Extended Quadrotor Dynamics: from Simulations to Experiments

Master's thesis
DC 2019.090

G.H. Brekelmans

Coach: dr. ir. A.A.J. Lefeber
Supervisor: prof. dr. H. Nijmeijer

Department of Mechanical Engineering
Dynamics and Control (D&C) group
Eindhoven University of Technology

Eindhoven, September 27, 2019

Summary

Autonomous Unmanned Areal Vehicles (UAVs or drones) have been a widely discussed topic, where a common goal is to have multiple drones simultaneously tracking predefined trajectories. Previous research includes the design of a cascaded controller for a quadrotor with proven stability that has shown a good tracking behavior for slow trajectories. When the velocities and thus the aggressiveness of the trajectories are increased, the tracking behavior deteriorated due to the lack of external force regulation in the controller. In the simulation model, however, it results in a zero error tracking behavior. Therefore, the main goal of this thesis is to improve the resemblance of the simulation model and the experimental setup. A representative simulation model is crucial for the controller design and the testing of new software. The experimental setup used in this case is the Parrot Mambo Fly, where previous work has focused on the Parrot AR.Drone 2.0. First experiments with this new drone have demonstrated that its default estimator is inadequate in horizontal position estimation. Further research, using position data from an external camera system, has indicated that this inadequacy is mainly caused by false data coming from the optical flow sensor. Due to the inaccessible Internal Measurement Unit (IMU) the cause of this false data cannot be indicated. However, including an altitude dependent factor on the optical flow data has resulted in a significant improvement of the position estimation. Consequently, the estimation data is assumed to be usable for further experiments for most trajectories, only in trajectories with a varying altitude the improved estimator still shows some inaccuracies. For the system identification, the mass moment of inertia and the PWM to thrust ratio are determined experimentally, where the latter one has shown to be a better representation of the actual ratio in comparison with the value that is given by the manufacturer. Thereafter, this research has focused on increasing the resemblance between the simulation and experimental results. To model the external forces acting on the drone, a lumped parameter has been used that includes multiple aerodynamic effects depending on the horizontal velocity. This parameter has been used in simulation to fit the output data with experimental results. Consequently, a value for this parameter has been found that reduces the gap between the simulation and experimental results for various trajectories.

Acknowledgments

First of all I would like to thank my supervisor prof. dr. H. Nijmeijer, both for providing me with the opportunity to work on this interesting project and for motivating me to obtain my Master's degree at the Dynamics and Control research group. Thank you for our monthly meetings during this project with useful discussions that provided me with new insights.

Furthermore, I would like to thank my coach dr. ir. A.A.J. Lefeber. During our weekly meetings your critical point of view and your constructive feedback have helped me to stay focused and motivated throughout the year. Thank you Erjen, for all the effort you have put into this project.

A special thanks goes out to my friends from the 'Hersenpan', for the countless hours of studying and coffee breaks throughout the years. Likewise I want to thank the students from the DCT lab for the games of table football and the pleasant time during this project.

Last but not least I want to thank my parents and my brother, for their unconditional support and for making me believe I am capable of doing so much more than I first thought.

Guido Brekelmans
September 27, 2019

Contents

Nomenclature

Reference frames

$\mathcal{I}$	Right-handed orthonormal inertial frame in North-East-Down (NED) configuration
$\mathcal{B}$	Right-handed orthonormal body fixed frame
$\mathcal{R}$	Right-handed orthonormal reference frame

Operators and Sets

$SO(3)$	The three-dimensional special orthogonal group
$\text{diag}(\cdot)$	Function for a matrix that contains the including vector on its diagonal
$\text{trace}(\cdot)$	Function that provides the sum of the diagonal elements in a matrix
$\det(\cdot)$	Determinant of the containing matrix
$\otimes$	Quaternion product
$\odot$	Quaternion rotation
$S(\cdot)$	Skew symmetric matrix of the containing vector
$\dot{x}$	Time derivative of a state x
A^T, x^T	Transpose of a matrix and vector

Acronyms

UAV	Unmanned Areal Vehicle
VTOL	Vertical Take-off and Landing
DOF	Degrees of Freedom
CPU	Central Processing Unit
IMU	Internal Measurement Unit
RGB	Red Green Blue color code
PWM	Pulse-Width Modulation

Constants and Variables

ϕ, θ, ψ	Roll, pitch and yaw angles
ρ	Position of $\mathcal{B}$ w.r.t $\mathcal{I}$
ρ_r	Reference position $\mathcal{R}$ w.r.t. $\mathcal{I}$
ρ_{IMU}	Experimental position estimation
ρ_{OT}	Position measured by Optitrack
ρ_{OF}	Position measured by integrating the optical flow
ν, ν_r	Body fixed velocities w.r.t. $\mathcal{B}$ and $\mathcal{R}$ respectively
q, q_r	Attitudes expressed in quaternions w.r.t. $\mathcal{B}$ and $\mathcal{R}$ respectively
ω	Angular velocity of the drone
Ω	Angular velocity of the rotor blade
F_D	Total drag force vector
k_c	Lumped parameter used to model drag forces
f_d	Desired thrust vector
q_d	Desired attitude
ω_d	Desired angular velocity
J	Mass moment of inertia matrix
T_i	Thrust of motor i
f	Total force generated by the motors
τ	Torque generated by the motors
Q	Optical flow factor

Chapter 1

Introduction

1.1 Background

For the last decade the market for Unmanned Areal Vehicles (UAVs or drones) has grown rapidly due to the improving technology and increasing possibilities. Drones have already proven themselves to be very useful, but now the commercial market has also grown due to the new applications. Currently, drones are most commonly used for filming and photography due to the affordability and the ease of controlling the drone. Meanwhile, companies are researching the endless possibilities, such as Amazon, which wants to deliver their packages using drones in the future. Other applications include the spraying and inspection of crops in the agriculture, or the inspection of other wide areas.

Within the drone industry a distinction is made between the multi rotor drones and the fixed wing UAVs. The main advantage of the multi rotor drones is the ability for Vertical Take-Off and Landing (VTOL). Multi rotor UAVs can therefore also reach difficult to access locations in a 3D space. The quadrotor type with four propellers is the most popular type, due to the relatively simple control of the UAV compared to a configuration with more or less rotors. A quadrotor has only four propellers with their thrust axes aligned and is, therefore, under-actuated since the 6 degrees of freedom (DOF) cannot be controlled directly. Consequently, the quadrotor first has to make an angular rotation in order to move horizontally. Where a fixed wing aircraft uses airflow on the wings, a quadrotor continuously has to compensate for the gravity and is, therefore, less efficient in forward flight.

Currently, one of the biggest challenges in the industry is to let UAVs execute their tasks autonomously. At Eindhoven University of Technology, a part of the Dynamics and Control research group is dedicated to make a relatively cheap drone fly fully autonomously using its own Central Processing Unit (CPU) and Internal Measurement Unit (IMU). Thereafter, the goal is to let multiple UAVs fly both autonomously and simultaneously. The drone that has been used so far for this research is the Parrot AR.Drone 2.0 due to its accessible IMU and its affordability. This drone can be flown using a smartphone or tablet using software of

the manufacturer. It has so far only been used indoors because of the possibility to use an external camera for the positioning and because of the decreased external influences.

In order to conduct research on a drone and to be able to test new software and controllers, a representative simulation model is crucial. One of the issues is that the simulation model of the drone provides a good representation of the experimental setup for flying at low velocities, but for higher velocities the simulation model does not coincide with experimental results. The difference is most certainly caused by non-modeled dynamics. Therefore, the goal for this research is to improve the simulation model of this drone by including dynamic behavior that is significantly present at higher velocities. Unfortunately, the Parrot AR.Drone 2.0 has been taken out of production and, therefore, an alternative is sought to continue the ongoing research. This alternative is found in the Parrot Mambo Fly, a significantly smaller drone which comes with a Matlab toolbox that can be used for simulations and experiments. The IMU of this quadrotor is slightly different compared to the AR.Drone 2.0. The most important difference is the use of an optical flow sensor for the position tracking instead of the ability to use an external camera. This, among other differences between the quadrotors, does not change the main goal of this thesis. However, issues concerning the adaptation of the new drone have to be resolved first.

1.2 Literature review

The modeling of a quadrotor, or drones in general, has been a challenging task for a long time. The interdependent and interacting factors of airflow, rotor bending and other (aero)dynamic behavior make it difficult to find a general solution in the modeling of a quadrotor. An extensive amount of research has already been done with different approaches on modeling this behavior, which is described in this section. An important aspect of the new drone is the use of optical flow for the velocity and position estimation. The use of optical flow in literature is also discussed in this section. Finally, we discuss the different attitude representations of a drone.

1.2.1 Parrot AR.Drone 2.0

Previous research includes the identification [18] and control [31] of the Parrot AR.Drone 2.0 on SO(3). Therein a simulation model has been derived that has coincident results with experiments done on the drone and [31] has subsequently designed controllers based on the simulation model. For flying at low velocities these controllers work well on the drone and it can follow a given trajectory. When the velocity of the trajectory is increased, the drone can not keep up where it can in the simulation model. During these experiments the motor inputs are never saturated. This implies that there are imperfections to the used simulation model or the used software to fly the drone. Experiments have shown that unmodeled damping is the main cause of these effects. Other unmodeled effects include the turbulence of the surrounding air and the changing performance of the motors. The most important drag forces acting on the drone to include in the simulation model are the consequences of blade

flapping, induced drag, translational drag, profile drag and parasitic drag [1] [2] [9] [22] [28]. The effects of blade flapping and other aerodynamic forces acting on the Parrot AR.Drone 2.0 were already studied [6] [23] [30]. This drone also has a camera whose data can be merged with the accelerometer and gyroscope for the calculation of optical flow [4].

1.2.2 Parrot Mambo Fly and optical flow

The Parrot Mambo Fly drone uses an optical flow sensor for the horizontal velocity and position estimation. This sensor is a part of the camera mounted on the bottom of the drone. The output of this sensor is merely a velocity and, therefore, the ground truth position is unknown. Thus, only an approximation of the horizontal position can be made. Literature shows how optical flow sensors are used to calculate the velocity of an object w.r.t. the sensor position [13] [16] [19]. In the drone industry, optical flow is often used in combination with camera detection or a Global Positioning System (GPS) for navigation and obstacle detection [10] [12]. Other research shows that four additional optical flow sensors on the sides of a drone can be used for an accurate position estimation and tracking in 3D [3]. Finally, [5] has shown how optical flow can be used for autonomous landing.

1.2.3 Attitude representation

In the software written for the Parrot AR.Drone 2.0 the attitude is represented in rotation matrices, where the default software of the toolbox for the Parrot Mambo Fly uses Euler angles. Another option would be to use quaternions. Euler angles are often used for UAV control, but they have possible singularities leading to gimbal lock [7]. The use of quaternions and rotation matrices prevent this from happening. An advantage of quaternions is that they take less storage and multiplications require less actions and is therefore more efficient [8]. The downside of using quaternions is the ambiguity, because a certain rotation can be represented in multiple ways. In the UAV industry quaternions are often used without an overall preference over rotation matrices [24] [28]. In some studies quaternions are used in the controller design [11] [15] for their properties in terms of energy.

1.3 Problem statement

The final goal of this research is to obtain a simulation model where the results coincide with experimental results for all velocities and trajectories. Future research, on the design of new controllers or other flight algorithms, will benefit from a representative simulation model. Because of the change of drone type, some of the previous work has to be revised or done differently for the implementation of the new drone. The original firmware of the Parrot Mambo Fly that can be used with an app on a smartphone or tablet has a good estimator and controller, concluding from observed stability and position tracking in flight. Unfortunately, this firmware is inaccessible and its content is confidential. The toolbox provided by Matlab

and Parrot does come with a default estimator and controller. However, the toolbox is meant for educational purposes and, consequently, the performance does not hold up to the performance of the original firmware. Therefore, a number of sub goals and steps are defined:

- Rewrite the dynamical model from the mentioned previous contributions with the attitude represented in quaternions instead of rotation matrices.
- Investigate the new drone and the provided software. Find and fix the parts that require improvement for flight.
- System identification: find the critical parameters that are needed to get a good approximation of the dynamical model for simulations and controller design. This includes the mass moment of inertia and thrust ratio of the drone.
- Implement the controller designed by van den Eijnden [31] in the software of the drone.
- Test the default estimator of the drone. This is done using Optitrack, a motion capture system that uses cameras to accurately track markers mounted on a moving object.
- When the estimator is validated, check the performance of the simulation in terms of similarities with experimental results.
- Use the knowledge from literature to find and approximate influential parameters to increase the resemblance of the simulation model and the experimental setup.

