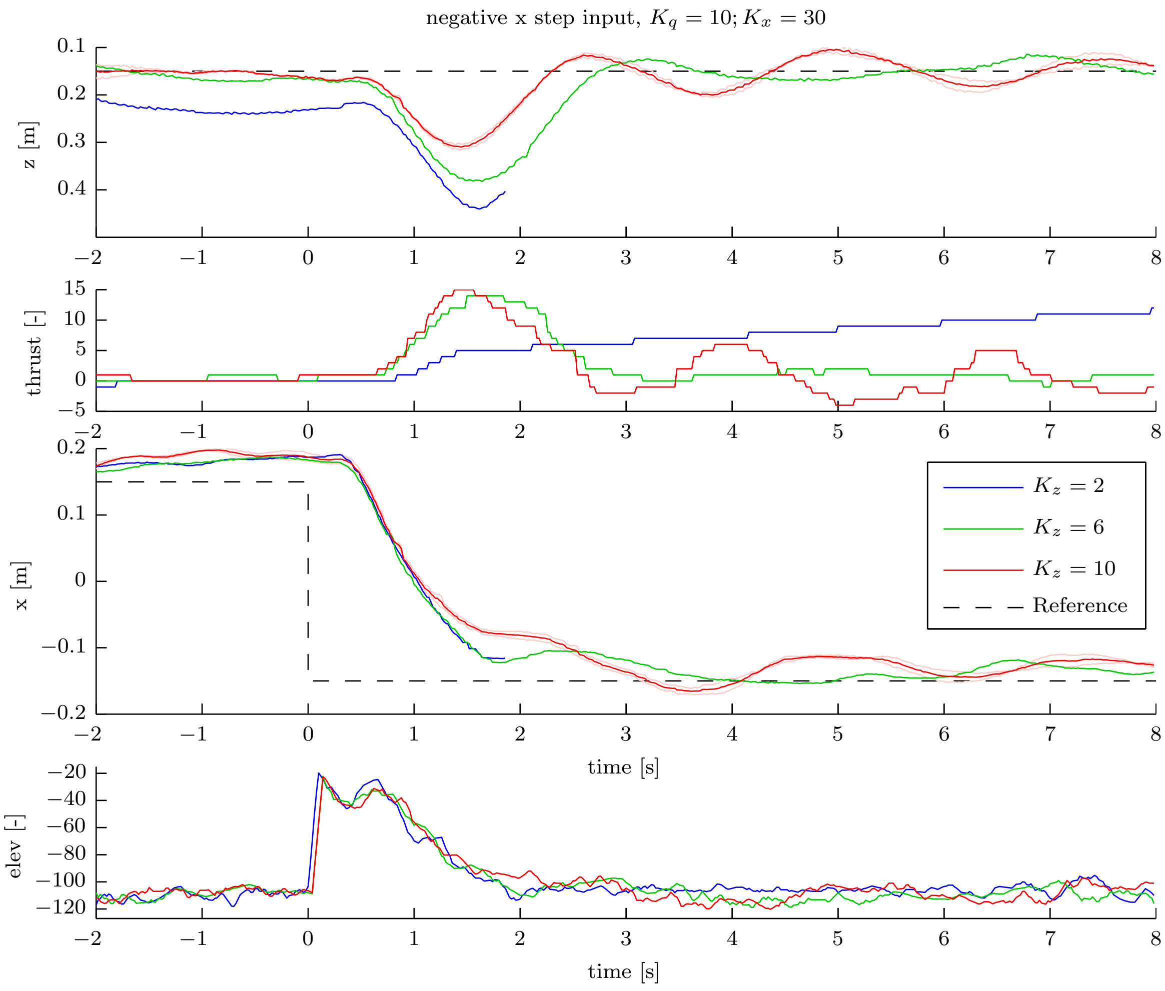


Figure 7-12: In the negative x direction, or a step with the direction of the relative wind, the Delfly is very quick to react. Increasing the vertical gain K_z reduces the effect on the altitude, but seemingly causes a undamped overshoot for $K_z = 10$. Note that for $K_z = 2$ there was a loss of the tracking LED, and no position data is available for this condition beyond $t = 1.9$ seconds.

Chapter 8

Discussion

First of all, it is striking how little control is needed as well the simplicity of the controller for the Delfly to achieve this degree of tracking accuracy, owing to its great inherent stability and the tunnels relatively low turbulence conditions.

8-1 Motor controller

The influence of the coarse throttle control, where the thrust was increased and decreased by discrete amounts, had the largest impact on the tracking performance. It tended to overrule all other sources of error. Luckily, it is also easy to remedy. The motor controller can be reprogrammed to provide more accuracy.

At moment where the throttle setting was constant, as is illustrated in Figure 7-5, the maximum deviation in z-direction decreased significantly, from ± 4.3 to only ± 1.5 cm (and the RMS error almost 30%). This also had its effect on the tracking performance in the two other dimensions, which increased by about 25 %. The latter can be attributed two couplings that were observed between the motion in the vertical plane and the longitudinal and lateral plane.

8-2 Longitudinal coupling

A coupling in the longitudinal plane was observed during testing, where an increase in horizontal velocity would cause an increase in altitude and visa versa. This effect is clearly visible in Figure 8-1, where the velocities in x- and z- direction are plotted.

The increase in horizontal velocity causes an increase in lift, which in turn accelerates the Delfly upwards. The resulting vertical velocity will cause the Delfly to slow down, and the Delfly drifts backwards, losing lift. This whole process causes a periodic motion where the vertical velocity lags 90° behind the horizontal velocity.

To counter this, the thrust should be slightly decreased when the horizontal velocity increases. The same effect is observed at helicopters, where an increase in forward velocity needs to be

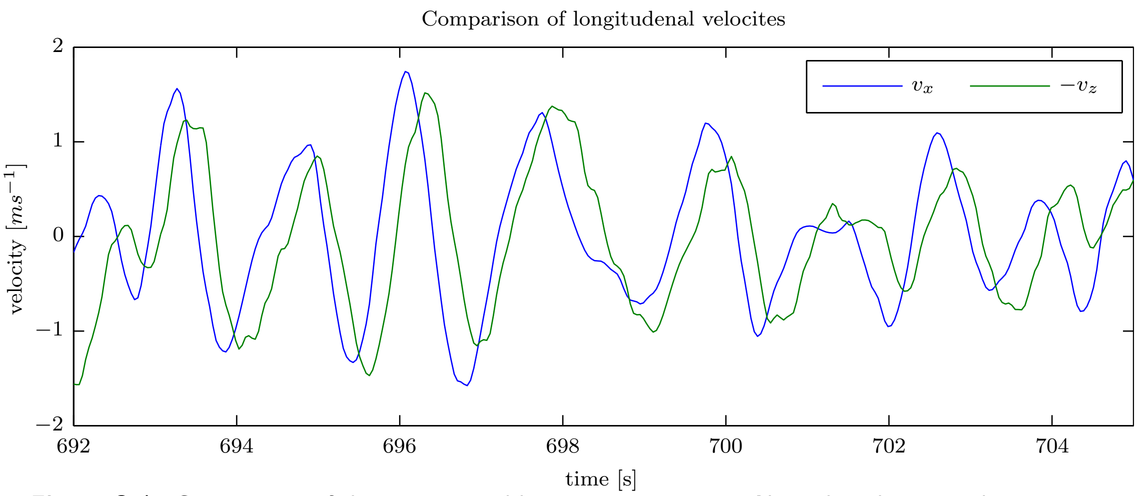


Figure 8-1: Comparison of the vertical and horizontal velocities. Note that, because the z-axis is normally defined downwards positive, the vertical velocity is plotted **inverted** to better show the relation between the two velocity components. This way it can easily be seen that the upward velocity lags 90 degrees behind on the forward velocity.

accompanied by a decrease in thrust. This so-called backside-of-the-power-curve is one of the non-linearities that can be identified in the Delfly.

8-3 Lateral coupling

From Figure 7-4 a coupling between the vertical and lateral position is distinguishable.

The motor controller appears to be responsible for the fact that this oscillation occurs. The range of the commanded motor setting is very small, only alternating between a value two values. An increase of just 1 unit causes the Delfly to rise, and keep rising, until the z-position reaches a certain threshold. The motor command then drops 1 unit, making the Delfly loose altitude. This is repeated and causes a continuous rise and fall with a period around 8 seconds. The motor controller does not have the required precision to adequately react to changes in height.

But the fact that the motor controller causes a periodic motion does not yet explain why the Delfly would deviate to the side whenever the thrust setting changes. It is thought that this is caused by asymmetries. The Delfly is built by hand, the battery is fastened with tape and replaced by hand, the wings are mounted by hand, and are subject to wear and tear. There are therefore many factors that can cause asymmetries.

When an asymmetric thrust is generated by the wings, due for example a difference in wing tension on both sides, the Delfly will have the tendency to yaw. In order to fly straight forward, the rudder will be deflected to restore the lateral force balance. This makes the Delfly fly with a side slip angle.

Now, when the thrust is increased, the force the rudder generates is increased, and the side slip angle is increased. Because of this different side slip angle, the camera will see the reference LED in the tunnel change, and the auto pilot commands the rudder to correct what it believes to be an error in heading. A new equilibrium is reached only when a lateral offset puts the reference LED at such an angle that it negates the side slip angle.

8-4 Influence of gyro feedback

Deriving information from the measurements on different gyro gains was not possible due to the few measurements that were taken and the large influence the imprecise motor controller had on the system. The discreteness of the thrust setting was the single biggest contributor to tracking errors, and it was therefore impossible to distinguish performance improvements due to the gyro's through the comparison of RMS errors.

It is believed though, that especially the yaw and pitch gyro contribute a lot to the stability and tracking performance of the Delfly in more dynamic situations, which will occur during normal flight. It was observed that the yaw gyro was very effective in reducing yaw oscillations when the Delfly was manually excited by giving a big rudder impulse input. Especially when the tail has lost some of its structural integrity, the yaw gyro feedback stabilized the Delfly within a second instead of a few seconds of oscillations without the gyro. The yaw gyro is currently not fitted on the auto pilot board (it had to be retro fitted in this project), but should be standard issue in future designs.

The effect of the pitch gyro on the low frequency pitch oscillation around 1 Hz was not directly observed. But the effectiveness of the elevator, as seen for example in the backward step input where an increase of 90 units would send the Delfly back at high velocities, indicates that it must be possible to reduce the oscillations with the proper filtering and feedback gain.

Chapter 9

Conclusions

It has proved possible to achieve autonomous flight with the Delfly in the wind tunnel for velocities up to 0.8 m/s. The employed PI controller is able to keep the Delfly at the reference point accurately, where the Delfly can be guaranteed to stay within ±2cm in x-direction, ±4.3cm in z-direction, and ±6.3 cm in y-direction of the reference point. At times performance can be even better, with stretches up to 15 seconds where the Delfly stays within ±1.7 cm, ±3.5 cm and ±1.7 cm for respectively x, y and z direction. This is sufficient performance to allow PIV measurements to be done. The current implementation is capable of delivering consistent performance over long periods of time, allowing for ample measurement time.

There is still significant room for improvement, where especially the motor controller needs attention, but it is the first undertaking in the world to let a flapping wing UAV fly freely in the wind tunnel, and a proof-of-concept has been achieved.

Furthermore, for the first time, good quality data has been gathered on the dynamic behavior of the Delfly in slow forward flight. The information gathered during the step input tests, can be used for future projects as a starting point for the design of more advanced controllers that cope with the observed non-linearities, and provide a reference for future research on the dynamics of the Delfly.

For forward flight at velocities of 2.0 m/s and above, the Delfly is not capable of flying in this flight regime without major modification to the position of the c.g.. Autonomous flight was not achieved with the employed system, because there was not enough time to thoroughly test the proposed controller. But if given more time to fine-tune and implement some minor improvements it is very probable that with the current implementation of the soft- and hardware it is possible to achieve autonomous flight at these velocities as well.

Chapter 10

Recommendations

Tracking system

During the project the value of the WiiMote 3d Tracking system was very soon appreciated. But it can readily be improved by increasing the number of camera's so the tracking volume can be increased. Also, by directly interfacing the camera to a serial port a very small and portable package can be made, with data rates up to 200 Hz. These improvements would also allow a cheap (at 50 euro per camera plus other hardware, in total 1000 euro for a 10+ camera set-up), flexible and high performance tracking.

State estimation

A Kalman filter could give a very good estimate of the complete state of the Delfly, and could use the gyro's to increase the reliability of the attitude information.

It is a very viable option to put three LEDs on the Delfly, two on either side of the horizontal tail plane, and one on the main body. Using the ability of the tracking systems to track up to four points, the attitude can be very precisely estimated (using the explained TRIAD algorithm) in all 3 DoF.

On-board systems

For more advanced controllers a better CPU should be fitted on the Delfly, because the current software is running into the limitations of the controller. 7 kilobyte of the available 8 kilobyte is used. About 5.5 kbyte is already used for all the functionalities, especially for the communication and code for reading all the sensors, leaving not much more room for more advanced techniques like a Kalman filter or more elaborate controllers.

A three directional gyro is really recommended to be included in the basic design. The retro fitting of a third gyro was usable, but a smaller and lighter package can be made when it is all integrated in one chip. Also the inclusion of accelerometers is advised, to gain even more knowledge on the flight properties of the Delfly.

The motor controller will need to be improved. At the moment the range is too high and the precision too low. From the complete range of -128 to 127 only up to around 30 has been used, and neither has lower than -50. Therefore, without increasing the bit depth it should be possible to increase the precision by a factor of 3, if the other components allow.

Communication

The communication through Bluetooth proved to be cumbersome throughout the project. When the conditions were not optimal, packages would pile up in the sending buffer, causing large latencies. The data rate had to be tuned down a lot in order to make sure no packages would get delayed in buffers, severely under utilizing the available bandwidth. But the Bluetooth protocol should be capable of providing a much better connection for streaming data, and it is strongly suspected that the low performance is caused by wrong settings in the flow control of the Bluetooth modules. This is the first thing that should be investigated in more detail.

Another possible improvement would be to buffer the down link data and send it out in larger packages, better tuned to the minimal package size of Bluetooth (which is around 480 bytes). This way the available bandwidth can be far better utilized.

If a closer investigation of the Bluetooth communication does not provide sufficient performance gain, a communication device which allows more control over the actual protocol than Bluetooth is advised. Especially for the up link, low and constant latency are of the utmost importance, and this can be better guaranteed when there are less protocol layers involved in the communication. Currently, wireless serial communication devices like the X-bee are not small and light enough, but maybe a custom solution can be found, modifying the X-bee to be as light weight as possible by removing all excess components and material.

Model identification

As can be seen from Chapter 7, the employed setup is very well suited for gathering useful data for model identification. It is reckoned that these preliminary measurements can provide an idea of what kind of measurements need to be performed.

Appendix A

Additional measurements

A-1 Camera distortion Wiimotes

Two WiiMote with infrared cameras were used, as blue one and a black one. In order to get an estimate of the distortion of the WiiMote cameras, they were mounted on a CNC milling machine, downward looking, and moved over a infrared LED in straight lines. In this way, the severity of the distortion could be estimated by looking at the straightness of the lines, drawn out by the projection of the LED. As can be seen, little distortion is present (it is only in the order of a few pixels), which was deemed not enough to justify extra distortion correction.