1.4 Thesis outline

The outline of this thesis is as follows. In Chapter 2 some preliminary notes are discussed which are used throughout this thesis. Specifically, the use of quaternions for the attitude representation and the use of homogeneous coordinates used in optical flow. Chapter 3 describes the basic dynamics of a quadrotor. Thereafter, the controller designed by [31] is elaborated and demonstrated in simulations. The Parrot Mambo Fly is introduced in Chapter 4 and a breakdown of the used support package is given. Thereafter, system identification is done to validate the parameters given in the support package. Chapter 5 describes how the default estimator from the support package is tested and improved for the position tracking. With the improved estimator the experimental values can be compared to the simulations, which is done in Chapter 6. Using the drag theory described in Chapter 3, the simulations are improved to increase the resemblance with experimental results. Finally, conclusions are drawn and recommendations are given in Chapter 7.

Chapter 2

Preliminaries

In this section we recall some definitions and relations that are used throughout this thesis. First we discuss the different attitude representations and why we use quaternions in this research. Thereafter, we show how homogeneous coordinates are commonly used in optical flow, which is used for the position estimation by the Parrot Mambo Fly.

2.1 Attitude representation

For every aerial vehicle the attitude is crucial for flight control. Within the control technology this attitude can be approached with different representation techniques. The most commonly used techniques to represent the attitude of a rigid body in three-dimensional space are the use of Euler angles, rotation matrices and quaternions [7]. All three methods can define the rotation of a body-fixed frame $\mathcal{B}$ w.r.t. an inertial frame $\mathcal{I}$. We only use right handed frames throughout this thesis.

2.1.1 Rotation matrices

One of the commonly used methods for representing attitude is the use of rotation matrices. The multiplication of a rotation matrix with a vector rotates the vector while preserving its length. The attitude is expressed in a 2×2 or a 3×3 orthogonal matrix R for planar and spatial rotations respectively. The set of these matrices is called a special orthogonal group and in a spatial rotation denoted as $SO(3)$. A rotation matrix $R \in SO(3)$ holds the properties [31] [7]

- $R^T = R^{-1}$, thus $RR^T = I$ where it follows that R^T describes the reversed rotation
- $\det(R) = 1$, called proper rotation matrices

where rotation matrices with $\det R = -1$ are mathematically possible, but these are called improper. Improper rotations, also called rotoinversions, consist of a rotation followed by an inversion operation, which is physically infeasible for a drone.

Rotation matrices are easy to use as attitude representations as the rotation of a vector is done using $v_{\text{rot}} = Rv$ with $R \in \mathbb{R}^{n\times n}, v \in \mathbb{R}^{n\times 1}$ for $n = 2, 3$ for planar and spatial rotations respectively. Sequential rotations can be multiplied as $R_{\text{total}} = R_2 R_1$. An advantage of using rotation matrices is the unique representation of a rotated configuration and the physical interpretation of the sequential rotations.

2.1.2 Euler angles

In rigid-body dynamics, Euler angles are most often used for the attitude representation. This method uses three sequential rotations (ϕ, θ, ψ) around the rotated initial axes. The order of rotations is predefined and several configurations are often used. In aircraft technology, a commonly used configuration is the rotation sequence of the roll (x), pitch (y) and yaw (z) angles (RPY) [27]. This rotation can be written in rotation matrices as

$$\begin{aligned} R =& R_z(\psi)R_y(\theta)R_x(\phi) \\ =& \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} \\ =& \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix}, \end{aligned} \tag{2.1}$$

where c_θ and s_θ are abbreviations for $\cos(\theta)$ and $\sin(\theta)$ respectively, the same counts for c_ϕ, c_ψ, s_ϕ and s_ψ. The main disadvantage of using Euler angles is the possibility of the situation where two of the axes are aligned, for example when $\theta = \pi/2$, which is called gimbal lock. This makes the roll and yaw angles rotate around the same axis.

2.1.3 Quaternions

A commonly used alternative for the use of rotation matrices and Euler angles is the use of quaternions. Quaternions are an extension to complex numbers invented by William Rowan Hamilton in 1843. They can describe a rotation with only four numbers as [14]

$$q = q_0 + q_1 i + q_2 j + q_3 k, \tag{2.2}$$

with i, j and k being imaginary numbers satisfying the properties

$$i^2 = j^2 = k^2 = ijk = -1. \tag{2.3}$$

Quaternions are often written as a vector $q = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T$. It describes a full Euler rotation as

$$q = \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix}, \tag{2.4}$$

or the other way around as

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{atan2}(2(q_0q_1 + q_2q_3), q_0^2 - q_1^2 - q_2^2 + q_3^2) \\ \text{asin}(2(q_0q_2 - q_3q_1)) \\ \text{atan2}(2(q_0q_3 + q_1q_2), q_0^2 + q_1^2 - q_2^2 - q_3^2) \end{bmatrix}, \tag{2.5}$$

with ϕ, θ and ψ the roll, pitch and yaw angles. Furthermore, a quaternion satisfies the following properties:

$$\text{Norm}(q) = ||q|| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}, \tag{2.6}$$

$$\text{Conj}(q) = q^* = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \end{bmatrix}^T, \tag{2.7}$$

$$q^{-1} = \frac{q^*}{||q||^2}. \tag{2.8}$$

Using the properties of (2.3), the multiplication of two quaternions can be calculated as

$$p \otimes q = \begin{bmatrix} p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3 \\ p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2 \\ p_0q_2 - p_1q_3 + p_2q_0 + p_3q_1 \\ p_0q_3 + p_1q_2 - p_2q_1 + p_3q_0 \end{bmatrix}, \tag{2.9}$$

which can also be written as the inner product of a skew-symmetric matrix $Q(p)$ and q as

$$p \otimes q = Q(p)q = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}. \tag{2.10}$$

Only a unit quaternion, which satisfies the property $||q|| = 1$, can describe a rotation. To use the quaternion as a rotational product, the rotating vector is taken as a so called pure quaternion whose scalar q_0 equals zero. It must be both pre- and post-multiplied by the quaternion and its conjugate respectively to find the rotated vector as

$$\begin{bmatrix} 0 \\ \rho_{\text{rot}} \end{bmatrix} = q \otimes \begin{bmatrix} 0 \\ \rho \end{bmatrix} \otimes q^*, \tag{2.11}$$

where ρ and ρ_{rot} are the starting and rotated vector respectively. This double quaternion product does not take less computational power compared to the rotation matrix computation, but it can be calculated more efficiently [8] as

$$\rho_{\text{rot}} = \rho + 2q_r \times (q_r \times \rho + q_0\rho), \tag{2.12}$$

where $q_r = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}^T$ is the so called vector part of the quaternion. Now a quaternion rotation takes 18 multiplications and 12 additions. For simplicity we write the double quaternion product as

$$\rho_{\text{rot}} = q \otimes \rho \otimes q^* = q \odot \rho. \tag{2.13}$$

Furthermore, the derivative of a quaternion is described as [11]

$$\dot{q} = \frac{1}{2} q \otimes \begin{bmatrix} 0 \\ \omega \end{bmatrix} = \frac{1}{2} Q(q) \begin{bmatrix} 0 \\ \omega \end{bmatrix}, \tag{2.14}$$

where $\omega = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ are the angular velocities w.r.t. the body-fixed frame.

2.1.4 Advantages and drawbacks of using quaternions

One of the claimed advantages of using quaternions is the increase of computation speed. Table 2.1 shows the difference in computational power needed for the rotating action and the multiplication for both rotational matrices and quaternions. This table shows that a rotational action is more efficient with the use of rotation matrices, while a multiplication of rotations is more efficient with the use of quaternions. Note that a rotation matrix needs a storage of 9 elements, while a quaternion needs only 4. With the use of quaternions the event of gimbal lock, where two of the rotation axes are aligned, is prevented. As for the current research in Eindhoven, gimbal lock has not been a problem yet. For further research however and doing special manoeuvres such as loopings, the implementation of quaternions is a good solution to future problems.

Table 2.1: The amount of computations needed for rotations and multiplications of both rotation matrices and quaternions [8].

Action	Additions	Multiplications
Rotation matrix multiplication	18	27
Quaternion multiplication (2.10)	12	16
Rotation matrix rotation	6	9
Quaternion rotation (2.12)	12	18

A disadvantage of using quaternions is the fact that the quaternion needs to be of unit length to describe a rotation. The rotational velocities described by quaternions $\dot{q}$ causes the quaternion norm to drift away from 1 due to numerical integration. This results in inaccurate and false values for a rotation. There are a number of possible solutions to this problem [26] such as including a control parameter in (2.14) as

$$\dot{q}_\omega(q,\omega) = \frac{1}{2}Q(q)\begin{bmatrix}0\\ \omega\end{bmatrix} + c(q)q, \tag{2.15}$$

where $c(q)$ contains a control parameter k to limit the drift as

$$c(q) = k(1 - q^T q). \tag{2.16}$$

This method, or any of the other known methods to prevent the drift, does bring extra necessary computations to the system. Of course it is also possible to include the control part in (2.15) for only certain time steps to limit the extra computations. Investigation is needed to find an optimal balance in performance and speed in these calculations.
Another important aspect of using quaternions, is the fact that the negative of a quaternion represents the same rotation, which can be observed from (2.5). The controllers were designed on SO(3) and the implementation of quaternions should result in the same control action, no matter if q or $-q$ is used to represent the attitude.

It can be concluded, following the literature and the theory described in this section, that there is no overall preference in the use of quaternions over rotation matrices or Euler angles. For this thesis, however, we have made the choice for the use of quaternions.

2.2 Homogeneous coordinates for optical flow

The optical flow sensor of the Parrot Mambo Fly drone gives a velocity in the body fixed frame. Unfortunately this velocity is given directly and the algorithm that is used is not visible/ accessible. Therefore, literature is used to understand the underlying algorithms for optical flow sensors.

Optical flow is defined as the change of the image intensities caused by the 2D projection onto a retina of the relative 3D motion of scene points [10]. This change leads to the approximation of the velocity field of the sensor w.r.t. an object in 3D space. The optical flow is created by both the translational and rotational displacement of a point $P(X, Y, Z)$ projected onto an image plane in the camera as $p(x, y)$, which is depicted in Figure 2.1 with the x direction perpendicular to the projected y-z plane. The point p can be expressed according to P and the focal length f_l as [16] [19]

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{f_\text{l}}{Z} \begin{bmatrix} X \\ Y \end{bmatrix}, \tag{2.17}$$

where the velocity of point p is calculated by differentiation, leading to the optical flow as

$$\begin{bmatrix} OF_x \\ OF_y \end{bmatrix} = T_{OF} + R_{OF}, \tag{2.18}$$

with the translational part

$$T_{OF} = \frac{1}{Z} \begin{bmatrix} -f_\text{l} & 0 & x \\ 0 & -f_\text{l} & y \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix}, \tag{2.19}$$

and the rotational part

$$R_{OF} = \begin{bmatrix} \frac{xy}{f_\text{l}} & -(f_\text{l} + \frac{x^2}{f_\text{l}}) & y \\ f_\text{l} + \frac{y^2}{f_\text{l}} & -\frac{xy}{f_\text{l}} & -x \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \tag{2.20}$$

where OF_x and OF_y are the optical flow components in x and y direction respectively, $\begin{bmatrix} V_x & V_y & V_z \end{bmatrix}^T$ and $\begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ are the translational and angular velocities, respectively, of the point P.

With the optical flow equations known there are several computation methods [12]. Methods based on differential or gradient methods are most commonly used [10]. Other methods include the correlation on block matching, methods based on energy and methods based on phase.

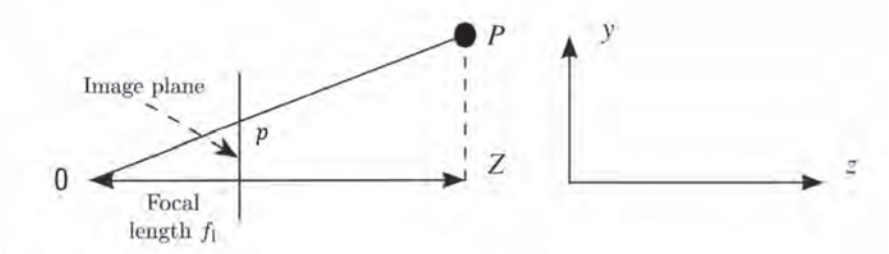


Figure 2.1: Scheme of projective geometry for optical flow measurements [10], the x direction is perpendicular to the depicted y-z plane

2.3 Concluding remarks

This chapter has described how the attitude of a rigid body can be represented using Euler angles, rotation matrices and quaternions. Throughout this thesis we use the quaternion representation, while most of the previous work in Eindhoven has used rotation matrices for the attitude representation. Therefore, the important equations are rewritten using the rules described in this section. The basics of optical flow, which is used by the Parrot Mambo Fly for the horizontal velocity estimation, are also described in this chapter. Because the algorithm that calculates the optical flow is inaccessible, this elaboration helps to understand the underlying procedures. In the next chapter we start with the modeling of a quadrotor.

Chapter 3

Quadrotor Model

In order to get a representative simulation model and to be able to design a proper controller for the quadrotor, it is fundamental to understand the underlying dynamics. This chapter gives an overview of models for the dynamics of a quadrotor. First the dynamical model is discussed with the containing translational and attitude subsystems. This is first treated without friction forces and, thereafter, the basics of the aerodynamics and other drag forces acting on a quadrotor are described. Subsequently, a previously designed controller is described which has been proven to be uniformly almost-globally asymptotically stable (UaGAS). Finally, this controller is demonstrated in a simulation without drag forces, as a starting point to compare with experimental results.

3.1 Quadrotor dynamics

To describe the dynamics of a quadrotor, we first define a ground fixed inertial frame $\mathcal{I}$ with base vectors $\{e_1, e_2, e_3\}$ and a body-fixed frame $\mathcal{B}$ with base vectors $\{b_1, b_2, b_3\}$ as depicted in Figure 3.1. This inertial frame is defined as an orthonormal North-East-Down (NED) frame. The body-fixed frame is a right-handed reference frame relative to the inertial frame. The relation between the body-fixed and the inertial frame can be split into a translational and a rotational part. The translational part of $\mathcal{B}$ w.r.t. $\mathcal{I}$ is described by a vector $\rho = \begin{bmatrix} x & y & z \end{bmatrix}^T$. For the rotational part we can write the attitude of the drone in terms of Euler angles, a rotation matrix or quaternions. We rewrite the model as described with rotation matrices in [31] into quaternions with rotation q. This results in the frictionless dynamical model of a quadrotor as

$$\dot{\rho} = q \odot \nu \tag{3.1a}$$

$$\dot{\nu} = -S(\omega)\nu + gq^* \odot b_3 - \frac{f}{m} b_3 \tag{3.1b}$$

$$\dot{q} = \frac{1}{2} q \otimes \omega \tag{3.1c}$$

$$J\dot{\omega} = S(J\omega)\omega + \tau, \tag{3.1d}$$

where $\nu \in \mathbb{R}^3$ is the body-fixed velocity, q the attitude of the quadrotor represented as a quaternion, $\odot$ the rotational product of a quaternion as described in (2.13), $\omega = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \end{bmatrix}^T$ the rotational velocity w.r.t. the body-fixed frame, $S(\cdot)$ represents a skew-symmetric matrix of the containing vector, g the gravitational constant, $f \in \mathbb{R}$ the total generated thrust, m the mass of the quadrotor, $J = \text{diag}\begin{bmatrix} J_{xx} & J_{yy} & J_{zz} \end{bmatrix}$ the mass moment of inertia matrix and $\tau \in \mathbb{R}^3$ the total moment vector in the body fixed frame acting on the drone caused by the motor thrusts.

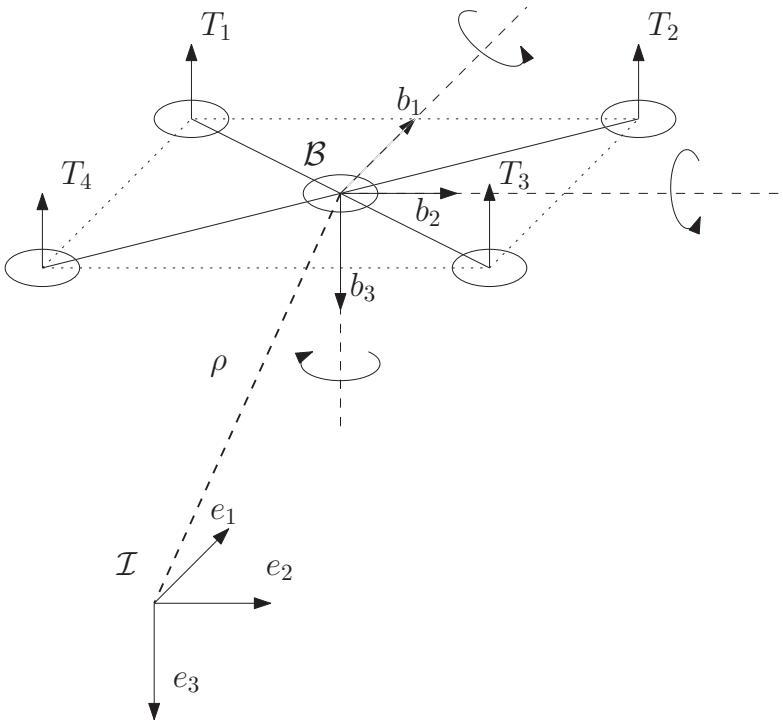


Figure 3.1: Schematic overview of the quadrotor. The inertial frame $\mathcal{I}$ is oriented in a North-East-Down (NED) frame which is fixed to the ground. The body fixed frame $\mathcal{B}$ can be expressed in the inertial frame with a translational component ρ and a rotational component q.

The total force and the torque generated by the motors can be calculated as

$$\begin{bmatrix} f \\ \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ l & -l & -l & l \\ l & l & -l & -l \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix}, \tag{3.2}$$

where l is the distance from the motors to the b_1 and b_2 axes and d a rotational damping

parameter. The torque τ_3 depends on the damping forces generated by the rotating propellers. Therefore, motors 1 and 3 rotate in clockwise direction and motors 2 and 4 in counterclockwise direction. This enables the ability to control the yaw angle ψ of the quadrotor.

3.2 Damping forces

The model represented in (3.1) assumes rigid body dynamics and no external forces are taken into account. Obviously, these forces do have an influence on the drone and are often depending on the velocity. External forces act directly on the translational and rotational acceleration and can therefore be included in the model as

$$\dot{\rho} = q \odot \nu \tag{3.3a}$$

$$\dot{\nu} = -S(\omega)\nu + gq^* \odot b_3 - \frac{f}{m}b_3 + \frac{F_D}{m} \tag{3.3b}$$

$$\dot{q} = \frac{1}{2}q \otimes \omega \tag{3.3c}$$

$$J\dot{\omega} = S(J\omega)\omega + \tau + \tau_D, \tag{3.3d}$$

where F_D and τ_D are the drag forces and moments respectively. These forces and moments contain a number of types of aerodynamic forces. The basics of these forces are explained in this section, see also [2] [22].

3.2.1 Blade flapping

The work of de Kleuver [6] contains a description on the implementation of blade flapping. This is the phenomenon where during the rotation of the rotor the advancing blade experiences more lift and the retreating blade reduced lift due to the air velocity. This causes a torque on the center of the rotor. The work of de Kleuver included the identification of the parameters concerning blade flapping using blade element theory. Because of a steady state assumption, the blade flapping angle has been described with a Fourier series. In [2] the flapping force on a rotor is described as

$$D_\beta = T\left(A_{\text{flap}}V_h + B_{\text{flap}}\frac{\omega}{\Omega}\right), \tag{3.4}$$

where T is the thrust, $A_{\text{flap}} \in \mathbb{R}^{3\text{x}3}$ and $B_{\text{flap}} \in \mathbb{R}^{3\text{x}3}$ contain parameters that depend on the geometry and properties of the blades, V_h is the horizontal velocity vector of the drone in the body-fixed frame as $\begin{bmatrix}\nu(1) & \nu(2) & 0\end{bmatrix}^T$ and Ω is the angular velocity of the rotor.

3.2.2 Induced drag

Another important phenomenon is the induced drag which is also described in [6]. In the case where the blades of the rotor do not flap freely due to stiffness, there is no balance of the aerodynamic forces. The airflow causes the blade to have a lower angle of attack and thus a lower thrust is generated. This is called induced drag and can be modeled as

$$D_i = -TK_I V_h, \tag{3.5}$$

where $K_I = \text{diag}(k_I, k_I, 0)$ and $k_I > 0$. The effect of induced drag depends greatly on the stiffness of the blades. A lower stiffness means more blade flapping and thus the rotor blades create a better balance of forces by themselves. When the quadrotors are meant to fly with higher velocities and thus required to be more agile, higher stiffness in the rotors is required, increasing the induced drag.

3.2.3 Translational drag

Translational drag is caused by the bending of the airflow as is goes through the rotor during translational motion. It is similar to induced drag, though it is caused by translational movement that redirects the airflow through the rotor and can be modeled as [2]

$$D_t = -TK_{T_1} V_h, \tag{3.6}$$

for low velocities and [2]

$$D_t = -TK_{T_2}(v_z^s + v_z^i)^4 V_h, \tag{3.7}$$

for higher velocities where $K_{T_1} = \text{diag}(k_{T_1}, k_{T_1}, 0)$ and $K_{T_2} = \text{diag}(k_{T_2}, k_{T_2}, 0)$ are lumped parameters with positive scalars that need to be determined, v_z^s and v_z^i are the vertical components of the stream and induced velocities respectively. These velocities are lower during forward flight, and thus decreasing the translational drag for higher horizontal velocities. The higher velocities are stated as $V_h > w$ where $w > 0$ is a function of the blade geometry, thus independent of the rotor speed. The models for both low and high velocities can be verified using the lift induced drag component from Blade Element Theory (BET) [21], as described in [2].

3.2.4 Profile drag

Profile drag is caused by the profile of the rotor blades as they move through the air. While the drone is hovering this profile drag is zero due to the equal magnitude on opposing sides. The profile drag is given by [2]

$$D_{\mathrm{p}} = -\int_0^R \rho c \left((c_{d0} + c_{d\alpha})\omega r^2 - (v_z^s + v_z^i) r \right) V_h \mathrm{d}r, \tag{3.8}$$

where R is the radius of the blade, c the cord length of the blade, c_{d0} and $c_{d\alpha}$ are drag coefficients of the rotor. This can also be written as a function of the horizontal velocity as

$$D_{\mathrm{p}} = -TK_{\mathrm{p}}V_h, \tag{3.9}$$

with $K_p = \mathrm{diag}(k_p, k_p, 0)$ as a lumped parameter.

3.2.5 Parasitic drag

Parasitic drag is the type of drag caused by the non-lifting parts of the drone, in other words the air friction caused by the frame of the drone. This type of drag is often neglected at velocities below 5 m/s. The parasitic drag can be described as [2]

$$D_{\mathrm{par}} = -T|\nu|K_{\mathrm{par}}\nu, \tag{3.10}$$

where $K_{\mathrm{par}} \in \mathbb{R}^{3\mathrm{x}3}$ such that K_{par} is positive semi-definite, depending on the geometry of the drone. Note that the parasitic drag can be active in all three body-fixed frame directions, due to the geometry of the frame.