Ignoring the slight distortion, the internal parameters could be determined using Zhang's method, which is used by the `calibrateCamera`-routine of the OpenCV library as described in Section 3-2. The complete results of the calibration are shown in Table A-1 and A-2. The final values are the average of the four calibrations.

	f_x	f_y	c_x	c_y
1	1.267	1.704	0.468	0.559
2	1.250	1.684	0.482	0.571
3	1.266	1.706	0.460	0.569
4	1.266	1.709	0.491	0.560
avg	1.2622	1.7007	0.4752	0.5647
(std)	(0.0082)	(0.0114)	(0.0139)	(0.0061)

Table A-1: Internal parameter calibration results for the blue WiiMote.

	f_x	f_y	c_x	c_y
1	1.267	1.744	0.513	0.500
2	1.282	1.708	0.499	0.508
3	1.290	1.719	0.504	0.500
4	1.349	1.794	0.504	0.493
5	1.334	1.778	0.518	-.—
avg	1.313	1.749	0.508	0.500
(std)	(0.0284)	(0.0370)	(0.0077)	(0.0061)

Table A-2: Internal parameter calibration results for the black WiiMote.

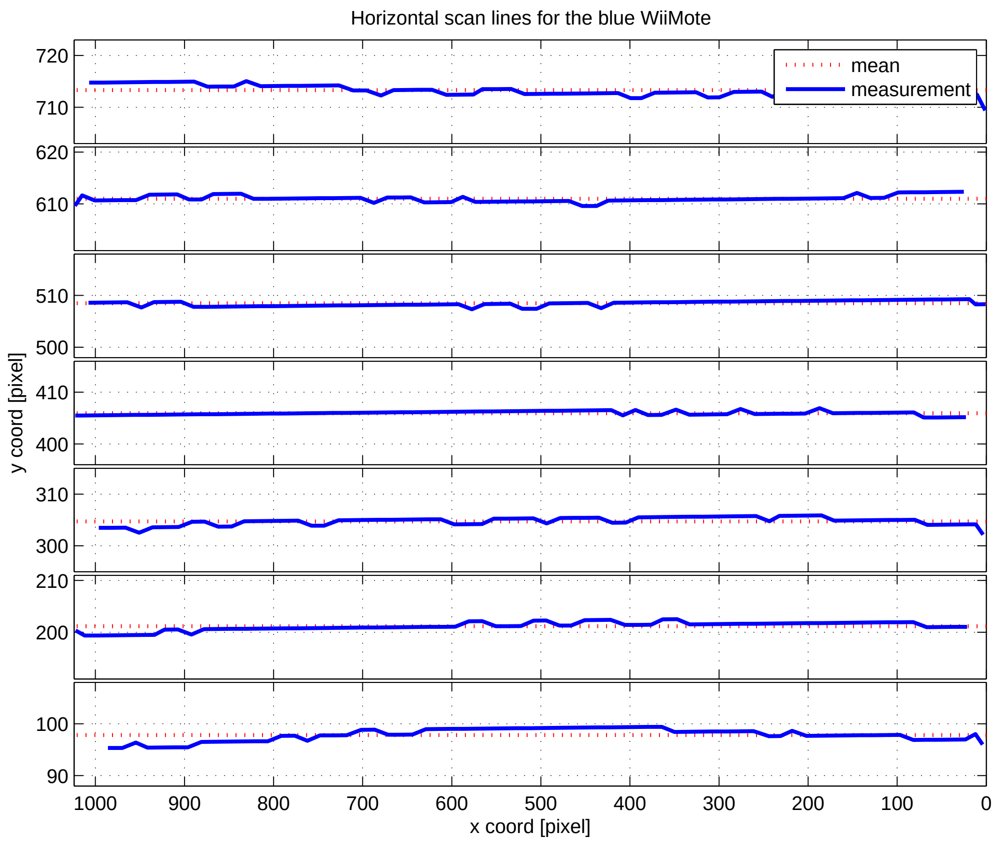


Figure A-1: Calibration tracks for the blue wiimote in the horizontal direction

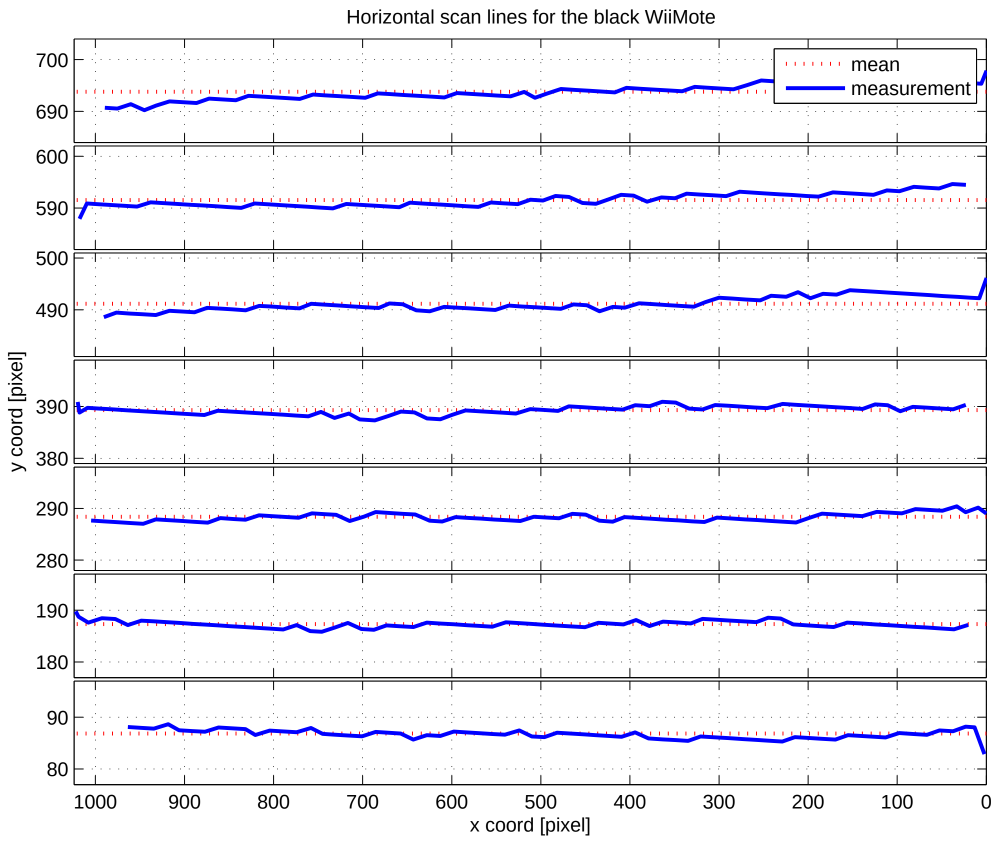


Figure A-2: Calibration tracks for the black wiimote in the horizontal direction

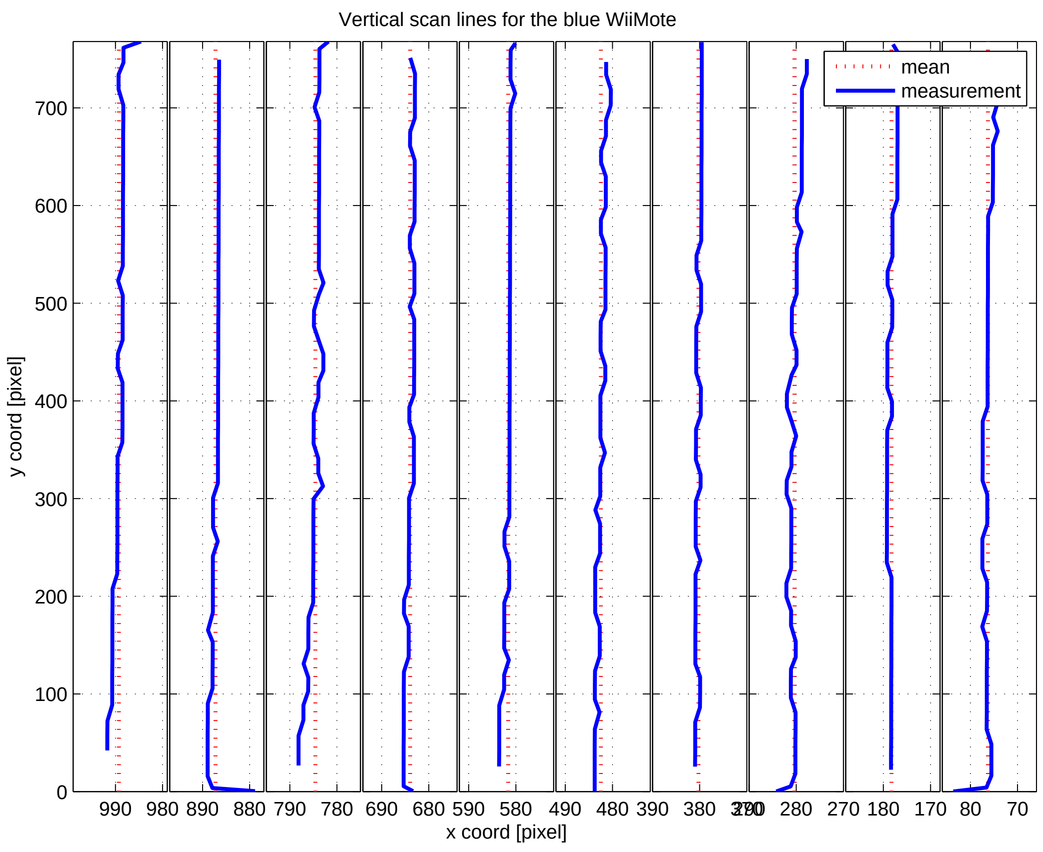


Figure A-3: Calibration tracks for the blue wiimote in the vertical direction

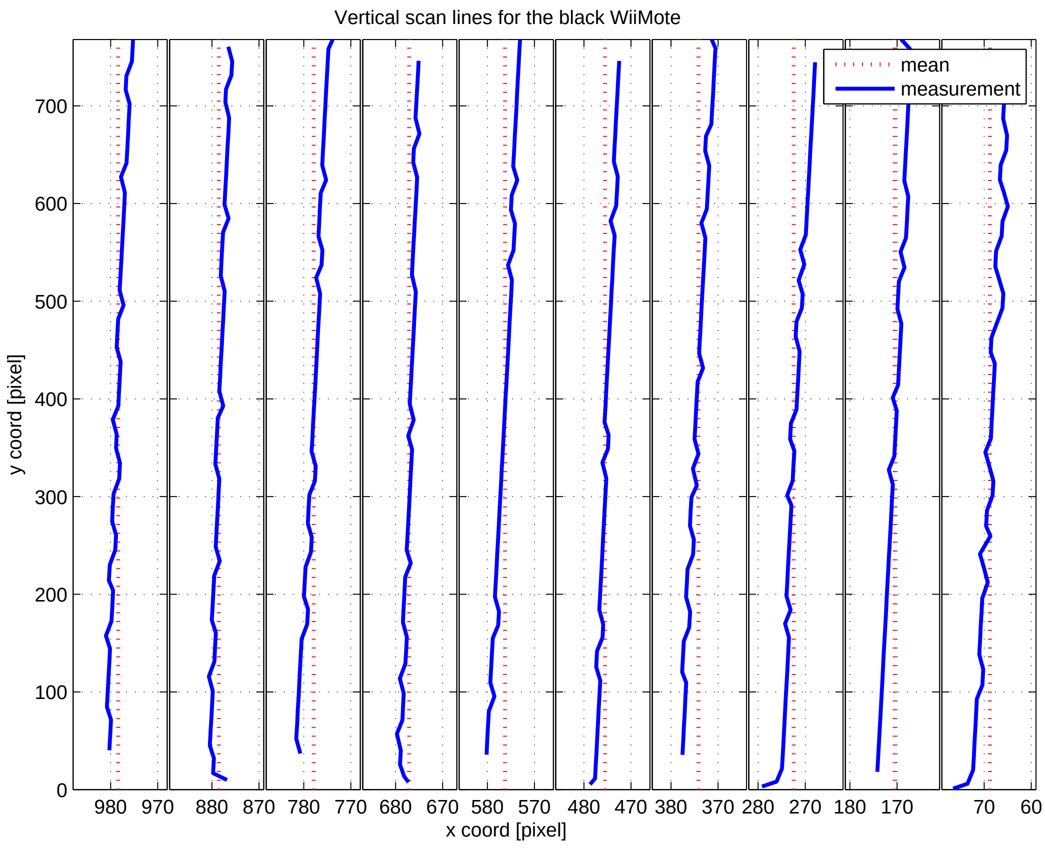


Figure A-4: Calibration tracks for the black wiimote in the vertical direction

A-2 Center of gravity

The center of gravity of the Delfly was measured by balancing it without a battery on a ballpoint pen. A photo was taken and three points were indicated: the leading edge (LE), trailing edge (TE) and the position of the tip of the pen. The LE was defined as the point where the arms of the two upper wings border each other. The TE was defined as the most forward point of the flat extension on the wing tensioner. See Figure A-5 for a visual indication of how these points are defined.

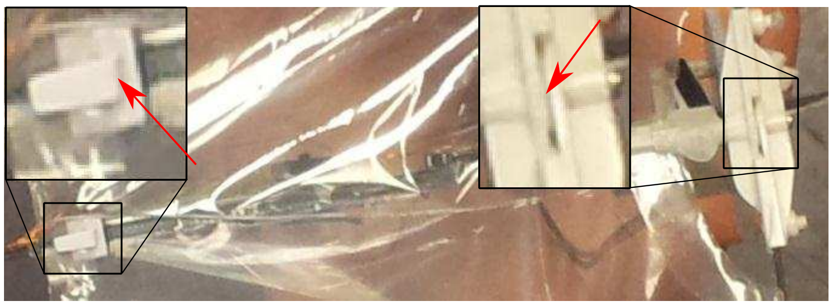


Figure A-5: A close up of the upper side of the wings, where the definition of the trailing edge (left) and the leading edge are indicated by the two red arrows.

The distance between the LE and the TE is called the chord length c. We can express any point along the fuselage in its coordinate d, where d is the distance between the LE and the point of interest, divided by chord length c. The LE is therefore by definition at $d = 0$ and the TE at $d = 1$.

From the photos the c.g. was determined by dividing the pixel distance between the LE and the ball point pen by the pixel distance between the LE and the TE.

Using the weight of the Delfly m_D and the weight of the battery m_{bat} (shown in Table A-3) and knowledge of the c.g. without a battery $cg_{\hat{D}}$, the location of the total c.g., including the battery, can be expressed as follows

$$cg_D = \frac{m_{bat}cg_{bat} + m_D cg_{\hat{D}}}{m_{bat} + m_D} \tag{A-1}$$

where cg_{bat} is the location of the c.g. of the battery expressed in chord lengths, as determined from photos.