3.2.6 Total drag

When the drag forces are written as functions dependent on the same velocity with lumped parameters and knowing that all rotors of the drone have the same drag properties, a lumped drag coefficient can be derived as [2]

$$k_c = 4(k_I + k_{T_1} + k_p + A_{1_c}), \tag{3.11}$$

for low velocities, where A_{1_c} is a parameter involved in the blade flapping term of A_{flap}. This

statement ignores small coupling terms in the flapping coefficient and assumes $\frac{\omega}{\Omega} \approx 0$. The total drag force acting on a single rotor j can be described as

$$D_j = D_{i_j} + D_{t_j} + D_{p_j} + D_{\beta_j}, \tag{3.12}$$

and the total drag force in the body fixed frame acting on the drone for low velocities, neglecting the parasitic drag, as

$$F_D = \sum_{j=1}^{4} D_j = -TK_cV, \tag{3.13}$$

where $K_c = \text{diag}(k_c, k_c, 0)$. To give an impression, Figure 3.2 shows the basic relations of the magnitude of the described drag forces with respect to the velocity. Herein we can observe that, if we neglect the parasitic drag, for velocities $V_h < w$ the total drag can be stated as linearly dependent on the horizontal velocity, including the effects of blade flapping, induced drag, profile drag and translational drag. Therefore, we can approach the drag with the described lumped parameter K_c as described in (3.13). Since w depends on the blade geometry, we could identify this parameter experimentally or mathematically using literature. However, due to time restrictions we assume $V_h < w$ for now. In Chapter 6 the lumped parameter K_c is included in the model to reduce the gap between the simulation results and experimental results.

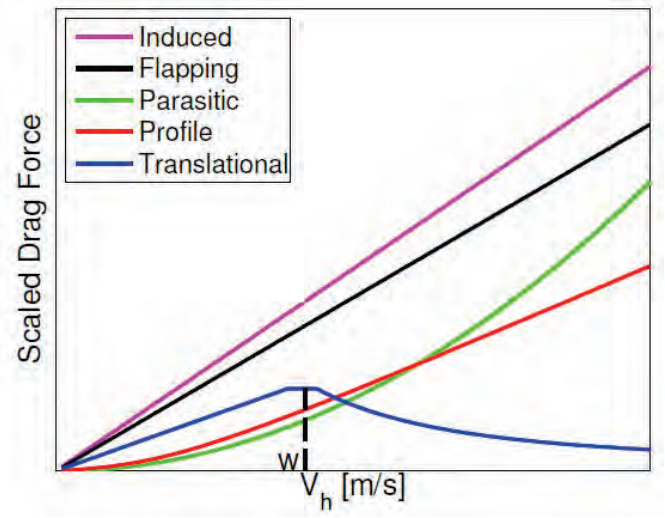


Figure 3.2: Impression of the magnitude of the described types of drag with respect to velocity of the drone [2]

3.2.7 Ground effect

During take-off and landing the drone flies close to the ground, which influences the airflow around the rotor. The airflow changes the vortexes and reduces the induced flow and therefore increasing the angle of attack and the lift. This is called the ground effect. A rule of thumb is that this effect becomes influential under a height of the rotor diameter. Since the Parrot Mambo Fly is a relatively small quadrotor with short rotor blades, we neglect the ground effect in this thesis.

3.3 Position control by v.d. Eijnden

This section describes the controller that is previously designed for the Parrot AR.Drone 2.0 in [31] which is slightly improved in [20]. The controller is designed using rotation matrices which is rewritten using quaternions, according to the rules described in section 2.1.3. The literature proves that this controller is uniformly almost-globally asymptotically stable (UaGAS) and should, therefore, work on the Parrot Mambo Fly as well. We assume the quadrotor dynamics of (3.1) and, therefore, the input for the quadrotor is the total thrust f and the moment vector τ. For the controller, we first assume that a feasible reference trajectory $(\rho_r, q_r, \nu_r, \omega_r, f_r, \tau_r)$ with reference frame $\mathcal{R}$ is given satisfying

$$\dot{\rho}_r = q_r \odot \nu_r \tag{3.14a}$$

$$\dot{\nu}_r = -S(\omega_r)\nu_r + g q_r^* \odot b_3 - \frac{f_r}{m} b_3 \tag{3.14b}$$

$$\dot{q}_r = \frac{1}{2} q_r \otimes \omega_r \tag{3.14c}$$

$$J\dot{\omega}_r = S(J\omega_r)\omega_r + \tau_r. \tag{3.14d}$$

The designed controller has a cascaded structure, consisting of a subsystem for position control and a subsystem for attitude control.

3.3.1 Position tracking control

First we take a closer look at the position control in the cascaded structure of the controller. The tracking error of the drone can be expressed in the body-fixed frame of the reference as

$$\begin{bmatrix} \rho_e \\ \nu_e \end{bmatrix} = \begin{bmatrix} q_r^* \odot (\rho_r - \rho) \\ \nu_r - (q_r^* \otimes q) \odot \nu \end{bmatrix}, \tag{3.15}$$

which leads to the tracking error dynamics

$$\dot{\rho}_e = -S(\omega_r)\rho_e + \nu_e \tag{3.16a}$$
$$\dot{\nu}_e = -S(\omega_r)\nu_e + \frac{f}{m}\left((q_r^* \otimes q) \odot b_3\right) - \frac{f_r}{m}b_3. \tag{3.16b}$$

The choice for expressing the tracking error in the body-fixed reference frame is to be able to assume $\frac{f}{m}\left((q_r^* \otimes q) \odot b_3\right) - \frac{f_r}{m}b_3$ to be a virtual input u by controlling the thrust and the attitude. Therefore, the proposition of [20] is to define

$$\dot{\rho}_e = -S(\omega_r)\rho_e + \nu_e \tag{3.17a}$$
$$\dot{\nu}_e = -S(\omega_r)\nu_e + u, \tag{3.17b}$$

in closed loop with the dynamic state feedback

$$u = q_r^* \odot (K_P P_e + K_p p_e) - q_r^* \odot \left(K_P(q_r \odot \sigma_1(\bar{\rho}_e))\right) - k_\rho \sigma_2(\bar{\rho}_e) - K_\nu \sigma_3(\bar{\nu}_e) \tag{3.18a}$$
$$\dot{P}_e = p_e \tag{3.18b}$$
$$\dot{p}_e = -K_P P_e - K_p p_e + K_P(q_r \odot \sigma_1(\bar{\rho}_e)), \tag{3.18c}$$

where $K_P = K_P^T > 0, K_p = K_p^T > 0, K_\nu = K_\nu^T > 0$ and $k_p > 0$ are controller gains, P_e and p_e provide integral action on the position control. The saturation function $\sigma_i(x) = \frac{x}{\sqrt{1+x^T x}}$ bounds the error functions

$$\bar{\rho}_e = \rho_e + q_r^* \odot P_e \tag{3.19a}$$
$$\bar{\nu}_e = \nu_e + q_r^* \odot p_e, \tag{3.19b}$$

which concludes the position tracking error dynamics.

3.3.2 Attitude control

The previous section has showed how the use of a virtual input $u = \frac{f}{m}\left((q_r^* \otimes q) \odot b_3\right) - \frac{f_r}{m}b_3$ can stabilize the position tracking error dynamics. Now we aim to use the inputs f and τ to let $f\left((q_r^* \otimes q) \odot b_3\right)$ converge to the vector $f_r b_3 + mu$ for the attitude control, where u is

given by (3.18a). After showing that by properly selecting $\sigma_i, P_e(t_0)$ and $p_e(t_0)$ the virtual input u is bounded in [20], this can be reduced to

$$f = ||f_r b_3 + mu||, \tag{3.20}$$

where we have the first input f. We now define a desired thrust direction

$$f_d = \begin{bmatrix} f_{d1} \\ f_{d2} \\ f_{d3} \end{bmatrix} = \frac{f_r b_3 + mu}{||f_r b_3 + mu||}. \tag{3.21}$$

From here [20] used the rotation matrix

$$R_d = \begin{bmatrix} 1 - \frac{f_{d1}^2}{1+f_{d3}} & -\frac{f_{d1}f_{d2}}{1+f_{d3}} & f_{d1} \\ -\frac{f_{d1}f_{d2}}{1+f_{d3}} & 1 - \frac{f_{d2}^2}{1+f_{d3}} & f_{d2} \\ -f_{d1} & -f_{d2} & f_{d3} \end{bmatrix}, \tag{3.22}$$

to rotate the desired thrust vector to the thrust vector of the reference b_3. This is rewritten into quaternions as $q_d = \frac{q_{d0}}{||q_{d0}||}$ with

$$q_{d0} = \begin{bmatrix} 1 + f_{d3} \\ -f_{d2} \\ f_{d1} \\ 0 \end{bmatrix}. \tag{3.23}$$

For the derivation of this equation see Appendix A. This expression also gives

$$\omega_d = \begin{bmatrix} -\dot{f}_{d2} + \dfrac{f_{d2}\dot{f}_{d3}}{1+f_{d3}} \\ \dot{f}_{d1} - \dfrac{f_{d1}\dot{f}_{d3}}{1+f_{d3}} \\ \dfrac{f_{d2}\dot{f}_{d1} - f_{d1}\dot{f}_{d2}}{1+f_{d3}} \end{bmatrix}. \tag{3.24}$$

If we use (3.20), (3.21) and the expression for q_d we can write $f_r b_3 + mu = f(q_d \odot b_3)$. Now the aim of converging $f\left((q_r^* \otimes q) \odot b_3\right)$ to $f_r b_3 + mu$ can be replaced by converging $q_r^* \otimes q$ to q_d. Therefore, the attitude error is defined as

$$q_e = q_d^* \otimes (q_r^* \otimes q), \tag{3.25}$$

in the body-fixed frame with the corresponding angular velocity tracking error as

$$\omega_e = \omega - (q^* \otimes q_r) \odot \omega_r - q_e^* \odot \omega_d, \tag{3.26}$$

following from

$$\dot{q}_e = \frac{1}{2} q_e \otimes \omega_e. \tag{3.27}$$

If we now look at the error dynamics

$$\begin{aligned} J\dot{\omega}_e =& S(J\omega)\omega + \tau - J((q^* \otimes q_r) \odot \dot{\omega}_r) \\ &+ JS(\omega_e)[\omega - \omega_e] + J\big(q_e^* \odot [S(\omega_d)(q_d^* \odot \omega_r) - \dot{\omega}_d]\big), \end{aligned} \tag{3.28}$$

we can select the input τ as

$$\begin{aligned} \tau =& - K_\omega \omega_e + K_R \sum_{i=1}^{3} k_i (e_i \times (q_e \odot e_i)) - S(J\omega)\omega - J\big(q_e^* \odot [S(\omega_d)(q_d^* \odot \omega_r) - \dot{\omega}_d]\big) \\ &- JS(\omega_e)(\omega - \omega_e) + J\left((q^* \otimes q_r) \odot \dot{\omega}_r\right), \end{aligned} \tag{3.29}$$

which, together with the position tracking control, results in the closed loop system

$$\dot{P}_e = p_e \tag{3.30a}$$
$$\dot{p}_e = - K_P P_e - K_p p_e + K_P(q_r \odot \sigma_1(\bar{\rho}_e)) \tag{3.30b}$$
$$\dot{\bar{\rho}}_e = - S(\omega_r)\bar{\rho}_e + \bar{\nu}_e \tag{3.30c}$$
$$\dot{\bar{\nu}}_e = - S(\omega_r)\bar{\nu}_e - k_p \sigma_2(\bar{\rho}_e) - K_\nu \sigma_3(\bar{\nu}_e) + \frac{f}{m}(q_r^* \otimes q) \odot b_3 - \frac{f}{m}([q_r^* \otimes q] \otimes q_e^*) \odot b_3 \tag{3.30d}$$
$$\dot{q}_e = \frac{1}{2} q_e \otimes \omega_e \tag{3.30e}$$
$$J\dot{\omega}_e = - K_\omega \omega_e + K_R \sum_{i=1}^{3} k_i (e_i \times (q_e^* \odot e_i)), \tag{3.30f}$$

which concludes the error dynamics of the controller.

3.4 Simulations

Now the basic dynamics and used the controller are described, we can do simulations to show the dynamic behavior of a quadrotor. Since the stability and performance of the controller is already widely discussed in [20] [31], this section shows the simulation results with realistic initial conditions which we can compare with experimental results in Chapter 6. For now we use the system parameters provided by Parrot in the Simulink support package as $m = 0.063$ kg and $J = \text{diag}\begin{bmatrix}0.5829 & 0.7169 & 1.0000\end{bmatrix} * 10^{-4}$, which are further discussed in section 4.4. For the controller we use the gains $K_P = 0.4I$, $K_p = I$, $k_p = 2.6$, $K_\nu = 2I$, $K_\omega = 30J$, $K_R = 70J$, $k_1 = 0.9$, $k_2 = 1$ and $k_3 = 1.1$. We define a reference trajectory as

$$\rho_r(t) = \begin{bmatrix} \cos(t) \\ \sin(t) \\ -1.5 - \sin(t) \end{bmatrix}, \tag{3.31}$$

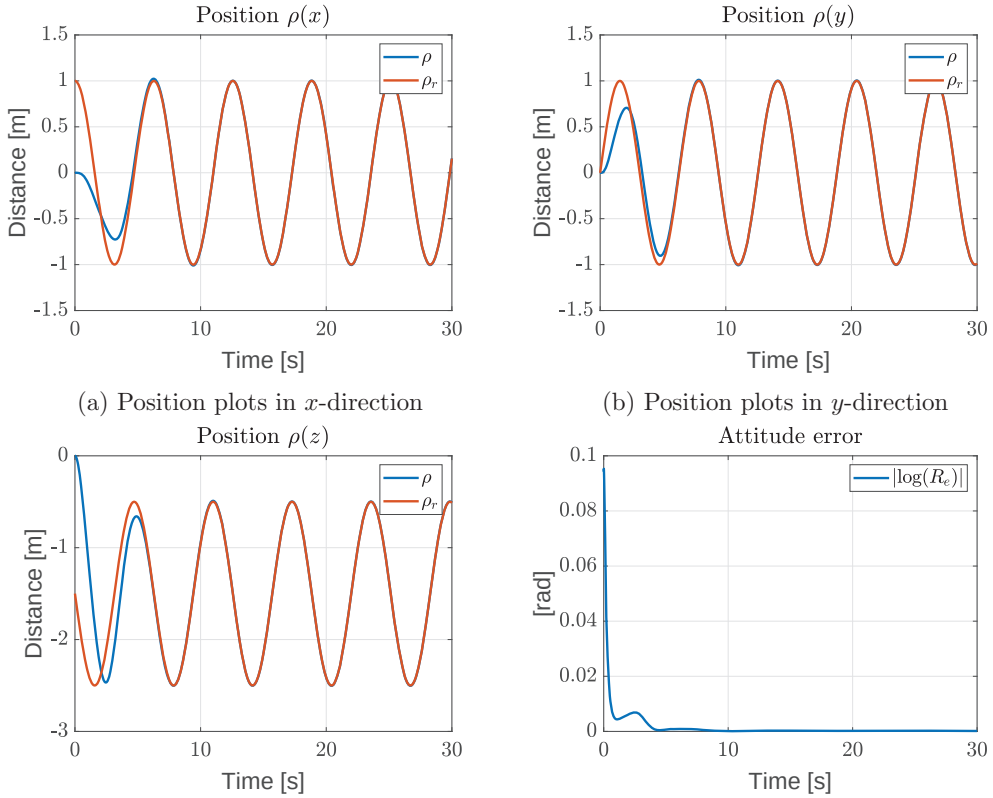
which, according to (3.14), also determines f_r, q_r, ω_r, τ_r and ν_r. The yaw angle is determined according to (3.21). We set the initial conditions as

$$\rho(t_0) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T \tag{3.32a}$$
$$q_0 = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix} \tag{3.32b}$$
$$\nu(t_0) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T \tag{3.32c}$$
$$\omega(t_0) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T. \tag{3.32d}$$

For now we assume no drag and no sensor noise. With these assumptions we simulate the system according to (3.1) with f and τ the controller output from (3.20) and (3.29). Figure 3.3 shows the resulting behavior of the quadrotor. It can be observed that the quadrotor accurately tracks the reference trajectory after $\pm$ 6 seconds. Note that the attitude error is shown in terms of $|\log(R_e)|$ instead of q_e for a clear view, since the error quaternion q_e converges to $\begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}$ where R_e converges towards I.

3.5 Concluding remarks

This chapter has first described the basics of quadrotor dynamics using the chosen inertial and body-fixed frames. Thereafter, the most important damping forces acting on the drone are discussed. These forces have an increasing influence on higher velocities. It has also shown that by making certain assumptions, most of these drag forces can be merged in the modeling of a quadrotor with a lumped parameter. Using the described dynamics, the controller designed by van den Eijnden is described and finally tested in simulations. These



(a) Position plots in x-direction

(b) Position plots in y-direction

(c) Position plots in z-direction

(d) Attitude error

Figure 3.3: Simulation results using the model (3.1), the controller as described in section 3.3 and the initial conditions (3.32).

simulations have shown a starting point from where we try to increase the resemblance with experimental results. The next chapter elaborates on the experimental setup.

Chapter 4

The Parrot Mambo Fly

In the previous chapter the basics of a quadrotor model and the designed UaGAS controller are discussed. To reach the goal of reducing the gap between simulation results and experimental results, we take a look at the provided quadrotor in this chapter, which is the Parrot Mambo Fly. We discuss the details of the quadrotor and the differences with the previously used Parrot AR.Drone 2.0. They are referred to as the Mambo and the AR-drone. The main motivation for choosing the Mambo is that Parrot provides a Simulink support package in cooperation with Mathworks. This support package is elaborated in this chapter and a breakdown of the model is shown. Finally, using the default controller and estimator provided by the support package the first experiments are conducted on the drone.

4.1 The Parrot Mambo Fly vs AR.Drone 2.0

As mentioned, during this research we have had to switch to the Mambo because the AR-drone is taken out of production. This drone has different hardware and implementation firmware. Consequently, the software has to be modified as well. This section elaborates on the relevant differences. Figure 4.1 shows both drones in the indoor configuration, i.e., with a bumper hull. The most striking difference is the size, where the AR-drone is ± 60x60 cm, the Mambo is ± 18x18 cm.

The AR-drone can be connected through WiFi using a DHCP host and therefore a Secure Shell (SSH) connection can be made. This facilitates the ability to find out more about the present hardware and thus the sensors. The information about the sensors can be useful during the programming of the drone. Unfortunately, the Mambo uses a Bluetooth connection with a host and therefore no SSH connection can be made. Due to the specs we do know what type of sensors are present, but no further details are known. After contacting Parrot on this issue, they have indicated that further information on the IMU and the sensors is confidential.

Figure 4.1: The Parrot AR.Drone 2.0 (left) and the Parrot Mambo Fly (right).

The differences in the available sensors of drones are shown in Table 4.1. The most important differences are the presence of an optical flow sensor and the absence of a magnetometer in the Mambo. The magnetometer is used for the estimation of the attitude and, therefore, the estimation of the Mambo could be poor without a proper state estimator. The optical flow sensor in the Mambo is used in combination with the gyroscope and the accelerometer for the estimation of the velocity and integrated to estimate the position, whereas the AR-drone uses an external camera to detect LEDs mounted on the drone to estimate the position. Consequently, the position of the Mambo drifts almost inevitably over time due to sensor noise since this estimation is not compared to a ground truth, where the AR-drone does. Note: a separately sold camera can be mounted on the Mambo as a front-facing camera to receive additional data.

Table 4.1: Overview of the sensors on the Parrot drones.

Sensor	AR.Drone 2.0	Mambo Fly
Down-facing camera	320 x 240	640 x 480
Front-facing camera	1280 x 720	No
GPS	Yes	No
Barometer	Yes	Yes
3-axis Magnetometer	Yes	No
3-axis Gyroscope	Yes	Yes
3-axis Accelerometer	Yes	Yes
Ultrasonic distance sensor	Yes	Yes
Optical flow sensor	No	Yes

4.2 Simulink support package

The main advantage of the Mambo is that Parrot provides the 'Simulink Support Package for Parrot Minidrones', which is also suited for other types of Parrot's minidrones. This support package has an environment where a flight control system can be both simulated and mounted on the drone through a Bluetooth connection. This simulation environment is not used because our goal is to make a proper simulation environment manually with direct control on the influential parameters. Plus, first experiments and simulations have shown that the simulation environment already shows less similarities with the experiments, compared to the simulations by using the model described in Chapter 3.

Thus, the most important part of the support package is the flight control system that can be mounted on the drone. This part is displayed in Figure 4.2. While the first input is referred to as the command input, it can only be used for external inputs such as a Flypad controller. Therefore, the reference input is given inside the flight control system block. The second input is the sensor input. This contains all the available sensor data from the sensors shown in Table 4.1. The third input contains the image data from the camera, which can be processed to RGB values and then used for further vision based data processing. Within the flight control system the input data is first used in a default estimator to estimate the states of the drone. Thereafter, these estimates are used in a default controller with the reference to determine the outputs. The outputs are the motor outputs in PWM signals and a flag signal to shut down the drone in case of an error. This application uses a fixed sampling frequency of 200 Hz and cannot be adjusted. This concludes the basics of the support package which enables the flight control system to be build onto the Mambo.

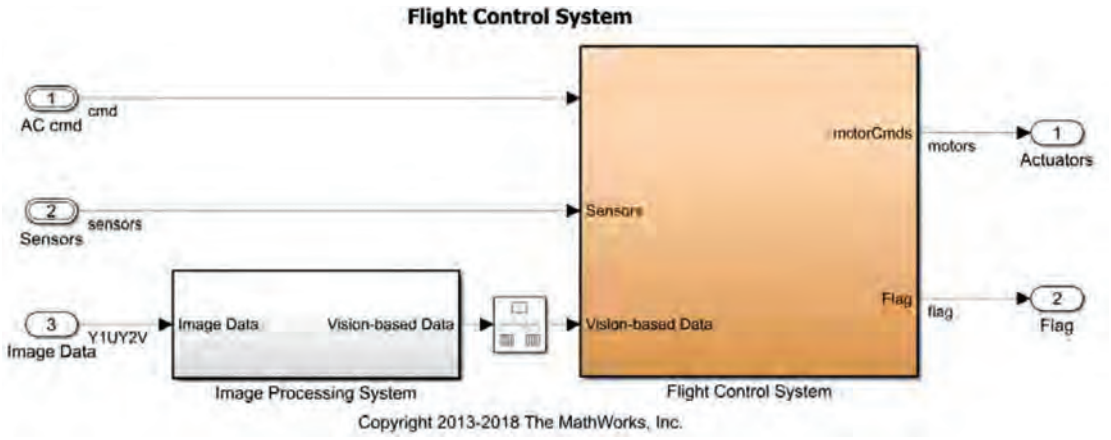


Figure 4.2: The flight control system that is mounted on the drone using the Simulink support package.