In order to put the c.g. sufficiently far forward for forward flight, an extension rod was mounted on the Delfly. This way, the battery could be placed in front of the wings and motor gearing, effectively placing the c.g. between 30% and 45% chord length. The weight of this rod has to be taken into account, so Equation A-1 becomes

$$cg_D = \frac{m_{bat}cg_{bat} + m_D cg_{\hat{D}} + m_{rod}cg_{rod}}{m_{bat} + m_D + m_{rod}} \tag{A-2}$$

where m_{rod} is the weight of the rod (as shown in Table A-3) and cg_{rod} the location of its c.g., expressed in chord lengths from the leading edge.

	weight [gr]	location [chord length]
Delfly	11.3	0.87
Battery	5.64	variable
Rod	0.176	-0.36

Table A-3: Weight breakdown of the Delfly.

Several measurements of the center of gravity were taken with the battery at different locations. The results are shown in Figure A-6.

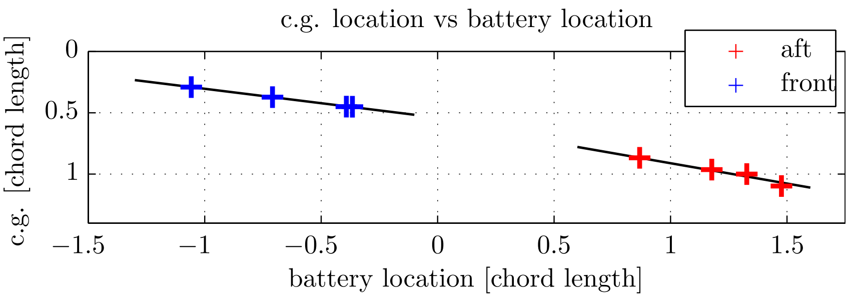


Figure A-6: Center of gravity measurements for various battery positions.

For both cases shown in the figure a linear least squares solution was found.

Linear least squares for the battery in front of leading edge (mounted on extension rod):

$$cg_D = 0.24cg_{bat} + 0.54 \tag{A-3}$$

Linear least squares for the battery aft of leading edge (no extension rod mounted):

$$cg_D = 0.33cg_{bat} + 0.58 \tag{A-4}$$

With Equation A-3 and A-4 the center of gravity can be determined for an arbitrary battery position. Before each test run, a side view photo was taken of the Delfly to register the position of the battery, and the c.g. was calculated. This information is documented in Appendix E-1. Although the battery was placed by hand and there was no predestined location for the battery, it was still placed within a margin of error of 2% every time it was replaced.

Appendix B

Implementation details

B-1 Controller

In practice, to translate the equations of the controller explained in Chapter 5 to C-code, several things need to be taken into account.

Most importantly, the on-board CPU can only handle fixed-point operations. Special care needs to be taken with any division, because all decimals will be thrown out. Also, to save memory on the CPU we like to use as little bit-depth as possible. All state values are stored in 16 bit integer format. When we calculate the controller input, the 16 bit state information is first converted to 32 bits, then the calculations are done and the resulting value is scaled back to 16 bit. The resulting value is then converted to 8 bit, the range of the servos and motor controller.

Good practice would be to include a check to see whether the 32 bit temporary value would overflow when cast back to the 16 bit command value. This is not implemented in the used controller, but no adverse effects have been observed. It is recommended though that this is always done (in this case, it was only done when converting the 16 bit command value to the 8 bit actuator command).

Because we can only set 8 bit integers (values between -128 and +127) as gains due to the way the communications protocol is set up, we need to do some hard coded scaling. The scaling is chosen such that the result of the total calculation falls within the range of the servo/motor controller for a gain of moderate magnitude (about ±50) and for average expected input. Take for example the integrator term in the rudder controller. The integrator is incremented each loop by 1/10 of the current y position value (which is in millimeters). It is expected that the integrator can accumulate to about 100 mm over a 15 second period. The loop rate is 100 Hz, so the value of the integrator would accumulate to

$$\frac{150}{10} \cdot 15 \cdot 100 = 22500$$

Gain	Divisor	Scaler	State	Unit
K_x	100	1000	x	$[m]$
K_z	100	1000	z	$[m]$
K_x^I	18000	10000	$\sum x$	$[ms]$
K_y^I	18000	10000	$\sum y$	$[ms]$
K_z^I	18000	10000	$\sum z$	$[ms]$
K_p	64	5.55	p	$^\circ s^{-1}$
K_q	100	5.55	q	$^\circ s^{-1}$
K_r	64	4.17	r	$^\circ s^{-1}$
K_ψ	256	8	$(u - 384)$	pixel

Table B-1: Overview of the scaling factors of the slow forward flight controller, used to calculate the relationship between the sensor input, and the actuator commands. The 3rd column shows on which variable the gain acts, and the 4th column its respective unit, with possibly a scaling factor.

if we expect this value to have the effect of an extra rudder deflection of 40 units for an average gain setting of about 50, we can estimate the scaler value needed to bring the integrator value in this range.

$$s = \frac{22500 \cdot 40}{50} = 18000$$

Of course, these values may have to be revised during actual flight, in case a gain cannot achieve sufficient effect.

Besides the gain scaling, state variables were also scaled because there were no floating point numbers. The angular rate for example was scaled by 5.55 from $^\circ s^{-1}$ so it would have a large enough range and sufficient precision, when stored in a 16-bit fixed point variable. The complete overview of all the scaling factors employed can be found in Appendix B.

The applied gain and state scaling factors are listed in Table B-1 for the slow forward flight controller, and Table B-2. Sometimes a scaling factor is already applied to the variable when is stored. This factor is shown in the 3rd column. So for example, the integrator value of variable x, called `x_integrator`, is not stored in $[ms]$, but is scaled by 10000.

So to know the gain as applied to a state expressed in physical SI units, the following formula has to be applied to a gain as how it is used in the code:

$$K_{phys} = \frac{\text{scaler}}{\text{divisor}} \cdot K_{code}$$

The C-code used for both controller is also shown in Listing B.1 and Listing B.2. The limitation of only having 15 gains to be set, meant that some gains were named a bit odd. Because the velocity feedback was not used in the slow forward flight controller, the gains originally created for this were used as the integrator gains K^I. For the forward flight controller, the velocity feedback was used, and therefore the unused gains K_p and K_θ were used for resp. the x- and z-integrator terms.

Gain	Divisor	Scaler	State	Unit
K_x	100	1000	x	$[m]$
K_z	100	1000	z	$[m]$
K_x^I	18000	10000	$\sum x$	$[ms]$
K_y^I	18000	10000	$\sum y$	$[ms]$
K_z^I	18000	10000	$\sum z$	$[ms]$
K_p	64	5.55	p	$^\circ s^{-1}$
K_q	100	5.55	q	$^\circ s^{-1}$
K_r	64	4.17	r	$^\circ s^{-1}$
$K_m u$	256	8	$(u-384)$	pixel

Table B-2: Overview of the scaling factors of the forward flight controller, used to calculate the relationship between the sensor input, and the actuator commands. The 3rd column shows on which variable the gain acts, and the 4th column its respective unit, with possibly a scaling factor.

Listing B.1: C-code of the employed controller during slow forward flight.

```
int32_t temp_calc = 0;

int16_t cmd_rudd = 0;
int16_t cmd_elev = 0;
int16_t cmd_thru = 0;

void vertical_controller(void)
{
    temp_calc = ( ((int32_t)delfly.gain.psi)     * ((int32_t)delfly.ledY_lp) )/256   +
                ( ((int32_t)delfly.gain.p)       * (((int32_t)delfly.state.p)/8))/8  +
                ( ((int32_t)delfly.gain.r)       * (((int32_t)delfly.state.r)/8))/8  +
                (  (y_integrator/10)             * ((int32_t)delfly.gain.vy) )/1800  +
                ( ((int32_t)delfly.gain.k) ); // trim
    cmd_rudd = (int16_t)temp_calc;

    temp_calc = ( ((int32_t)delfly.gain.x)       * ((int32_t)delfly.state.x) )/100   +
                (  (x_integrator/10)             * ((int32_t)delfly.gain.vx) )/1800  +
                ( ((int32_t)delfly.gain.theta)   * ((int32_t)delfly.state.theta))/100+
                ( ((int32_t)delfly.gain.q)       * ((int32_t)delfly.state.q) )/100   +
                ( ((int32_t)delfly.gain.m));     //trim
    cmd_elev = (int16_t)temp_calc;

    temp_calc = ( ((int32_t)delfly.gain.z)       * ((int32_t)delfly.state.z) )/100   +
                (  (z_integrator/10)             * ((int32_t)delfly.gain.vz) )/1800  +
                (  (int32_t)delfly.gain.n);  //trim
    cmd_thru = (int16_t)temp_calc;
}
```

Listing B.2: C-code of the employed controller during forward flight.

```
int32_t temp_calc = 0;

int16_t cmd_rudd = 0;
int16_t cmd_elev = 0;
int16_t cmd_thru = 0;


void horizontal_controller(void)
{

    temp_calc = ( ((int32_t)delfly.gain.psi)     * ((int32_t)delfly.state.psi)    )/64     +
                ( ((int32_t)delfly.gain.r)       * (((int32_t)delfly.state.r)/8))/8        +
                ( ((int32_t)delfly.gain.y)       * (((int32_t)delfly.state.y)/8))/20 +
                ( ((int32_t)delfly.gain.k) ); // trim
    cmd_rudd = (int16_t)temp_calc;

    temp_calc = ( ((int32_t)delfly.gain.z)       * ((int32_t)delfly.state.z) )/100   +
                ( ((int32_t)delfly.gain.vz) * ((int32_t)delfly.state.vz)     )/100   +
                ( ((int32_t)delfly.gain.theta)   * (z_integrator/10))/1800   +
                ( ((int32_t)delfly.gain.q)       * ((int32_t)delfly.state.q) )/100   +
                ( ((int32_t)delfly.gain.m));     //trim
    cmd_elev = (int16_t)temp_calc;

    temp_calc = ( ((int32_t)delfly.gain.x)  * ((int32_t)delfly.state.x) )/100   +
                ( ((int32_t)delfly.gain.vx)      * ((int32_t)delfly.state.vx)     )/100   +
                (  (x_integrator/10)             * ((int32_t)delfly.gain.p)       )/1800  +
                    (  (int32_t)delfly.gain.n);   //trim
    cmd_thru = (int16_t)temp_calc;

}
```

B-2 Communication protocol

Because of data limitations, the messages were encoded in the formats showed below to make optimal use the available bandwidth. A message can either be a position/velocity update, or a configuration message. A message is build up either with 5 bytes (position/velocity message) 3 bytes (configuration message) or 5 bytes (joystick message). The first bit of each byte is reserved for the start indicator. For the first byte this bit is one, for the other bytes it should be zero. Furthermore, the second (and third bytes) represent a header that tells what message type it is, see Table B-3. Note that for a position message the third bit actually contains information, and is no part of the header.

2nd bit	3rd bit	Message type
0	x	Position/velocity
1	1	Configuration
1	0	Joystick message

Table B-3: The header definition for communication messages

The complete messages are shown in the tables below. For reference, the tables also include the bit-mask for each variable.

Position/velocity message

This message consists of 9 bytes. The second bit of the first byte is 0 for a position message. The rest of the bits form three 10 bit and three 8 bit integers, one after another. The last 12 bits compose the platform velocity.

Byte	0	1	2	3	4	5	6	7	Mask	
0	1	0	x9	x8	x7	x6	x5	x4	0x3F	
1	0	x3	x2	x1	x0	y9	y8	y7	0x78	0x07
2	0	y6	y5	y4	y3	y2	y1	y0	0x7F	
3	0	z9	z8	z7	z6	z5	z4	z3	0x7F	
4	0	z2	z1	z0	u7	u6	u5	u4	0x70	0x0F
5	0	u3	u2	u1	u0	v7	v6	v5	0x78	0x07
6	0	v4	v3	v2	v1	v0	w7	w6	0x7C	0x03
7	0	w5	w4	w3	w2	w1	w0	p12	0x7E	0x01
8	0	p11	p10	p9	p8	p7	p6	p5	0x7F	
9	0	p4	p3	p2	p1	p0	--	--	0x7C	

Table B-4: Position/velocity message

The position is interpreted in mm, while the velocity indicates mm/sec.

Config message

A config message is indicated by a 1 in the second bit of the first byte and a 0 in the third bit. The The next 4 bits indicates which value, and the last bit plus the last 7 of the second byte contain the value.

In practice, a joystick message was sometimes mistaken for a configuration message, which could lead to disastrous results. Therefore, a check sum was included. This makes sure the message we receive indeed is a configuration message, and that the content is not corrupted.

The checksum is generated by applying the XOR (exclusive OR) operator to the two bytes that make up the configuration messages, and appending this as the third byte. The Delfly performs the same check upon arrival of the message, and only accepts it when the check sum is the same. Therefore it is good practice to send a configuration message twice to make sure it arrives, because there is no acknowledgment feature implemented at the moment. The XOR is only performed on the last 7 bits of the two configuration bytes.

Byte	0	1	2	3	4	5	6	7	Mask	
0	1	1	0	i3	i2	i1	i0	b7	0x1E	0x01
1	0	b6	b5	b4	b3	b2	b1	b0	0x7F	
2	0	c6	c5	c4	c3	c2	c1	c0	0x7F	

Table B-5: Config message. Bits named *i* compose the 4 bit index, *b* the value of the gain referenced by the index. The last 7 bits of the 3rd byte, labeled *c*, is the checksum.

The gains are put in a union structure. This means the gains can also be accessed by their index in the array. These indexes are displayed in Table B-6.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
x	y	z	vx	vy	vz	ph	th	ps	p	q	r	k	m	n

Table B-6: Overview of the indices of the gains.

A value with index 15 is used for other kind of messages, the meaning of which is determined by the value of the 8 bits that normally compose the gain value.

val	meaning
0	set delfly.fms to FMS_LAND (0)
1	set delfly.fms to FMS_HOVER (1)
2	set delfly.fms to FMS_FLY (2)
5	set delfly.mode to MOD_NONE (0)
6	set delfly.mode to MOD_AP (1) (auto-pilot)
7	set delfly.mode to MOD_MANUAL (2) (joystick control)
8	initiates gyro calibration
9	puts the Delfly in emergency mode (centers actuators, shuts off engine, zeros joystick inputs and goes into FMS_LAND and MOD_NONE
11	rudder step input
12	elevator step input
13	motor step input
14	rudder sinoid input (not implemented)
15	elevator sinoid input (not implemented)
16	zeros the values of the x,y,z position integrators

Table B-7: Meaning of different values of the value byte when the index is set to 15.

Joystick message

A joystick input message is longer, containing the x,y,z and thrust of the joystick, in 8 bits signed integer format.

B-3 Latency

The system that was the most troublesome throughout the project was the communication sub-system. Because of the weight and size restraints of every component on the Delfly, the choice for the hardware providing a two-way digital link was very limited.

Early on in the project it was decided that a small Bluetooth module would be used, because it was easy to interface with any laptop, relatively cheap, small and light-weight.