4.3 First experiments

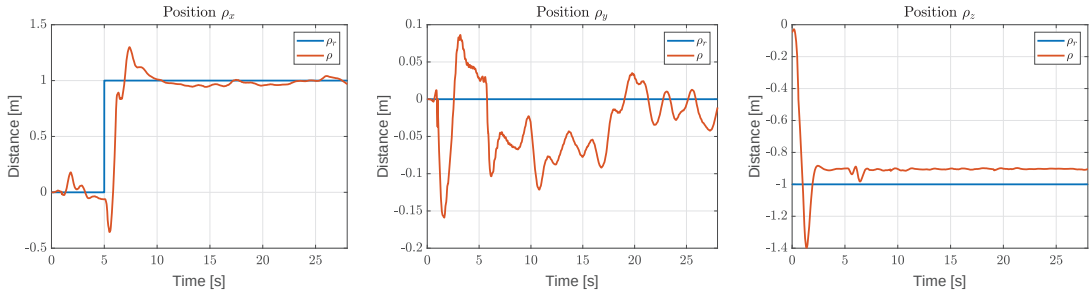
Now that the support package is introduced, this section shows the experimental performance of the default flight control system on the Mambo, provided by Parrot. For instructions on connecting the drone and how to use the support package, see the manual attached in Appendix B. The flight control system includes a default controller and a default estimator. The default estimator is elaborated and tested in Chapter 5. If we take a closer look on the default controller we observe that the total thrust force f is an output of a simple proportional-derivative (PD) controller depending on the z position and velocity. Similarly, the yaw angle is controlled with a PD controller depending on the yaw angle and angular velocity. For the roll and pitch angles, a proportional-integral-derivative (PID) controller is used. However, the reference attitude used to calculate the attitude error for the PD action herein switches between the attitude reference input, and values depending on the x and y position errors. In other words, the roll and pitch input torques depend on either the attitude reference or the position reference. In addition, all derivative actions of the controller do not take a reference velocity into account, i.e., the penalty on the velocity is proportional to the velocity, instead of the velocity error. Hence, the tracking behavior of a certain trajectory using this controller should be poor compared to the controller described in section 3.3 since the input forces f and τ are not dependent on all trajectory values from (3.14) as in (3.20) and (3.29). This difference in performance could increase with the aggressiveness of the reference trajectory.

To show the performance of the drone with the default estimator and controller, we experiment with two trajectories. First a step response is measured as the x reference is set to 1 after 5 seconds, i.e., after takeoff. The results are shown in Figure 4.3. In x and y direction the behavior seems normal with a low tracking error. In the z direction we can observe that the reference could not be reached. This is an expected problem since the control in z direction involves no integral action. Other factors such as the thrust ratio or an incorrect mass for the gravity could also influence this error. Another experiment is done using a full 3D trajectory as reference as

$$\rho_r(t) = \begin{bmatrix} \frac{1}{2}\sin(t) \\ \frac{1}{2}\cos(t) \\ -1.5 - \frac{1}{2}\sin(t) \end{bmatrix}. \tag{4.1}$$

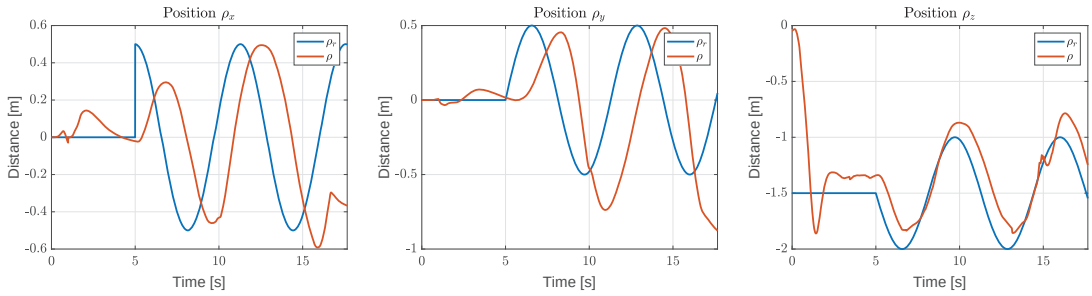
The results of this experiment are shown in Figure 4.4. It can be observed that there is a delay in x and y direction. This delay could be caused by the fact that the controller does not take the full reference into account, i.e., no velocities, attitude or angular accelerations as the controller described in section 3.3 does. The velocities can be implemented by using the error between a reference trajectory velocity and the actual velocity in the derivative action, which should improve the tracking performance after some gain tuning. However, since these are already taken into account in the cascaded controller described in section 3.3, this controller is implemented in the flight control system in all further simulations and experiments in this thesis. Furthermore, the last experiment has been terminated early, because the Mambo would fly into the surrounding net. The drone has drifted away while the position plots do not show a major drift. This implies that there is an error in the default estimator. Looking

back at the previous experiment, the mambo should have traveled a distance of 1m in x direction. This distance is actually observed to be larger, which again implies that there is an error in the default estimator. Therefore, the estimator also needs further investigation which is done in Chapter 5.



(a) Position plot in x direction (b) Position plot in y direction (c) Position plot in z direction

Figure 4.3: Experimental results of a step function using the default estimator and controller of the support package.



(a) Position plot in x direction (b) Position plot in y direction (c) Position plot in z direction

Figure 4.4: Experimental results with a circular reference trajectory using the default estimator and controller of the support package. The experiment has been terminated early because the Mambo has flown into its surroundings.

4.4 System Identification

This section describes the steps taken for the identification of certain parameters of the Mambo, which is crucial for deriving a representative simulation model. The simulation environment of the support package described in section 4.2 contains, among others, parameters for the mass, mass moment of inertia and a required thrust to PWM ratio. The mass already has significant difference when the Mambo is weighted on a scale: 68g on the scale compared to 63g as indicated in the toolbox. This is possibly caused by the four bumpers that can be dismounted and have a weight of $\pm$ 1g each, but this is not indicated anywhere. Next, the mas moment of inertia and the thrust ratio are experimentally determined.

4.4.1 Mass Moment of Inertia

To verify the values for the inertia provided by the support package, experiments are done. Previous work on measuring inertia [17] has been applied to the AR-drone by [18]. Their method uses an experimental setup with the drone suspended as a bifilar pendulum as shown in Figure 4.5. The test object hangs on two vertical wires with a length h and distance D between the wires. An initial rotation θ_0 is applied to the object and the induced oscillations are measured with the gyroscope. These measurements are then used with the equations of motion

$$\begin{cases} x_1 = \theta \\ x_2 = \dot{\theta} \end{cases} \tag{4.2a}$$

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -(\frac{K_D}{J}|x_2| + \frac{C}{J})x_2 - \frac{mgD^2}{4Jh}\frac{\sin x_1}{\sqrt{1-\frac{1}{2}(\frac{D}{h})^2(1-\cos x_1)}}, \end{cases} \tag{4.2b}$$

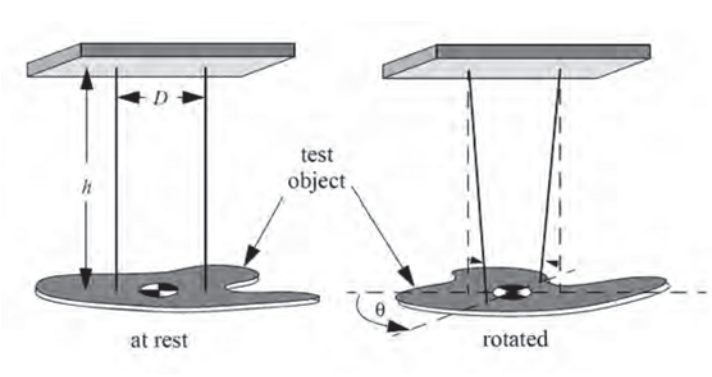
where K_D and C are damping parameters. This system can be simulated using the known initial rotation. The unknown parameters K_D, C and J are determined using an optimization algorithm that minimizes the cost function

$$f_{\text{cost}} = \parallel \dot{\theta}_{\text{sim}} - \dot{\theta}_{\text{exp}} \parallel, \tag{4.3}$$

where $\dot{\theta}_{\text{sim}}$ and $\dot{\theta}_{\text{exp}}$ are vectors containing the simulated and measured values, respectively, of the angular velocity for all time steps. The algorithm fits the equations of motion onto the experimental data and finds the estimated values for the unknown parameters, where J is obviously the most important one. The described procedure is done multiple times for all three body-fixed axes. Figure 4.6 shows the optimized results for an experiment in J_{yy} direction. Multiple experiments have shown small differences in optimized values ($\pm$ 2%), so the mean values of 8 results are calculated. Table 4.2 shows the resulting values for the mass moment of inertia and the values initially given in the support package. It can be observed that there is a significant increase of 19, 8 and 50 % for the J_{xx}, J_{yy} and J_{zz} direction respectively.

Table 4.2: Initial values of the mass moment of inertia and the estimated values following from the bifilar pendulum experiment.

	Initial	Measured
J_{xx}	$0.5829 * 10^{-4}$	$0.69 * 10^{-4}$
J_{yy}	$0.7169 * 10^{-4}$	$0.775 * 10^{-4}$
J_{zz}	$1.000 * 10^{-4}$	$1.5 * 10^{-4}$



(a) Illustration of a bifilar pendulum experiment [17]

(b) Experimental setup of the Mambo as a bifilar pendulum

Figure 4.5: Bifilar Pendulums

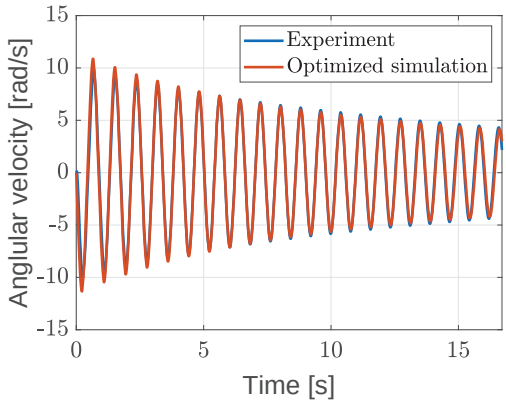


Figure 4.6: Optimized result for the J_{yy} direction.

4.4.2 Thrust Ratio

In the provided support package, after the required total force f and torque τ are calculated by the controller, the thrust per motor is determined. For that, the conversion matrix (3.2) is inverted and given in the support package as

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 0.25 & 103 & -5.7 & -5.7 \\ 0.25 & -103 & -5.7 & 5.7 \\ 0.25 & 103 & 5.7 & 5.7 \\ 0.25 & -103 & 5.7 & -5.7 \end{bmatrix} \begin{bmatrix} f \\ \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}. \tag{4.4}$$

In order to find the required PWM signals to be sent to the motors, the thrust per motor values are multiplied by a constant of 1530.7 in the support package. Hovering experiments

have shown, however, that a higher input force is needed than expected. Theoretically, the total force needed to hover a mass of 68g would be $0.67N$, but during experiments an average of $\pm\, 0.72N$ is sent to the motor commands after the integral action of the controller. Note that this value also includes the force of the virtual input as described in section 3.3 to converge towards zero error in x and y direction.

To validate the given thrust/ PWM ratio, a simple experiment is done to measure the ratio between motor input and rotor thrust. The drone is mounted upside down on a scale where the force of the rotors can be measured in weight. The fact that the drone is mounted upside down also reduces the ground effect which can cause measurement errors. The assumptions are made that all four motors and propellers have an equal PWM to thrust ratio and that the thrust is constant for a continuous PWM input. The measurements are done as follows:

- A force signal is sent to the drone according to (4.4) and multiplied by 1.5307e+03. The torque signals are zero. Therefore each rotor should have equal thrust.
- The measured added mass is read from the scale.
- The thrust per motor is calculated with this measured mass m using $T = mg/4$ with g being the gravitational constant.