The disadvantage of Bluetooth is the protocol consists of a stack of layers, each providing a progressively low-level interface from the layer above. This causes a lot of things to happen in the background over which there was no control. The most straightforward Bluetooth 2.0

Byte	0	1	2	3	4	5	6	7	Mask	
0	1	1	1	x7	x6	x5	x4	x3	0x1F	
1	0	x2	x1	x0	y7	y6	y5	y4	0x70	0x0F
2	0	y3	y2	y1	y0	z7	z6	z5	0x78	0x07
3	0	z4	z3	z2	z1	z0	t7	t6	0x7C	0x03
4	0	t5	t4	t3	t2	t1	t0	0	0x7E	

Table B-8: Joystick message structure profile was used, Serial-Port-Profile (SPP), which should provide a TCP-like virtual serial port, which should have a practical data rate of about 2.1 Mbit per second.

The information send over the up-link was a stream of position/velocity messages of 9 bytes each, apart from the occasional configuration messages to change the flight mode, engage joystick, or change a gain value. This means that at a loop rate of 100 Hz, the data rate usage would be about 7.2 kbit per second, ignoring overhead. The down link message size was 16 bytes, which at 100 Hz would mean a data rate of 12.8 kbit per second. Combined this would mean 20 kbit per second, about 1% of the data rate of the Bluetooth 2.0 specification. With the overhead included, this would still stay well within 2% of the specified value. It was also within the capability of the UART connection. which was set at 38400 Baud,

But in practice, when sending the information back and forth at 100Hz, the communication system could not keep up, and a buffer would fill up. A severe time delay would occur, growing over time to over 1 second. This also indicated that the SPP was not functioning in a TCP kind of way, but was making sure that packages arrived by resending the data until it was received. This is not desirable behavior when we rather have the system reject old position messages whenever newer information is available.

Because latency of this magnitude is unacceptable, the rate at which the data would be send was decreased, until no adverse effects were observed. After testing at several different rates, it was observed that the system could handle a send rate for the up link of 12.5 Hz, and a send rate for the down link 20 Hz. Receiving was done at 50 Hz for the ground station, and the Delfly would check for new position messages at 100 Hz. In Figure B-1 the flow of information is shown due to these different sent and receive rates.

At these sending rates, the total round trip time was about 160 ms on average. This was measured by looking at the time between when the tracking system would report that the Delfly was in sight and when the first 'position received'-acknowledgement came back from the Delfly (the Delfly would always report if it had received a fresh position update in between two subsequent log-messages). As is indicated in Figure B-1, an average of 5 ms delay can be expected due to the 100Hz loop on-board the Delfly, and another 10 ms delay can be expected to be caused by the ground station 50Hz loop rate. If the up link and down link delay are assumed to be equal, this would mean a one-way delay of $\frac{160-5-10}{2} = 73$ ms.

B-3-1 Message interval

Besides low latency, a regular interval between subsequent messages is also important. Gaps in received position updates would indicate problems with packages getting lost, or staying in a buffer too long. Figure B-2 shows the time difference between two received acknowledgements. This should be on average 8 ms, and it turns out this is the case. The spread around 8 ms does not necessarily indicate that the reception of position messages by the Delfly is irregular.

The Delfly send and receive loop are synchronized, as they run on the same chip. The ground station send and receive loop are also synchronized. But the total round trip time will vary because the two are not synchronized with one another. The maximum round trip *difference* is achieved when the first sample arrives very early, and the last arrives very late. Assuming that the one way transmission delay is equal to 70 ms, and is equal for both up and down link, and the he minimum round trip is equal to twice the transmission delay,

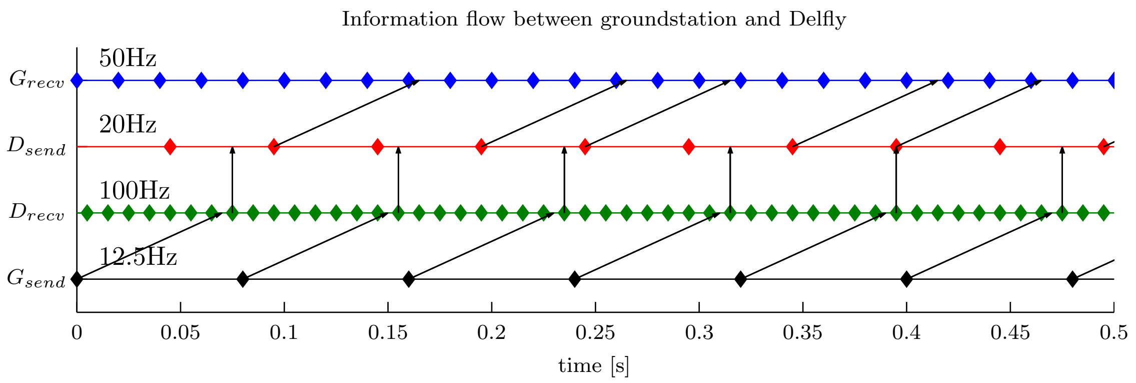


Figure B-1: The flow of information due to the different rates of each platform. Each node is the time at which the data is processed by each actor, which can be either sending or receiving. The delay between transmitting a message and receiving a message (one-way), indicated by the slant of the arrows, was estimated to be about 70 ms, based on the observed two-way delay of 160 ms.

or $\Delta_T^{min} = 2 \cdot 70 = 140$ms. The maximum round trip is equal to twice the transmission delay, plus the maximum time that can pass on board the Delfly between receiving and sending (= 50 ms) and the maximum time between subsequent read action at the ground station (= 20 ms). This is equal to $\Delta_T^{min} = 2 \cdot 70 + 50 + 20 = 210$ms. Now, if we send a position message at 12.5Hz (i.e. every 80 ms), and we look at the time passed between receiving two subsequent acknowledgments, this can vary between $(140 + 80) - 210 = 10$ms and $(210 + 80) - 140 = 150$ms. Note that in both cases the time delay between the actual sending of the position and receiving this message on the Delfly is at most $70 + 10 = 80$ms.

The results of looking at the acknowledgments time interval is therefore not very informative of the true performance of the communication. A time stamp should be included in the position message and corresponding acknowledgments, to get a better idea of the regularity of the arrival of position messages.

Ninety percent of the acknowledgments were received within 140 ms of each other. Combined with the fact that, after counting the number of outgoing messages and received acknowledgments, the drop rate is 0%[1], the connection was reliable at the rate it was used. That being said, the performance of the Bluetooth is really abismal, considering only about 2% of the bandwidth could be used. This issues needs investigation in the future, becaue a lot of performance can be gained.

B-4 smartUAV

The tracking algorithm and Bluetooth communication was implemented in smartUAV, a developed software package developed in-house by the MAVLAB. The software works exactly like Simulink, but is completely geared towards real time implementation of controllers for UAV's.

[1]The fact that 100% of the messages sent also arrived is another indication that the Bluetooh was not rejecting packages, but kept resending the message until it arrived. Again, this is not behavior that is desirable for the type of message sent.

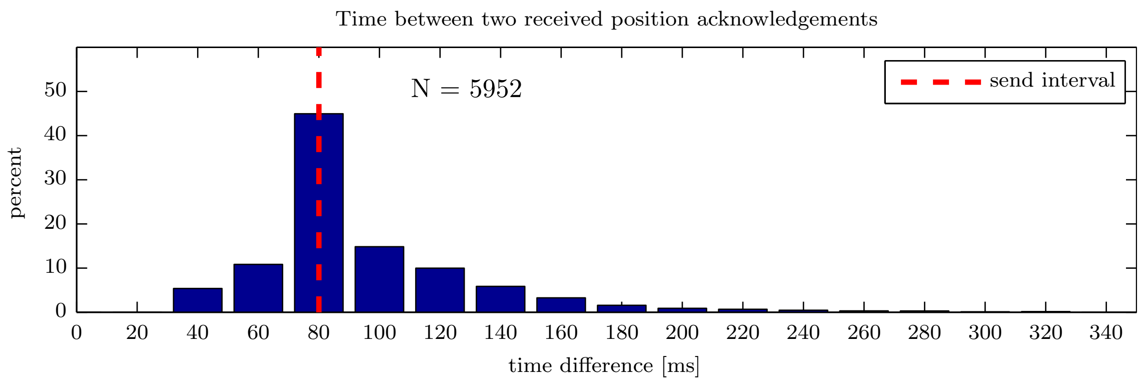


Figure B-2: A histogram of the time interval between two subsequent acknowledgements of the Delfly indicating it had received a new position update.

In Figure B-3 the block diagram is shown that is used during the experiments. A description is given explaining the function of each block.

modSleep Controls the rate at which the software runs, and was set at 50Hz for the experiments.

WiiPos This is the implementation of the position determination algorithm. This module connects to two WiiMotes, and reads the calibration values from the three text files in the smartUAV folder:

`CalibrationParameters.txt` Contains the intrinsic calibration matrices for both WiiMotes.

`CalibrationParametersStereo.txt` Contains extrinsic parameters R and $\mathbf{T}$.

`CalibrationParametersInertial.txt` Contains the inertial calibration parameters.

XYSlide This block can be set to match the wind tunnel velocity.

Butterworth The Butterworth filter applied to the discrete derivative of the position signal.

Differntiator Outputs the difference between the current input and the input of the previous time step. It does not scale the value with the loop rate. This is scaling is done in the true implementation, but is not shown in this diagram for clarity.

Rising edge switch Makes sure that when the position signal is regained after a loss of the tracking LED, the velocity signal is only passed if it is based on two new measurements. This removes the spikes in the velocity signal is these cases.

Joystick Source of the joystick commands. The signals x,y,z and th are scaled with 127, but these blocks are omitted for clarity.

WiiPilotDriver Sends and receives data through the Bluetooh connection with the Delfly. It also creates a log file, named `###wiipilotFF`, that contains every command a user has given, like changing a gain, changing control mode or issuing a step input, together with the exact time it happened. It can send at a different frequency than the main thread.

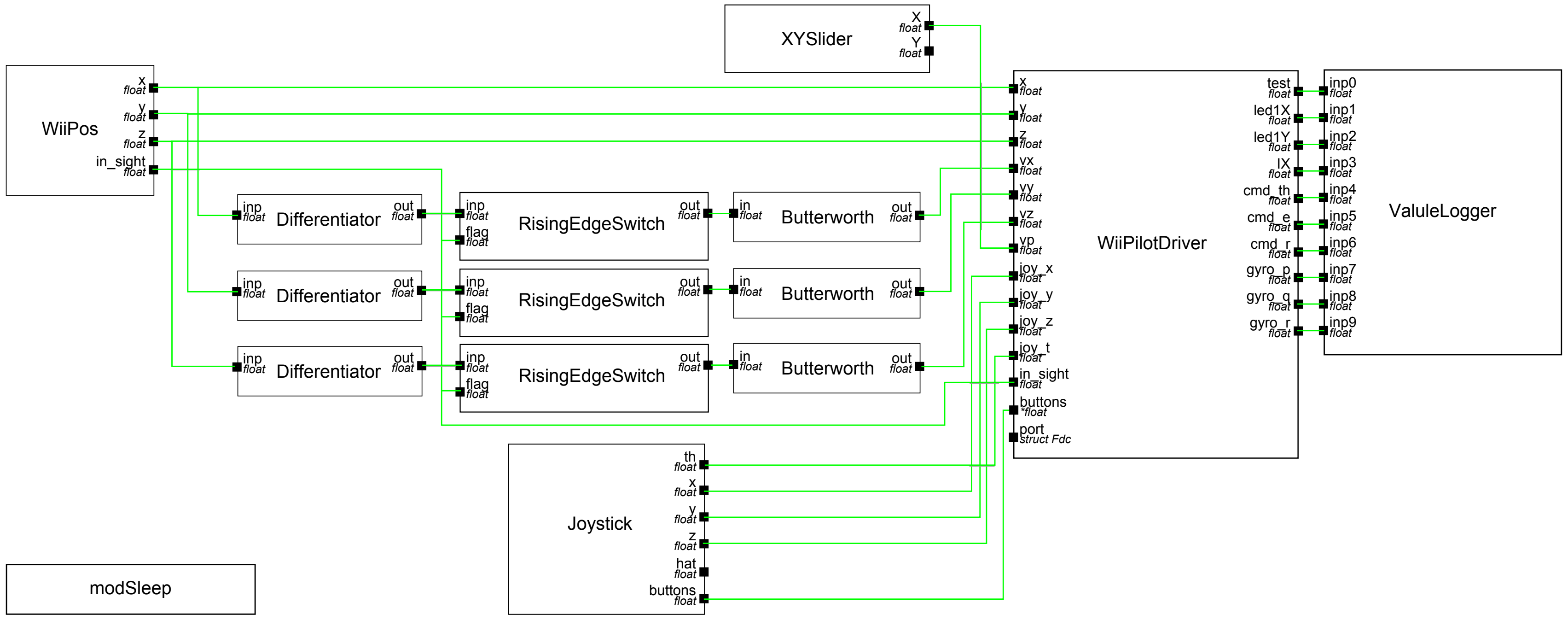


Figure B-3: The implementation of ground system in smartUAV.

When this skip_turn function is set, the block will continue to read the Bluetooth buffer at normal frequency, so that this information is read out timely. WiiPilot driver also only sends out a position and velocity update when the Delfly is actually in sight of the tracking system.

ValueLogger Logs all values that are received from the Delfly, together with the time stamp. The Delfly state is logged in a file named `###DelflyFF`. Another `ValueLogger` is used to log all data from the ground station, i.e. position, velocity and joystick commands, and produces a file called `###groundFF`. This second logger is not shown here for clarity purposes.

Appendix C

Attitude determination

At first the controller was of a more complex design which required full attitude information. In the end a more simple controller was used that directly uses the camera information, without actually calculating the angles. But the algorithm for calculating the attitude from camera observations is still outlined below, for the purpose of post processing. This way the exact attitude of the Delfly can be calculated afterwards on ground.

The attitude is determined by looking at a fixed IR LED in a known position. The principle is shown below for the 2D case of determining the pitch. Note that this algorithm calculates the attitude of the camera, not the Delfly body axes.

The pitch angle θ is the sum of the apparent pitch α and the geometric pitch ν which is calculated by

$$\nu = \tan^{-1}(\tfrac{b}{a})$$

where a and b are the distance from the on-board camera to the reference LED in the wind tunnel, in resp. x- and z-direction:

$$\begin{aligned} b &= Z_{LED} - z_w - z_{tl} \\ a &= X_{LED} - x_w - x_{tl} \end{aligned} \tag{C-1}$$

where Z_{LED} and X_{LED} are the z and x coordinate of the reference LED, x_w and z_w the position of the Delfly, and x_{tl} and z_{tl} the distance from the tracking LED to the on-board camera, which is a correction for the off set between the point that is tracked, and the actual position of the camera.

The apparent pitch α is calculated from the observation of the reference LED

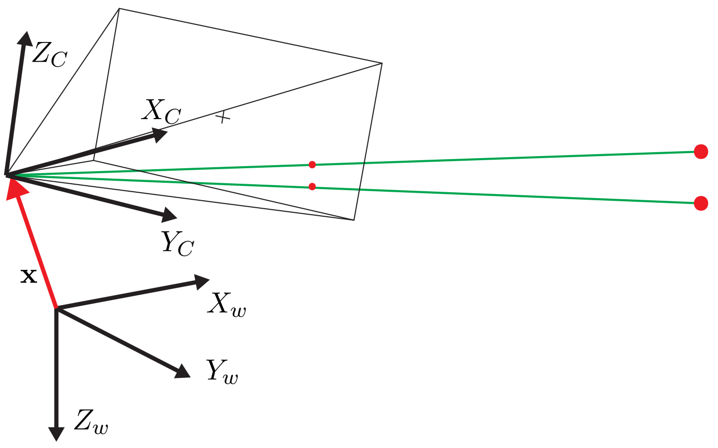


Figure C-1: The inertial wind tunnel reference frame $\mathcal{F}_w$ and the camera reference frame $\mathcal{F}_C$ at location **x**. The on board camera is looking at two LEDs placed in front of the camera. The green lines indicate the line-of-sight of the two LEDs whose projection on the camera is also shown as two red dots.

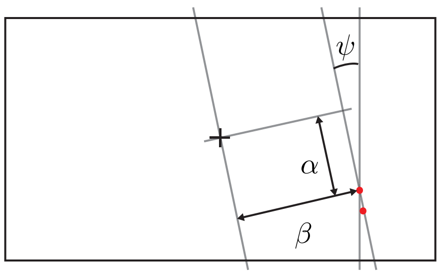


Figure C-2: Camera projection of two points, with the attitude angles indicated. For small angles, the effect of a yaw angle on the measured pixel coordinates can be ignored.

$$\alpha = s_x(x_{obsv} - c_x)$$

where s_x is the vertical *pixel pitch* (the angle one pixel represents) and c_x is the center of the camera image in x direction. θ can thus be calculated by

$$\theta = s_x(x_{obsv} - c_x) + \tan^{-1}(\tfrac{b}{a})$$

Extending this to the roll angle ϕ we get

$$\begin{aligned} \mu &= \tan^{-1}(\tfrac{g}{a}) \\ \beta &= s_y(y_{obsv} - c_y) \\ \phi &= \mu + \beta \end{aligned} \tag{C-2}$$

where μ is the geometric angle, g the y position of the camera in $\mathcal{F}_w$, β the roll angle, and ϕ the roll angle.

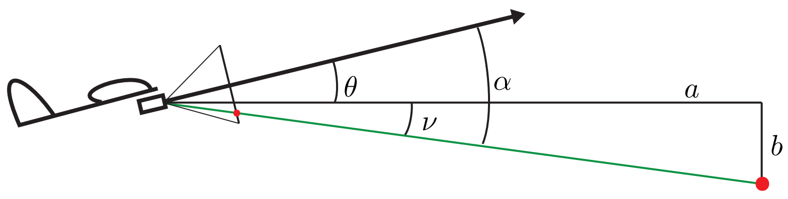


Figure C-3: Side view for angle determination.

For the yaw angle one LED is not sufficient. Therefore a second IR LED can be placed directly below the other. Now the yaw angle follows from the relative position of the two LEDs in the camera view.

$$\psi = \tan^{-1}(\tfrac{y_2-y_1}{x_2-x_1})$$

During horizontal flight the on-board camera is no longer looking forward due to the large change in pitch, and is therefore not able to look at the reference LED positioned in the middle of the wind tunnel for a heading reference. The camera is pointing down at an angle of about 30 degrees from the vertical. To resolve this, two LED's are placed on the ground during forward flight. The same algorithms apply in this case, but the change in position of the reference LEDs needs to be taken in to account.

C-1 Effects of the time delay on attitude estimation

The position information is send up through the Bluetooth, and that introduces an up link delay of about 73 ms on average. Because the attitude is calculated as a function of position, a delay in the position signal causes a miscalculation. If the Delfly has a high velocity, say $0.5ms^{-1}$, it moves $0.073 \cdot 0.5 = 0.037m$ in the time the position signal arrives. The angle error η_{err} this introduces is equal to

$$\eta_{err} = \tan^{-1}\frac{z}{x} = \tan^{-1}\frac{0.037}{2.0} \approx 1.06^{\circ}$$

for a nominal distance to the reference LED of 2 m. During stationary flight these velocities do not occur, but during maneuvers like step inputs describes above, velocities indeed do reach up this value. Figure C-4 shows the results of post-processing the camera sighting and position data. There is a clear relationship between the velocity and the error caused by the time delay.

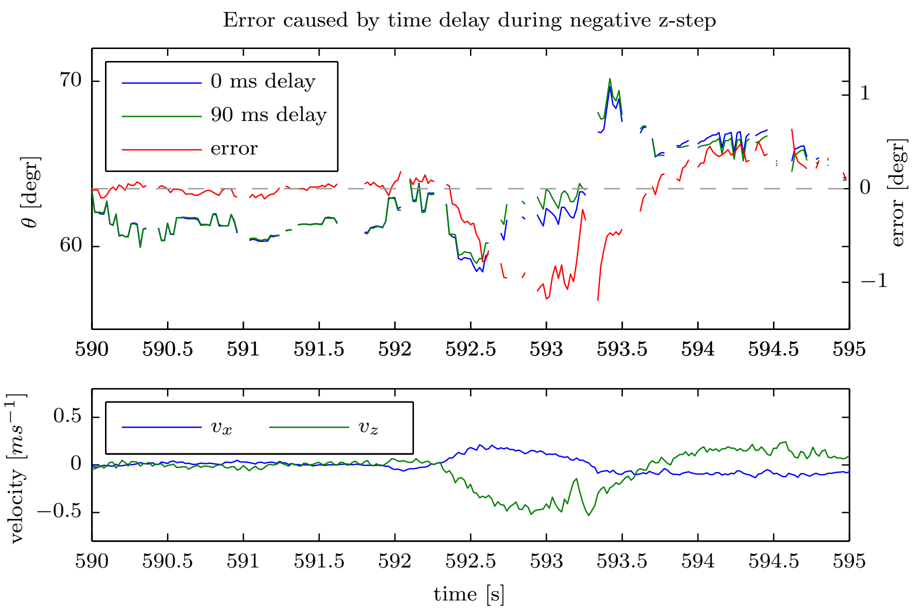


Figure C-4: Error in theta estimation caused by a delay in the position signal. At low velocities the effect is not large, but when doing a vertical step at $0.5ms^{-1}$ the error can reach over 1 degree.

Appendix D

Preliminary test results

D-1 Quadcopter

Most subsystems were first tested on a less fragile platform than the Delfly to see whether or not the designed system would work as required. The Parrot ARDrone, a commercial quad copter produced by Parrot S.A. shown in Figure D-1, was used for this purpose. This platform is used for a variety of tasks by the MAVLAB, and has the advantage that it is already interfaced in the software environment smartUAV that was used for this project. It can easily be controlled by giving it pitch, roll and yaw commands owing to its inner-loop control.

Figure D-1: The Parrot ARDrone.

A copy of the auto pilot that was used for the Delfly was mounted on the top of the quad copter. It could sent the pitch, roll and yaw commands to the quad through a serial interface. This way the communication and other subsystems of the auto pilot could be tested without having to work with the Delfly. An IR LED was mounted on the underside so it could be tracked by the tracking system.

Apart from providing validation of performance of the various subsystems, it would also

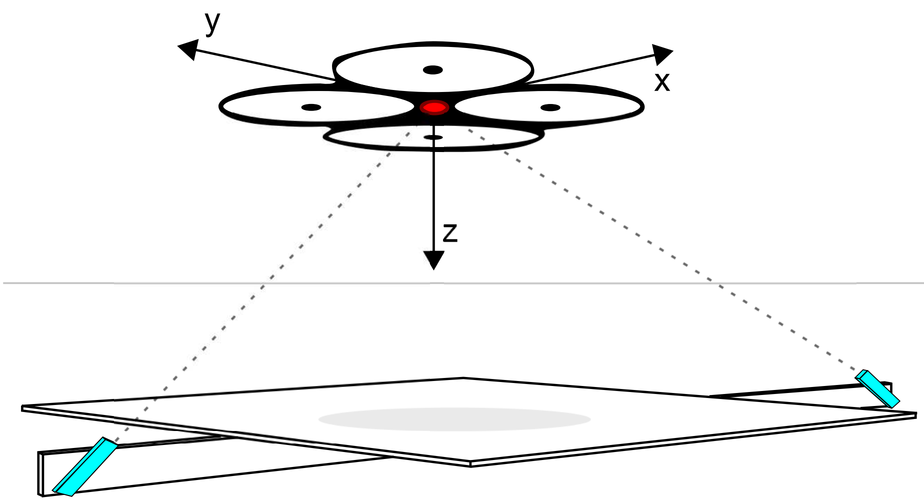


Figure D-2: Schematic drawing of the hover test setup. The quad copter hovers above a flat plate, the attached IR-LED (red) in sight of the two tracking cameras (light blue).

provide valuable experience in flight testing, data acquisition and processing and allow the early discovery of flaws in the system.

During the first hover test, the quad copter was controlled fully from the ground using the built in WiFi control. The flight test data is presented in the section below. The forward flight test was conducted using our custom autopilot, which would receive state information from the ground station (position and velocity) and do the control on-board. The communication was done via Bluetooth.

D-1-1 Hover

The testing of the tracking system was done using the quad copter. The AR Drone was placed on a plate lying over the support for the tracking system. This was done to have a flat underground for the sonar of the drone. The drone was then given the command to lift off, and manually steered to the center of the box. As soon as the drone was in the middle of the virtual box of the tracking system, the auto-pilot was engaged.

This preliminary test was done without a external reference for the heading. Instead, it relied on the inner loop control of the drone keeping the nose in the right direction for the relatively short time duration of the test flight. The gyroscopes of the autopilot were not used for doing this inner loop. The reason for this is that the quad copter uses its own inner loop structure, and it was very hard to circumvent this controller. Also, the Delfly is a completely different platform so implementing and testing this inner loop controller would take a lot of time, producing results not readily transferable to the Delfly. Therefore, tests for the gyro's for the inner loop will be left for the Delfly, therefore spending as little time as possible on the quad copter.

The controller block (see Figure D-3) used the following equations to generate the commands for the quad copter

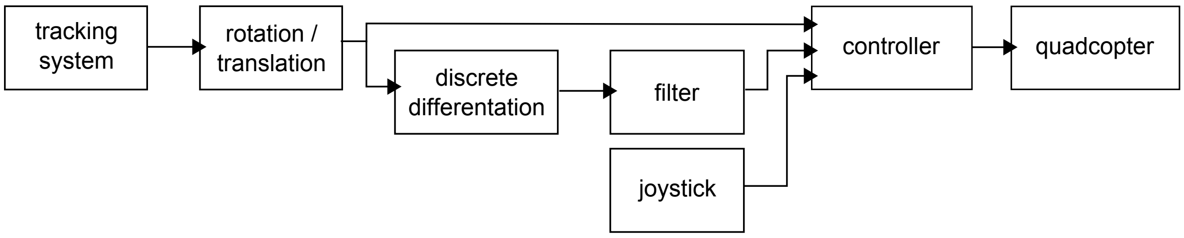


Figure D-3: Flow diagram of the ground station and the quad copter. The command signals are generated on ground and send to the quad copter by WiFi.

$$\begin{aligned}
\delta_\theta &= K_{ux}u_x + K_x(x - x_{ref}) + K_{vx}v_x \\
\delta_\phi &= K_{uy}u_y + K_y(y - y_{ref}) + K_{vy}v_y \\
\delta_t &= K_{ut}u_t + K_z(z - z_{ref}) \\
\delta_\psi &= K_{uz}u_z
\end{aligned}$$

With the position x, y, z in meters, velocities v_x, v_y in ms^{-1}. Joystick input u_x, u_y, u_z, u_t was between -1 and 1. The gains were as indicated in Table D-1.

gain	value	gain	value	gain	value
K_{ux}	1	K_x	0.11	K_{vx}	13
K_{uy}	1	K_y	-0.11	K_{vy}	-13
K_{uz}	1	K_z	0.2		
K_{ut}	1				

Table D-1: Gains during the hover test flight

The positive sign of the x and z direction has to do with the definition of the reference system. As can be seen in D-2, the positive x axis points in negative pitch (nose down) direction, hence the sign reversal. The same holds for the thrust: more thrust is an decrease in z position, therefore a positive thrust command is correcting for a positive vertical position error (i.e. when the quad is too low). The values of these gains were established empirically.

The test flight proved to be successful, as it succeeded in keeping the quad at the center of the tracking box, within 20 cm of the reference point, see Figure D-4. But the position keeping was far from perfect. It was suspected that this was mainly due to large disturbances caused by the induced airflow reflecting of the many objects in the test lab (a later test in a more spacious room with less objects around, showed a significant increase in performance.)

D-1-2 Moving platform

For initial testing of the systems a moving platform was designed and build, allowing to test forward flight without having to rely on the wind tunnel availability.

A photo of the platform is shown in Figure D-5. The rear wheels are larger to accommodate for a aluminum disc used for the velocity measurements. In front is a wooden structure

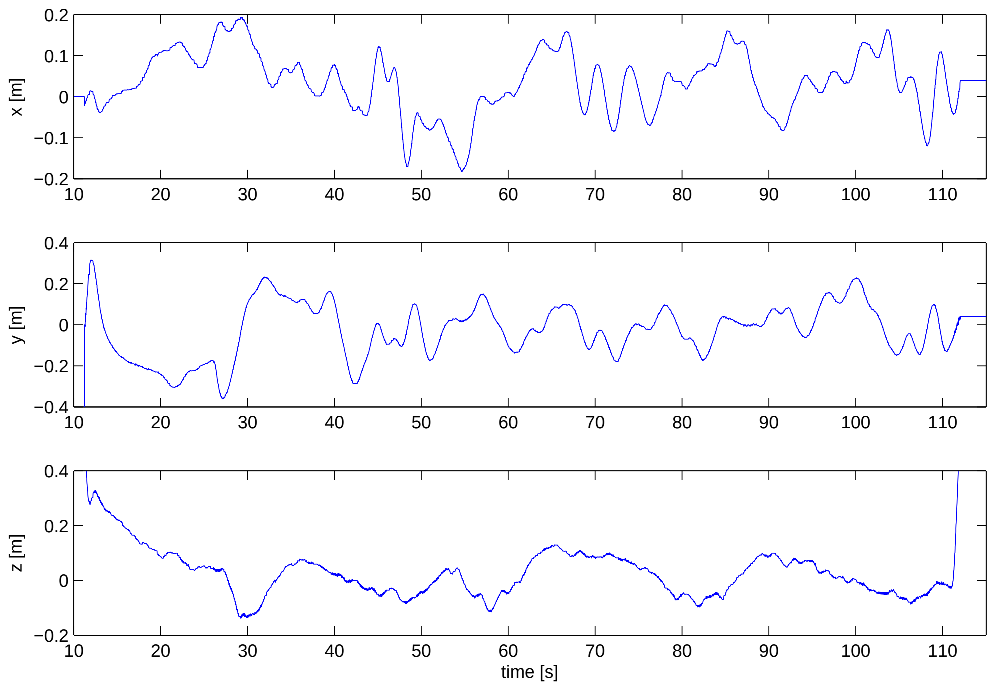


Figure D-4: Position data of the hover test flight.

that allows tethering of the Delfly, so it can be tested without the danger of crashing. This provides also a mounting point to attach a reference LED to, needed as a heading reference for the Delfly. A large and heavy wooden plate is used as the basis. This server two purposes. First it provides the flat surface the quad copter needs for its sonar, and secondly it makes the platform heavier so there are less vibrations and so the platform keeps a more constant velocity when pulled along.