This procedure is repeated 5 times for input forces $0.1N$ to $1.2N$ with steps of $0.1N$. The results are shown in Figure 4.8a with the blue line representing the linear relation from the support package between four motors receiving the same command and the red dots representing the experimental values. The curve that can be fitted through the experimental data points indicates that there is a relation of the form

$$T = aX^b, \tag{4.5}$$

where X represents the PWM signal and a and b are constant values that have to be determined. An optimization algorithm can be used to find the values of a and b. Experiments have shown, however, that for the experimental data points above motor commands of ± 350, the thrust is rapidly decreasing after applying the constant command input. Therefore, these data points are less trustworthy. Moreover, flights and simulations have shown that for most manoeuvres the total thrust command would not exceed $0.8N$. This means $\pm 0.2N$ per motor and thus the first series of data points are assumed to be of higher importance. Therefore we can also take a look at the logarithmic plots as

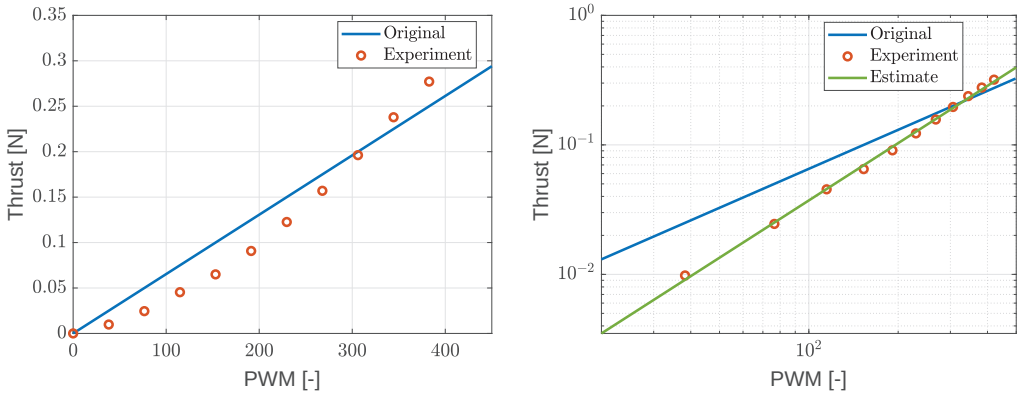
$$\begin{aligned} \log(T) &= \log(aX^b) \\ &= \log(a) + b\log(X), \end{aligned} \tag{4.6}$$

which weighs the first data points heavier in an optimization algorithm. For the optimization we use a least squares method to minimize the error between the measured data points and

the corresponding points of (4.6). This results in values $a = 4.29 * 10^{-5}$ and $b = 1.47$. The resulting function is shown in green in Figure 4.8b. The relation to calculate the PWM signal depending on the desired thrust per motor follows from (4.5) as

$$X = \left(\frac{T}{a}\right)^{\frac{1}{b}}. \tag{4.7}$$

Figure 4.7: Setup of the thrust experiment: the drone is mounted upside down to a scale.



(a) The default PWM/ thrust ratio and the experimental values.

(b) PWM to thrust on a double logarithmic scale with the PWM/thrust estimate.

Figure 4.8: PWM to thrust ratios

4.5 Concluding remarks

In this chapter the characteristics of the Parrot Mambo Fly are described. First it is compared to the previously used Parrot AR.Drone 2.0. The most important difference is the position estimation in the horizontal direction. Thereafter, a breakdown is given of the Simulink model, provided by Parrot and Mathworks, that can be used as the flight control system on the Mambo. Using this model, first experiments are done which have shown that the provided flight control system needs improvement, both in the default controller and the default estimator. Therefore, the controller described in section 3.3 is implemented on the Mambo in further experiments and the estimator is investigated in the next chapter. Finally, experiments are done for system identification and to verify the given parameters. These experiments have shown deviations compared with the given parameters and, therefore, the new measured inertia and thrust ratio could provide improvement in the similarities between simulations and experiments. This improvement is tested in Chapter 6. More of the system parameters can be investigated and identified such as the damping parameters described in section 3.2, which is also done in Chapter 6.

Chapter 5

State Estimation

For the comparison of simulations and experiments, a proper state estimation in the experiments is important. In the simulations we neglect all disturbances for now, thus we can use the model described in section 3.1, where we can extract all output states. In the experiments however, the state estimation requires filtering of the sensor data and a state estimator, since not all states of the quadrotor are measured. This chapter first describes how the state estimation is done in the support package. We discuss the used filters and actions taken in order to estimate the states. Subsequently, this state estimation is compared to the ground truth using the Optitrack Motion Capture System [25]. These experiments have shown non matching results in horizontal positioning and, therefore, the state estimation of the support package needs improvement. This improvement is sought in the optical flow since this is the most important instrument for the position estimation of the Mambo. Gradually a solution for the mismatch is found using a factor on the optical flow that depends on the altitude. This solution provides a proper estimation on the horizontal position, but unfortunately not for certain trajectories with varying heights.

5.1 Default estimator of the support package

The support package provided by Parrot described in section 4.2 contains a default estimator for the state estimation. This section describes the most important characteristics of this estimator. Figure 5.1 shows a schematic overview of the state estimator that is used. The raw sensor data is first processed where the calibration data is used for the gyroscope and the accelerometer measurements. Thereafter, a filter is used that merges accelerometer and gyroscope data to estimate the attitude q and the angular velocity $\dot{q}$. The altitude ρ_z and velocity ν_3 are estimated with a Kalman filter using the data from the ultrasonic sound sensor, the accelerometer, the attitude and the (previous) velocity $\nu_{1,2}$. Thereafter, the optical flow and accelerometer data is used along with the altitude and the attitude to estimate the body fixed velocities $\nu_{1,2}$ with a Kalman filter as well. Finally, this data is used to calculate, mostly by integration, the horizontal position $\rho_{x,y}$. This concludes the state estimation since all

states of (3.1) are covered.

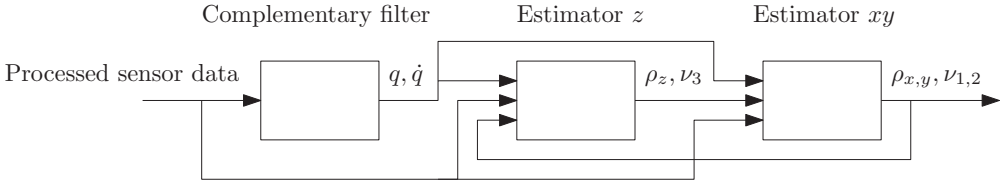


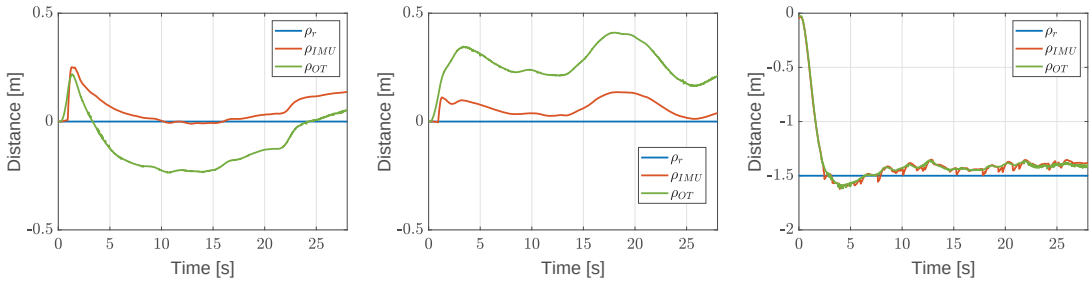
Figure 5.1: Schematic overview of the state estimation on the Mambo

5.2 State estimation vs Optitrack

The first experiments from section 4.3 have implied that the default estimator from the support package provides a poor state estimation. To confirm this statement we want to compare the experimental estimation data with the ground truth. This ground truth is found using the Optitrack Motion Capture System [25]. This system uses a minimum of four cameras to locate and track markers within a certain area. In this case we use eight cameras and apply five tracker balls to the Mambo as shown in Figure 5.2. Usually more trackers are used with Optitrack to increase the consistency and accuracy, but since the drone is relatively small and lightweight only five trackers are used. The fifth tracker in the middle helps the software to estimate the orientation of the drone and should prevent ambiguous configurations. During an Optitrack measurement the software recognizes a rigid body following from the locations and configuration of the trackers. Depending on the number of cameras, the settings and the quality of the calibration, the accuracy of the position measurement is $\leq$ 1mm. The rigid body recognition also enables the possibility to calculate the attitude. Therefore, the Optitrack system gives the position and attitude of the Mambo and we can compare these values with the data estimated by the Mambo, which we refer to as the IMU measurements. Note that by adding the trackers the mass and the mass moment of inertia of the drone increase. The new inertia is determined using Steiner's theorem. These new values are used in the controller for better tracking behavior, but we assume that this does not significantly affect the estimator accuracy.

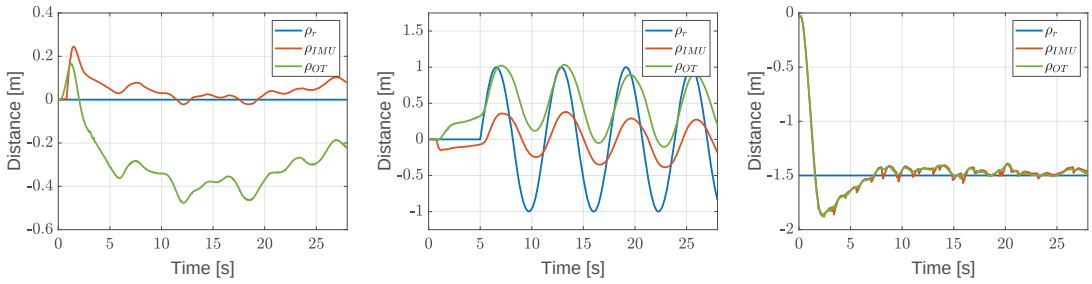
Figure 5.2: Configuration of the Optitrack balls: four in the corners and one in the center.

We first test the default estimator as described in section 5.1 in a hovering experiment. Figure 5.3 shows the experimental results for a constant reference of $\rho_r = \begin{bmatrix} 0 & 0 & -1.5 \end{bmatrix}^T$. From the plot in z direction it can be observed that the estimation of the altitude is done accurately. The plots in x and y direction, however, show a relatively large error. As mentioned, for the estimation in x and y direction the optical flow sensor is used among other sensors. Looking into the estimator of the support package, we observe that the optical flow is only enabled above an altitude of 0.4m. Beneath this value the horizontal velocity is obtained by simply integrating the accelerometer data. Raw accelerometer data is known to often have a drift and therefore this data is not very trustworthy if it is solely used for the positioning. After ascending, however, the horizontal position data neither seems to be accurate. Therefore, another experiment is done using a horizontal trajectory starting after 5 seconds as $\rho_r = \begin{bmatrix} 0 & \sin(t) & -1.5 \end{bmatrix}^T$. The results are shown in Figure 5.4. Again we observe that the altitude estimation accurately follows the Optitrack data. If we look at the horizontal position, however, we observe a significant difference both during take-off and during the rest of the flight. In order to find the critical cause of these errors we look further into the sensor data used for the horizontal position estimation.



(a) Position plots in x-direction (b) Position plots in y-direction (c) Position plots in z-direction

Figure 5.3: Experimental results with the reference signals in blue ρ_r, the estimated values by the default estimator ρ_{IMU} in red and the Optitrack values ρ_{OT} in green.



(a) Position plots in x-direction (b) Position plots in y-direction (c) Position plots in z-direction

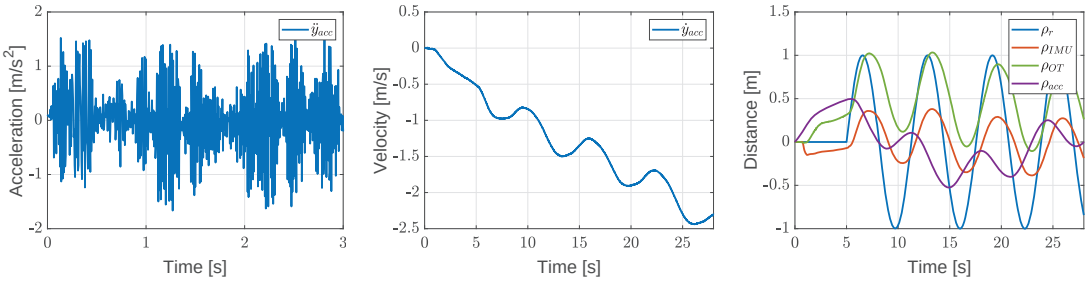
Figure 5.4: Experimental results with the reference signals in blue ρ_r, the estimated values by the default estimator ρ_{IMU} in red and the Optitrack values ρ_{OT} in green.