For the velocity estimate an aluminum disc was cut out using a CNC milling machine, with a radius of 104.5 mm. It has 150 evenly spaced holes around the perimeter, each 2 mm wide and 2 apart. An optocoupler was mounted around this disc, which is basically a light source and a light sensitive receptor. It can count the time interval between each passing hole, therefore providing a way of measuring the speed of the platform. This is necessary because the inertial speed of the aircraft flying above it needs to be known. Figure D-6 shows a photo of the disc mounted on the moving platform, along with the microprocessor used to process the data from the optocoupler. The microprocessor relays the information by serial port to the ground station, where a block in smartUAV reads out the pulse information, and translates it to a velocity.

The optocoupler is connected to a microprocessor that counts the number of clock ticks (running at 250 kHz) between each subsequent hole, and this tick count n is provided read out in smartUAV. The velocity of the platform v_p in ms^{-1} can be expressed by

$$v_p = 250\frac{\beta}{n}$$

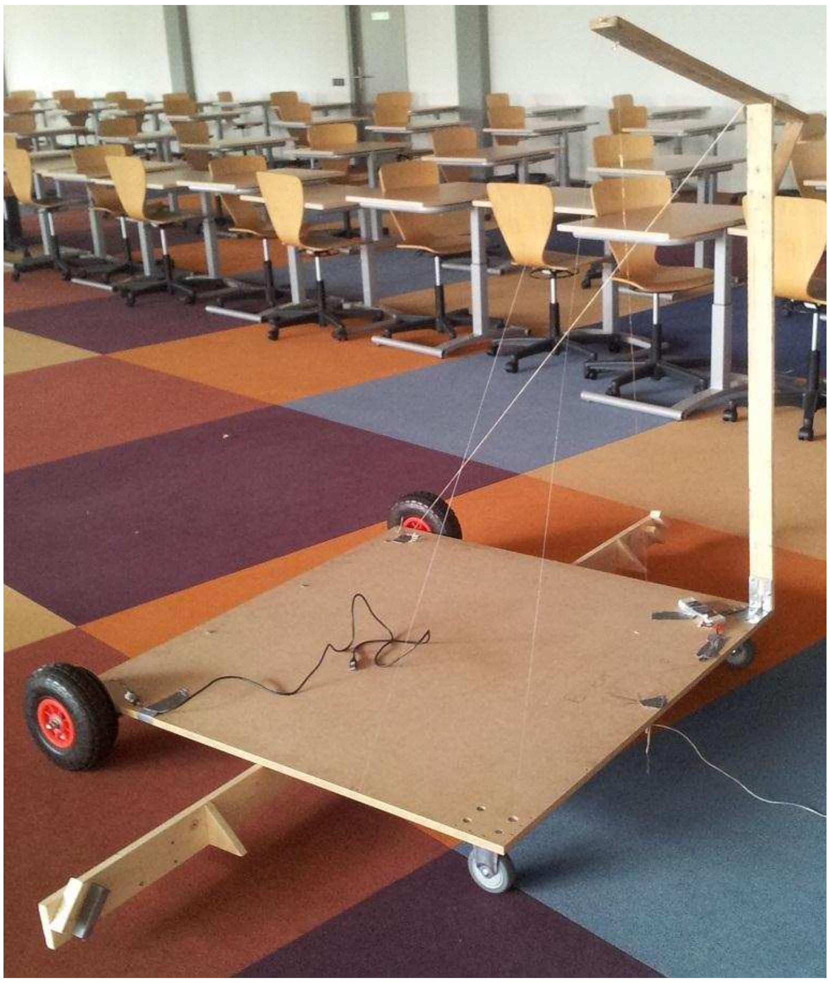


Figure D-5: The moving platform, showing the wooden support in front to attach the Delfly to by a safety tether. The larger back wheels accommodate the velocity tracking system

using the calibration parameter β which is the traveled distance in millimeters per pulse, and n the number of ticks since the last hole. β can be easily calibrated by measuring a certain length out on the ground and moving the platform this distance. β is then equal to the distance traveled divided by the number of pulses counted by the optocoupler.

Although the setup provides very accurate readings and there is very little noise at lower speeds, testing showed that at higher velocities filtering was necessary. Raw measurements were taken and a filter was designed in MATLAB, see Figure D-7. A butterworth filter of 2nd order and a time constant of 0.04 is used. The filter is able to suppress the noise sufficiently, but with some delay, ~ 0.12 second. This can be allowed however because the platform will be used at a relatively constant velocity, so response time is not that important.

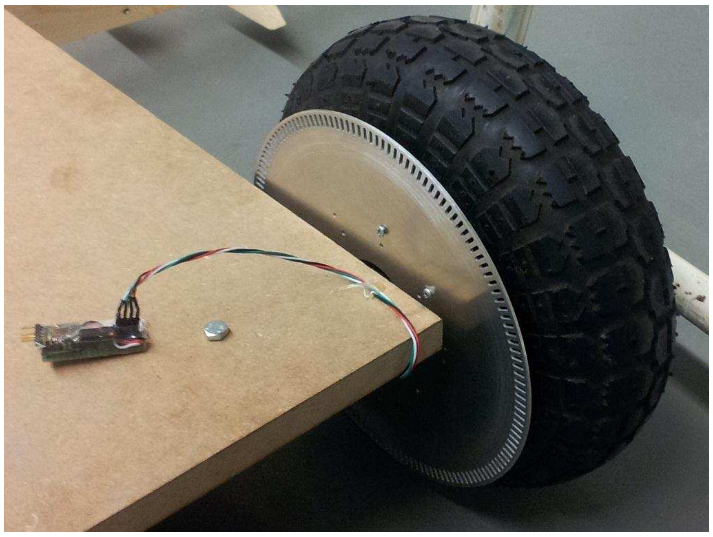


Figure D-6: The microprocessor and aluminum disc mounted on the wooden moving platform. The optocoupler, not visible in this photo, is mounted on the front side of the wheel, underneath the platform.

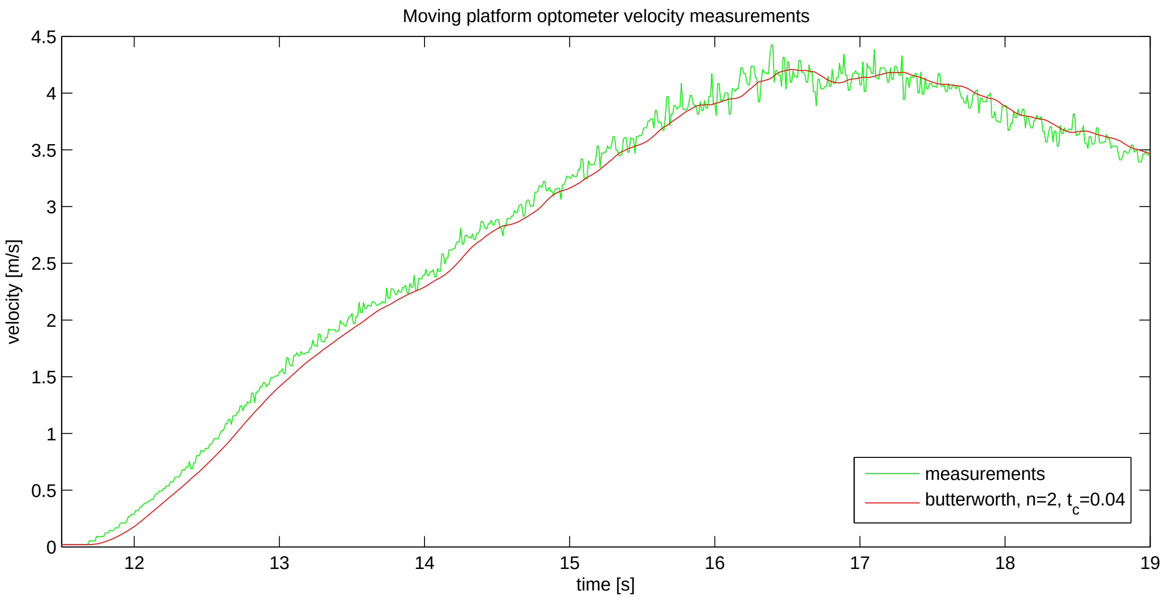


Figure D-7: Velocity measurements of the platform using the optocoupler.

D-1-3 Forward flight

A forward test flight was also performed. This test was done in order to see how well the same setup would function on the moving platform. The tracking cameras are extended quite far from the platform, and thus could oscillate with the vibrations of the wheels, degrading tracking performance especially for the velocity estimation.

In this test, the ground station sends position and velocity information to a on-board autopilot, which would in turn calculate the pitch, roll, yaw and thrust command for the quad copter. Also the information about the velocity of the platform is send up. See Figure D-8 for the flow diagram of the ground station.

The command signal is generated on-board of the autopilot, which was attached to the top of the quad copter, on the basis of the received position of the quad copter (x, y, z) and velocities of the quad copter (v_x, v_y, v_z) and the platform velocity (v_p). This position and velocity information was sent through bluetooth.

The equations used to calculate the command signal are shown below. It can be seen that the controller is just a PD position controller.

$$\begin{aligned}\delta_\theta &= K_{ux}u_x + K_x(x - x_{ref}) + K_{vx}v_x + K_{vp}v_p \\ \delta_\phi &= K_{uy}u_y + K_y(y - y_{ref}) + K_{vy}v_y \\ \delta_t &= K_{ut}u_t + K_z(z - z_{ref}) \\ \delta_\psi &= K_{uz}u_z\end{aligned}$$

The extra term on the right hand side for δ_θ accounts for the velocity of the platform. This acts as a feed forward term correcting for the moving platform. The reference position $\begin{bmatrix} x_{ref} & y_{ref} & z_{ref} \end{bmatrix}^T$ is equal to zero, so these terms drop out. Also, because the vertical position is 1st order and not 2nd order, the velocity term was omitted. The values of the gains used are shown in Table D-2. The values of the commands were limited to ±100. This value was empirically established as the bounds where the quad copter would be agile enough whilst not being too aggressive.

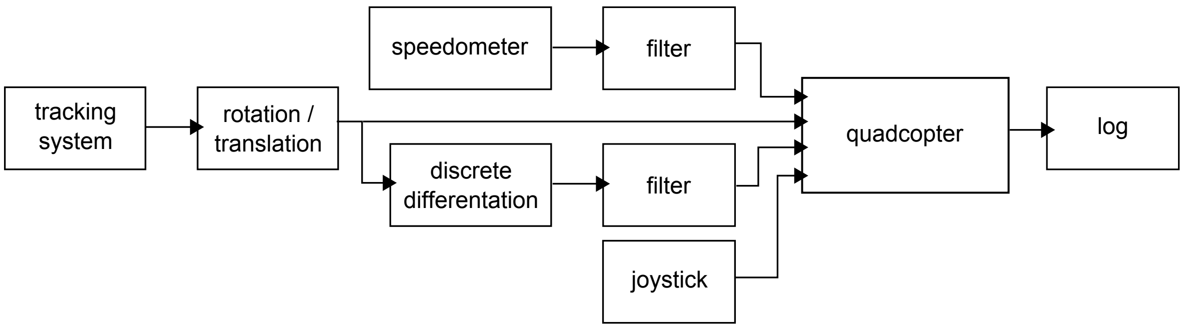


Figure D-8: Ground station flow diagram for the forward-flight test.

The gains for the velocity were set at 12.7 and not at 13 like during the hover test. This was because of the way the gains were sent to the autopilot. The gains are only 8-bit, therefore only allowing values between -127 and $+127$ (a scaling factor of 10 was applied).

The position data for the forward-flight test is shown in Figure D-9. For the first 13 seconds, at a velocity up to 1 m/s the quad copter was following the platform reasonably well, with only slightly worse performance than during the stationary hover test. It can be seen that when the platform is speeding up, the quad copter has troubles catching up, and it shows a slight overshoot when the platform is again slowing down. This effect is especially apparent at higher velocities. Although the quad copter is able to follow the acceleration at $t = 203$, at $t = 205$ it starts lagging behind, leaving the tracking box completely at $t = 207$. Because the last position is held when leaving the box (which is the edge of the tracking box), slowing the cart down makes the quad copter enter the box quite soon again. But it is somehow unable to limit its forward velocity in time, and it overshoots very quickly, staying there until the platform came to a complete stop. It is not exactly clear what happened here.

Although it was not a complete success, this preliminary forward-flight test did successfully demonstrate the viability of:

- using bluetooth for two-way communication
- the tracking system mounted on the moving platform
- measuring the inertial velocity of moving platform
- controlling a flying platform on the basis of above information

gain	value	gain	value	gain	value
K_{ux}	1	K_x	0.1	K_{vx}	12.7
K_{uy}	1	K_y	-0.1	K_{vy}	-12.7
K_{uz}	1	K_z	0.2	K_{vp}	-0.1
K_{ut}	1				

Table D-2: Gains used for the forward-flight test.

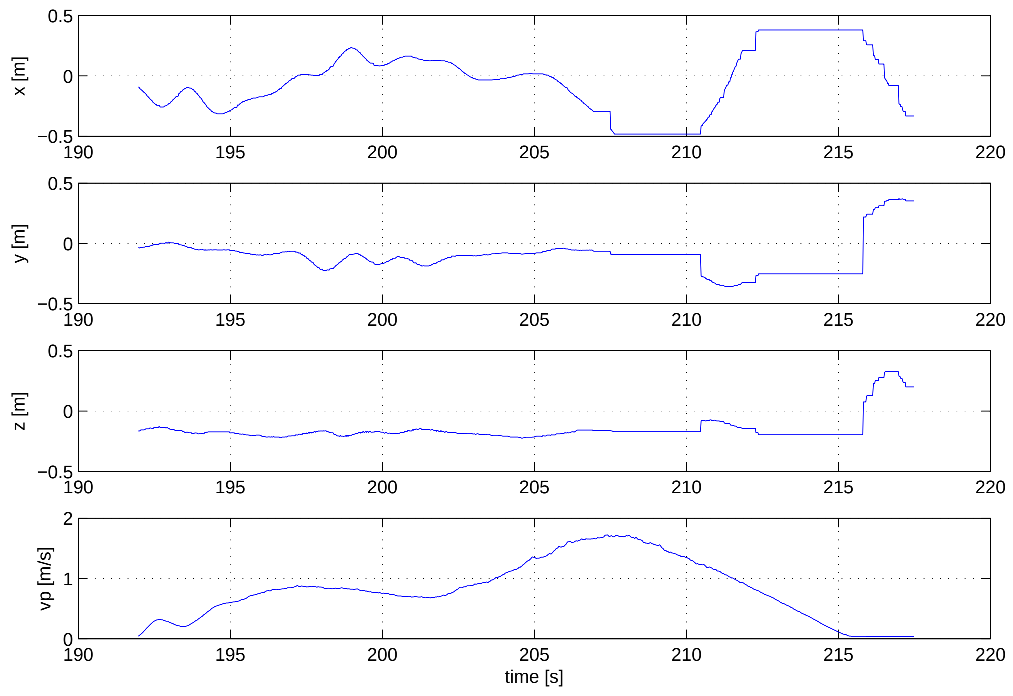


Figure D-9: Position data for the forward-flight test, with the fourth graph showing the velocity of the moving platform. The quad copter left the box twice (where the position signal goes flat), one time it lagged behind, the other time it overshot.

D-2 Delfly Moving Platform

The Delfly was tested above the moving platform, to see whether all systems would work, and to acquire experience in flight testing the Delfly and to see where improvements needed to be made. It turned out that the moving platform was not a good way to flight test the Delfly. The combination of a too small viewing angle of the on-board camera (only 33 degrees in horizontal direction) and quite some turbulence, meant that the Delfly would loose There was too much turbulence and only a few times was the Delfly able to follow the LED for a while.

Therefore, the moving platform tests for the Delfly were no success. It did allow small bugs to be worked out, and it gave the opportunity for a good dry run and several improvements were made to the software. For example during these tests the value of having information about in what kind of state the Delfly was operating (for example whether or not it was seeing the LED, or whether it had indeed received a position update) was discovered and this functionality was subsequently added.

D-3 Conclusions

The use of the moving platform was less than expected, but in general the preliminary tests with the quad copter have provided a very good preparation for the final wind tunnel tests. It allowed to test all the hardware, especially communication and the software, and find flaws in the setup early on, without wasting precious wind tunnel time.

Although the tests with the quad copter gained valuable lessons, a disproportionate amount of time was spent to interface the auto pilot PCB with the ARDrone software.

The experiences with the platform learned that it does not provide a good platform for flight tests with the Delfly. The Delfly is too susceptible to turbulence, and the moving platform can not provide the required low turbulence conditions. Also, the OJF proved such a formidable testing area, where, in combination with the live tuning possibilities of the GUI in smartUAV, a lot of testing can be done in little time.

Appendix E

OJF Wind tunnel test results

E-1 Test run data

This appendix gives an overview of all the tests that are performed, and give an overview of the data collected for each of these tests.

Definitions

Three log files were created for each flight. They each contain a time stamp in the first column, but the content of the other columns varies. The variables are shown in the table below.

File	1	2	3	4	5	6	7	8	9
delflyFF	t	led1X	led1Y	IX	cmd_e	cmd_r	cmd_th	p	q
groundFF	t	x	y	z	vx	vy	vz	vp	flag
WiiPilotFF	t	FMS	Kx	Ky	Kz	Kvx	Kvy	Kvz	Kphi
File	10	11	12	13	14	15	16	17	
delflyFF	r	theta							
groundFF	joy_x	joy_y	joy_th						
WiiPilotFF	Ktheta	Kpsi	Kp	Kq	Kr	Kk	Km	Kn	

Table E-1: Log file column definition

The following values are of importance for each test run. If the value of a variable is different for each test run, it is indicated in the tables below.

Z_{calib} Height of the origin of the wind tunnel reference frame above the platform. This was calibrated at 1.100 meter.

X_{LED} The distance of the LED from the origin of the wind tunnel reference frame. This was measured to be exactly 2.000 meter

Z_{LED} The height of the reference LED above the platform. Changed somewhat in between tests. It is indicated per test.

smartUAV rate The rate at which the smartUAV was run, almost always at 50 Hz. This is set in the modSleep module.

send rate The send rate was different from the main loop speed to reduce the communication loading. It was usually run at $^1/_4$ the speed of the main loop.

downlink rate The downlink rate as set on the Delfly. Usually 20 Hz, or $^1/_5$ of the main loop speed (which was 100 Hz).

wind velocity The wind velocity in the tunnel at the time of the measurement. The wind tunnel velocity was not very accurate, so it could deviate about 0.1 m/s.

tail angle The tail angle was changed when the forward flights began. The angle indicates the angle between the horizontal tail plane and the fuselage. A negative angle means that the TE of the horizontal tail plane is higher than the LE.

type of battery Always the 180 mAh battery were used. The weight is given in the weight breakdown in A-2.

FMS Flight management system. A value of 5 is none, 6 is autopilot mode and 7 is manual flight.

IX Contains flags pertaining to the Delfly's state. Each bit has a meaning:

1. not used
2. IX_FLYING_BLIND
3. IX_LED_IN_SIGHT
4. IX_MOD_NONE
5. IX_MOD_AP
6. IX_MOD_MANUAL
7. IX_FRESH_POSITION

Roll and gyro tests

Wind velocity	0.5	ms^{-1}	smartUAV rate	50	Hz
Tail angle	-2	°	send rate	12.5	Hz
Ref. led height	703	mm	down link rate	20	Hz
Date	21/5		time	17:18	
Folder	OJF gyro test		number	1337613564	
Side photo	0097		Movie	MVI_0098 & 0099	

Table E-2: Tried several combinations of K_p and K_r. Use of the K_r gain makes that the K_p can be put higher before instabilities occur. Movies are not very useful.

Wind velocity	0.5	ms^{-1}	smartUAV rate	50	Hz
Tail angle	-2	°	send rate	12.5	Hz
Ref. led height	703	mm	down link rate	20	Hz
Date	21/5		time	14:33	
Folder	OJF gyro test		number	1337603862	
Side photo	0088		Movie	-	

Table E-3: Longer runs were performed this time. Several different values were used for the p gain. There was hardly any difference visible between different K_p settings.

Wind velocity	0.5	ms^{-1}	smartUAV rate	50	Hz
Tail angle	-2	°	send rate	12.5	Hz
Ref. led height	703	mm	down link rate	20	Hz
Date	21/5		time	14:51	
Folder	OJF gyro test		number	1337604773	
Side photo	0089		Movie	-	

Table E-4: Manual trimming of the thrust and rudder. Tested a K_p gain of 12, and 15 (unstable).

Wind velocity	0.4	ms^{-1}	smartUAV rate	50	Hz
Tail angle	-2	°	send rate	12.5	Hz
Ref. led height	703	mm	down link rate	20	Hz
Date	21/5		time	15:27	
Folder	OJF gyro test		number	1337607198	
Side photo	0091		Movie	-	

Table E-5: Tried a lower K_x gain, then changed the K_p gain. It seemed to improve performance with the gain. Thrust was constantly manually trimmed.

Integrator tests

Wind velocity	0.5	ms^{-1}	smartUAV rate	50	Hz
Tail angle	-2	°	send rate	12.5	Hz
Ref. led height	703	mm	down link rate	20	Hz
Date	21/5		time	16:00	
Folder	OJF integrator		number	13377609835	
Side photo	0093 & 0094		Movie	MVI_0093	

Table E-6: Three movies were made with a smart phone as well. (2012-05-21_16-40-11, 16-35-35 and 16-35-04). This is an very long flight, of about 900 seconds.

Wind velocity	0.5	ms^{-1}	smartUAV rate	50	Hz
Tail angle	0	°	send rate	12.5	Hz
Ref. led height	696	mm	downlink rate	20	Hz
Date	22/5		time	17:40	
Folder	OJF integrator		number	1337700754	
Side photo	0109		Movie	2012-05-22_17-34-52	

Table E-7: Wings were just repaired. Test flight of the PI controller. No gyro feedback was applied. There was a person a little bit behind the Delfly during testing, which made quite an influence on the performance.

Wind velocity	0.5	ms^{-1}	smartUAV rate	50	Hz
Tail angle	0	°	send rate	12.5	Hz
Ref. led height	696	mm	downlink rate	20	Hz
Date	22/5		time	17:49	
Folder	OJF integrator		number	1337700754	
Side photo	0110		Movie	-	

Table E-8: Tried several different gain values, especially for yaw. No person was in the wind stream, clean data.

Step inputs

Wind velocity	0.5	ms^{-1}	smartUAV rate	50	Hz
Tail angle	0	$^\circ$	send rate	12.5	Hz
Ref. led height	696	mm	downlink rate	20	Hz
Date	22/5		time	17:12	
Folder	OJF step test		number	1337704584	
Side photo	0111		Movie	MVI_0122	

Table E-9: K_p at -40, with three different thrust gains. Made several vertical step inputs, afterwards 2 more elevator steps.

Wind velocity	0.6	ms^{-1}	smartUAV rate	50	Hz
Tail angle	0	$^\circ$	send rate	12.5	Hz
Ref. led height	696	mm	downlink rate	20	Hz
Date	22/5		time	11:35	
Folder	OJF step test		number	1337684836	
Side photo	0104		Movie	MVI_0106	

Table E-10: Performed several steps in both directions. Very good data.

Wind velocity	0.4+	ms^{-1}	smartUAV rate	50	Hz
Tail angle	0	$^\circ$	send rate	12.5	Hz
Ref. led height	696	mm	downlink rate	20	Hz
Date	22/5		time	19:08	
Folder	OJF step test		number	1337707053	
Side photo	0113		Movie	MVI_0114	

Table E-11: Tried several combinations of gains, also several K_q gains.

Wind velocity	0.4	ms^{-1}	smartUAV rate	50	Hz
Tail angle	0	$^\circ$	send rate	12.5	Hz
Ref. led height	696	mm	downlink rate	20	Hz
Date	22/5		time	19:41	
Folder	OJF step test		number	1337708608	
Side photo	0115		Movie	MVI_0116	

Table E-12: Rudder step input. Gave a 100 units rudder command at the press of a button, for several combinations of K_p, K_q and K_r.

Forward flight

Wind velocity	3.0	ms^{-1}	smartUAV rate	50	Hz
Tail angle	0	°	send rate	12.5	Hz
Ref. led height	-	mm	downlink rate	20	Hz
Date	23/5		time	16.00	
Folder	OJF 3ms		number	1337780243 & 1337781453	
Side photo	-		Movie	MVI_0123 & MVI_0124	

Table E-13: The output of the down link was slightly changed. The output was LED1_y and LED2_y instead of LED1_x and LED1_y.

E-2 Step Input Results

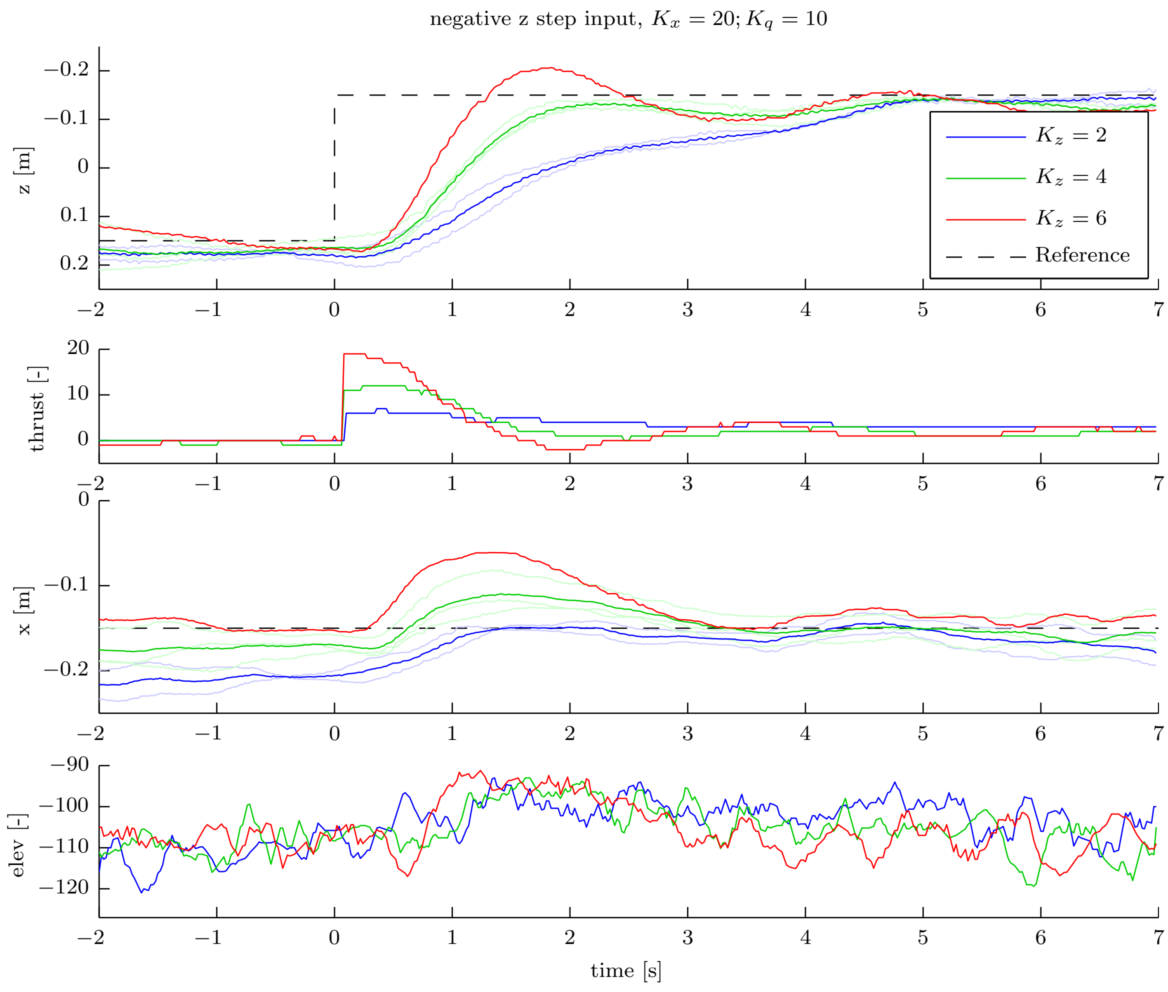


Figure E-1

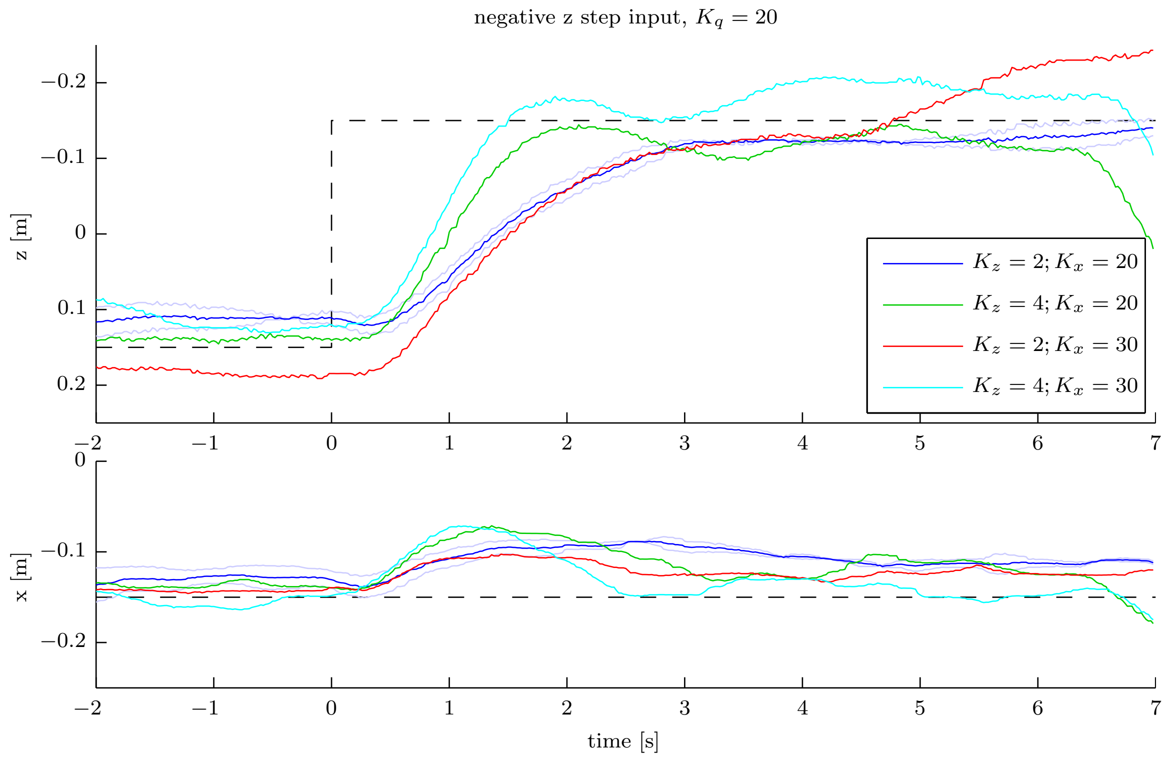

negative z step input, $K_q = 20$
z [m]
−0.2
−0.1
0
0.1
0.2
$K_z = 2; K_x = 20$
$K_z = 4; K_x = 20$
$K_z = 2; K_x = 30$
$K_z = 4; K_x = 30$
x [m]
0
−0.1
−0.2
−2
−1
0
1
2
3
4
5
6
7
time [s]

Figure E-2

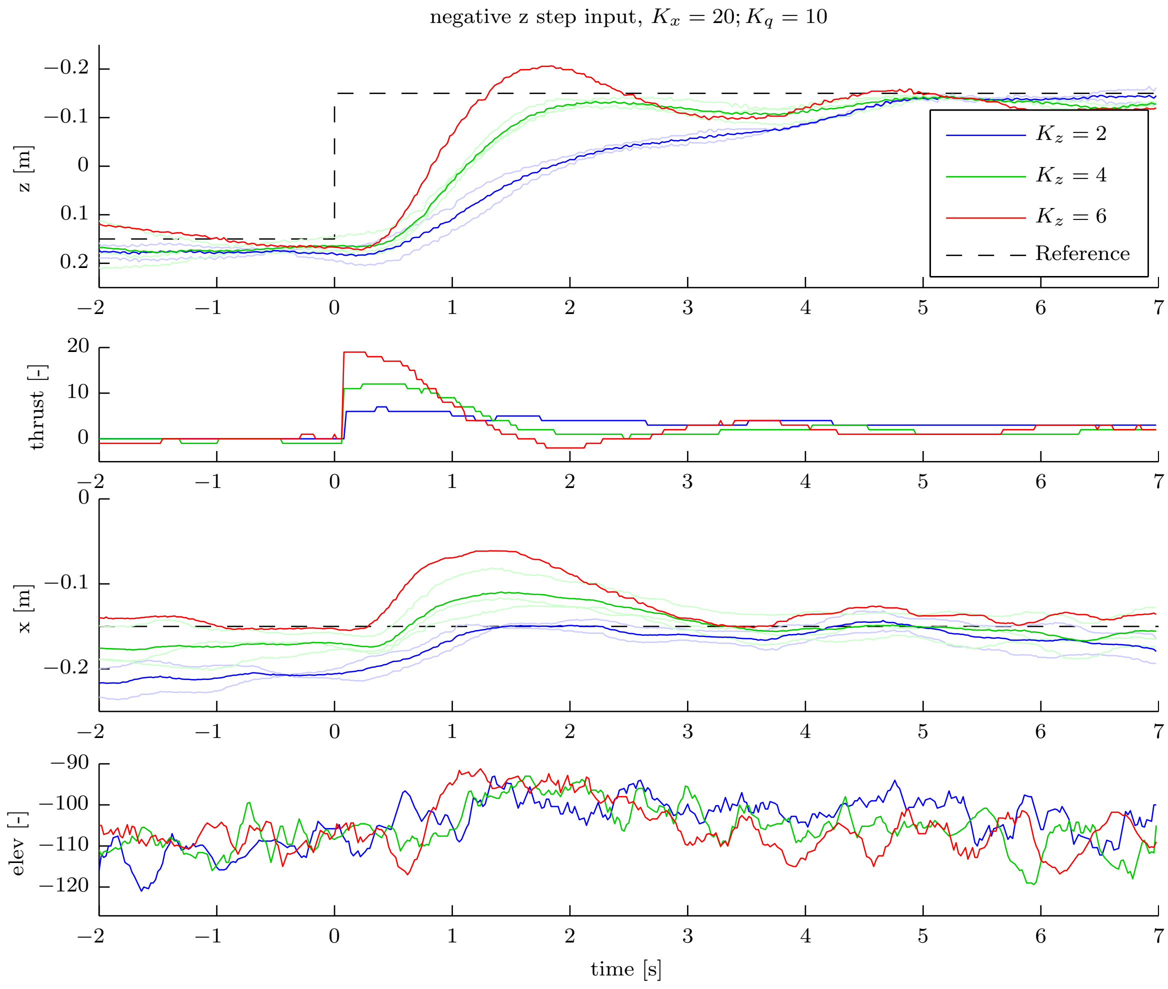

negative z step input, $K_x = 20; K_q = 10$
z [m]
thrust [-]
x [m]
elev [-]
time [s]
$K_z = 2$
$K_z = 4$
$K_z = 6$
Reference
−0.2
−0.1
0
0.1
0.2
20
10
0
−0.1
−0.2
−90
−100
−110
−120
−2
−1
0
1
2
3
4
5
6
7

Figure E-3

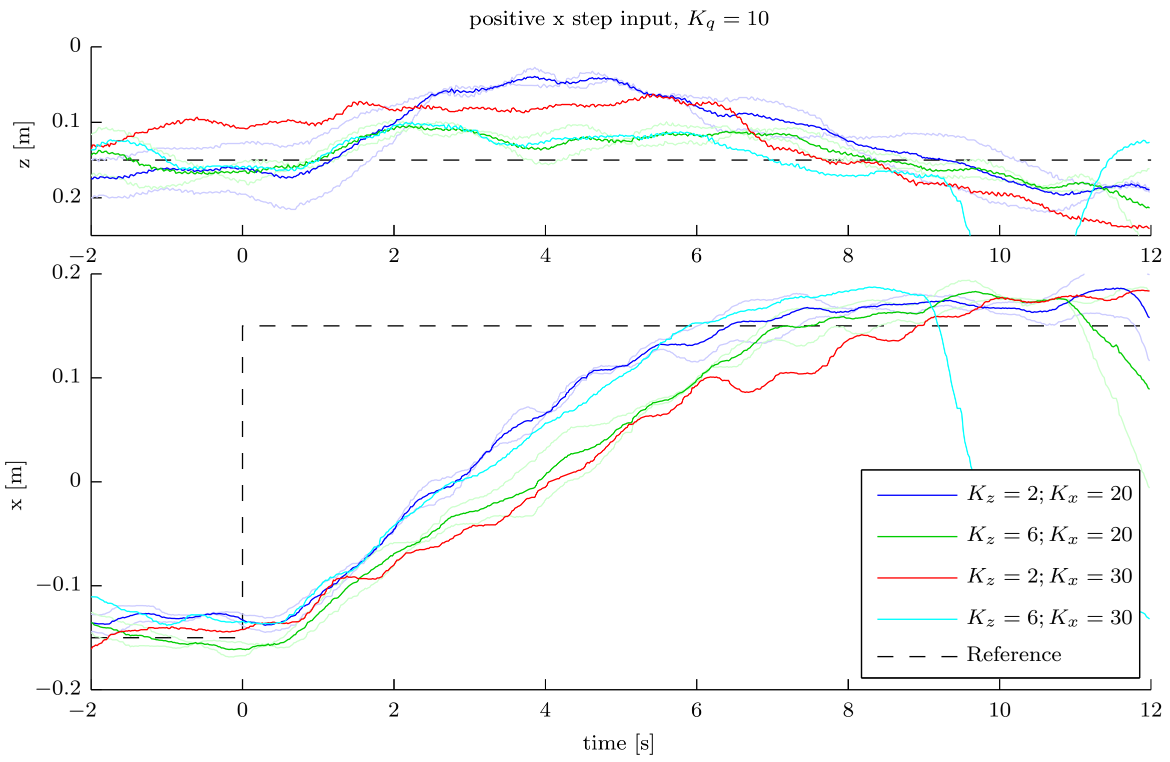


Figure E-4

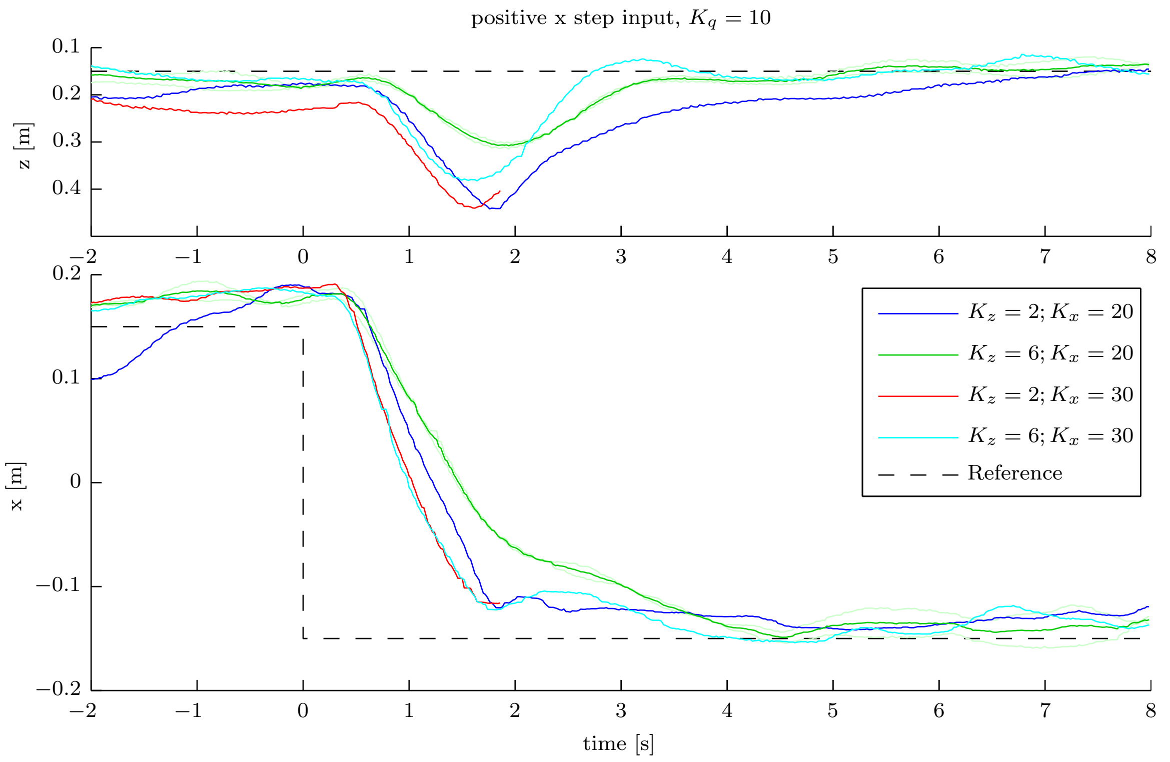


Figure E-5

Bibliography

Baek, S. S. (2011). *Autonomous Ornithopter Flight with Sensor-Based Behavior.* Unpublished doctoral dissertation, University of California, Berkely.

Biesel, W., & Nachtigall, W. (1987). Pigeon flight in a wind tunnel *. *Journal of Comparative Physiology B: Biochemical, Systemic, and Environmental Physiology*, 99–109.

Bomphrey, R. J., Lawson, N. J., Taylor, G. K., & Thomas, A. L. R. (2006, January). Application of digital particle image velocimetry to insect aerodynamics: measurement of the leading-edge vortex and near wake of a Hawkmoth. *Experiments in Fluids*, *40*(4), 546–554. Available from `http://www.springerlink.com/index/10.1007/s00348-005-0094-5`

Croon, G. C. H. E. D., Clercq, K. M. E. D., Ruijsink, R., & Remes, B. (2009). Design, aerodynamics, and vision-based control of the DelFly. , *1*(2), 71–98.

De Clercq, K. M. E., Kat, R. D., Remes, B., Van Oudheusden, B. W., & Bijl, H. (2009). Flow visualization and force measurements on a hovering flapping wing MAV Delfly II. *International Journal of Micro Air Vehicles*.

Gl.itter. (2010). *WiiYourself!* Available from `http://gl.tter.org`

Groen, M. (2010). PIV and force measurements on the flapping-wing MAV DelFly II. (December). Available from `http://www.delfly.nl/MSc-Groen.pdf`

Hartley, R. I., & Sturm, P. (1997, May). Triangulation. *Computer Vision and Image Understanding*, *68*(2), 146–157. Available from `http://www.ncbi.nlm.nih.gov/pubmed/21424191`

Hedenström, A., Muijres, F. T., Busse, R., Johansson, L. C., Winter, Y., & Spedding, G. R. (2009, February). High-speed stereo DPIV measurement of wakes of two bat species flying freely in a wind tunnel. *Experiments in Fluids*, *46*(5), 923–932. Available from `http://www.springerlink.com/index/10.1007/s00348-009-0634-5`

Nowack, J., & Alles, W. (2008). Free Flight Wind Tunnel Tests For Parameter Identification.

Pines, D. J., & Bohorquez, F. (2006). Challenges facing future micro-air-vehicle development. *Journal of aircraft*, *43*(2), 290–305. Available from `http://cat.inist.fr/?aModele=afficheN&cpsidt=17667289`

Raffel, M., Willert, C., & Wereley, S.T., Kompenhans, J. (2007). *Particle Image Velocimetry* (2nd ed. ed.). Springer.

Shuster, M. D., & Oh, S. D. (1981, January). Three-axis attitude determination from vector observations. *Journal of Guidance, Control, and Dynamics*, *4*(1), 70–77. Available from `http://doi.aiaa.org/10.2514/3.19717`

Willert, C. (1997). Stereoscopic digital particle image velocimetry for application in wind tunnel flows.

Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. In *iccv* (Vol. 00, p. 666). Published by the IEEE Computer Society. Available from `http://www.computer.org/portal/web/csdl/doi/10.1109/iccv.1999.791289`

Zhang, Z. (2001). *Microsoft Easy Camera Calibration Tool.* Microsoft. Available from `http://research.microsoft.com/en-us/um/people/zhang/calib/`