5.3 Sensor data

The previous section has shown that the default estimator does not suffice for a proper horizontal position estimation. Before trying to improve the estimator itself, we first look into the raw sensor data for irregularities or other bad behavior to conclude whether or not this data can be used. As mentioned in section 5.1, the estimator uses a gyroscope, an accelerometer, an ultrasound sensor, and an optical flow sensor. Unfortunately, the Optitrack measurements have not provided valid experimental data on the attitude due to the low amount of tracker balls and the symmetrical configuration. If we increase this number the dynamics of the Mambo can be compromised. However, previous hovering data has shown very stable behavior in terms of attitude. Therefore, the gyroscope measurements and the attitude estimation are assumed to be sufficiently accurate for now, in addition to the altitude estimation as shown in the previous section.

5.3.1 Accelerometer

One of the important sensors for the position tracking is the accelerometer. This sensor measures the acceleration in the inertial frame. Accelerometers usually contain a lot of noise and an offset. Figure 5.5a shows the raw accelerometer data in y direction for the first experiment from previous section with reference $y_r = \sin(t)$. The default estimator calibrates the accelerometer by calculating the mean values of the first series of time steps and subtracting those values from the measured values every time step to reduce the drift. The gravitational acceleration is also included in this calibration. When the resulting data is integrated to find the velocity, however, a drift still remains on this velocity, as shown in Figure 5.5b. When the constant drift in this velocity data is taken out and the data is integrated, the resulting position as perceived by the accelerometer is shown in Figure 5.5c. This result implies that no actual improvement can be made by using the accelerometer differently as there is no correlation between the double integrated acceleration data and the error in position estimation.



(a) Raw accelerometer data of the x direction.

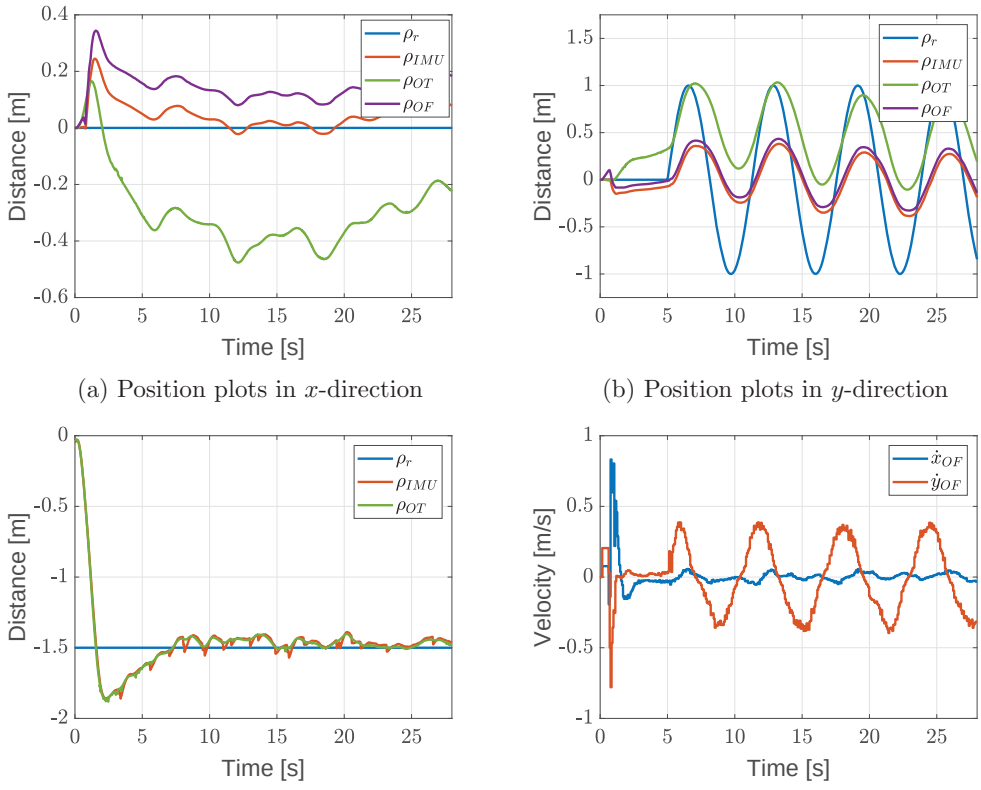
(b) Integrated accelerometer data

(c) Position plots in y-direction including the double integrated acceleration

Figure 5.5: Accelerometer data and the corresponding velocity and position after calibration and integration.

5.3.2 Optical flow sensor

The optical flow sensor gives, following from an algorithm that contains the optical flow theory as described in section 2.2, a signal containing a body fixed velocity in b_1 and b_2 direction. These velocities are then used in a Kalman filter along with accelerometer data and the estimated attitude to estimate the horizontal position. We now take a look at the optical flow data. Figure 5.6 shows the same results as in the previous section, but with the optical flow data shown in Figure 5.6d integrated in the horizontal directions. These results show that the IMU estimation greatly depends on the optical flow data. The ascension is the only part where the optical flow does not strictly follow the IMU. If we compare the data of the IMU/ optical flow and Optitrack, we can observe better similarities after the altitude is settled, which is the case at $\pm$ 6s. The difference seems to be a magnitude error on the velocity. This indicates that a factor on the optical flow might be the solution for the estimation error.



(a) Position plots in x-direction

(b) Position plots in y-direction

(c) Position plots in z-direction

(d) Raw optical flow data

Figure 5.6: Experimental results with the reference signals in blue ρ_r, the estimated values by the default estimator ρ_{IMU} in red, the Optitrack values ρ_{OT} in green and the integrated optical flow ρ_{OF} in purple, except for the raw optical flow data.

5.4 Optical flow factor

Following the findings of the previous section, we search for a factor on the optical flow to decrease the estimation error. Since the take-off is an aggressive maneuver that influences the important sensors, we look at the behavior after take-off and ignore the estimation error caused by the take-off. Therefore, the Optitrack data and integrated optical flow are equalized with the IMU data after $\pm$ 6 seconds in experiments shown in this section, depending on the take-off duration of the specific experiment. First, we take a look at the experimental data used in the previous section. Figure 5.7 shows the same results in y direction, but with the equalized data. We can observe that a single multiplication factor on the optical flow data could work for increasing the resemblance with the Optitrack data. If we now multiply the optical flow with a constant before integrating we can, after trial and error, find the results shown in Figure 5.8. The integrated optical flow now has a high resemblance with the Optitrack measurements.

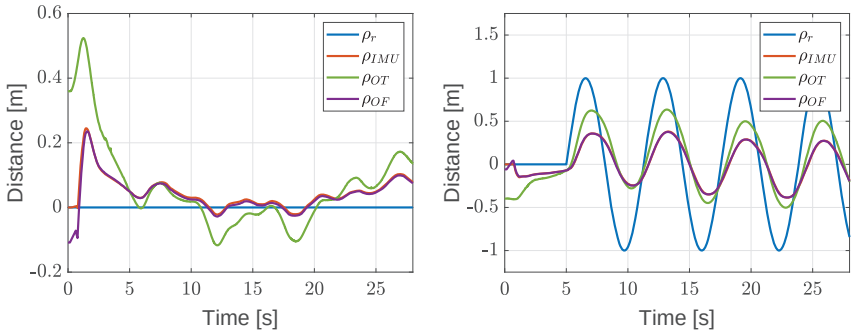
If we apply the same factor to the optical flow in other experiments, a similar improvement is found. However, when we apply this to trajectories at different altitudes, we need a different value for the constant to achieve the same improvement. For example, Figure 5.9 shows the results for a trajectory of

$$\rho_r = \begin{bmatrix} 0 \\ 0.5\sin(t) \\ -2.5 \end{bmatrix}. \tag{5.1}$$

The multiplication factor required for the integrated optical flow values to coincide with the Optitrack values is, just like the altitude, higher. Similarly, for a trajectory at lower altitudes the required factor is lower. These findings imply that the optimal multiplication factor is depending on the altitude. Therefore, we propose the relation

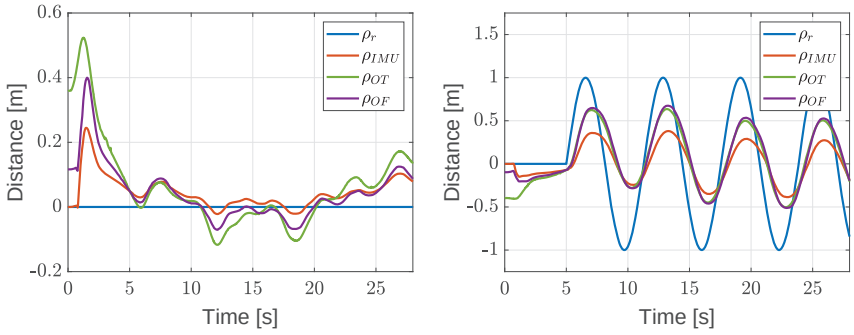
$$\dot{\rho}_{xy} = -\rho_z \dot{\rho}_{xy,s} Q, \tag{5.2}$$

where $\dot{\rho}_{xy,s}$ is the measured optical flow data, which is multiplied by the altitude ρ_z and the optical flow factor Q to calculate the horizontal velocity estimation of the optical flow $\dot{\rho}_{xy}$. Note that the minus sign follows from the NED frame, thus negative values for ρ_z. The value for Q is found by dividing the multiplication factor, found according to the described procedure, over the altitude. When we conduct the described experiment multiple times for altitudes between 0.5 and 2.5 meters, we find an average of $Q = 1.06$. In the next section this proposition with the mentioned value for Q is validated using experimental results from other trajectories.



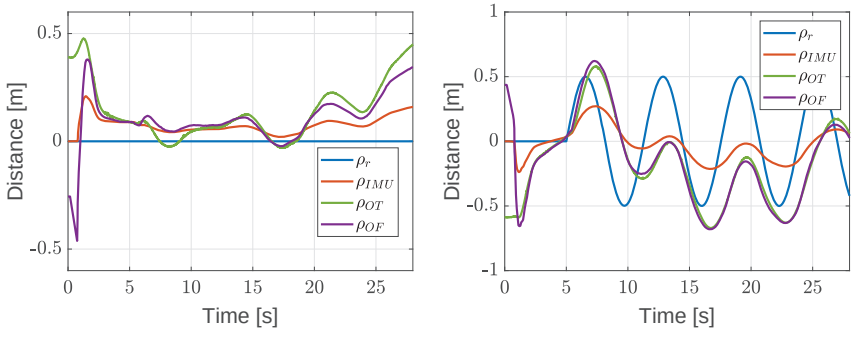
(a) Position plots in x-direction. (b) Position plots in y-direction.

Figure 5.7: Experimental results where the IMU, Optitrack and optical flow data are equalized after 6 seconds. It shows that the IMU and optical flow data are nearly identical.



(a) Position plots in x-direction. (b) Position plots in y-direction.

Figure 5.8: Experimental results with the optical flow data multiplied by a constant. Now the integrated optical flow is almost identical with the Optitrack measurement.



(a) Position plots in x-direction. (b) Position plots in y-direction.

Figure 5.9: Experimental results of a trajectory at 2.5m altitude. This required a higher multiplication factor for the optical flow to acquire a signal similar to the Optitrack measurement.

5.5 Optical flow factor validation

In the previous section we have proposed a solution to the horizontal position estimation error. Now we validate this proposition by implementing (5.2) in the estimation algorithm and by using it on different trajectories. First we test the factor on a circular trajectory as

$$\rho_r = \begin{bmatrix} \cos(t) \\ \sin(t) \\ -1.5 \end{bmatrix}. \tag{5.3}$$

The results are shown in Figure 5.10, including the position estimation error in terms of the root-mean-square (RMS) value. Note that the IMU results now include the proposed factor and therefore we do not show the integrated optical flow. We remain to equalize the Optitrack and the IMU data after take-off. The results show that the introduction of the proposed factor gives the IMU a proper estimation on the horizontal position for a 2D horizontal trajectory. Obviously there is still an error visible, but since the positions are estimated mostly by integrating an estimated velocity and no information of the ground truth is known, the estimation can now be stated to suffice. Similar experiments on other, constant, altitudes have shown matching results.

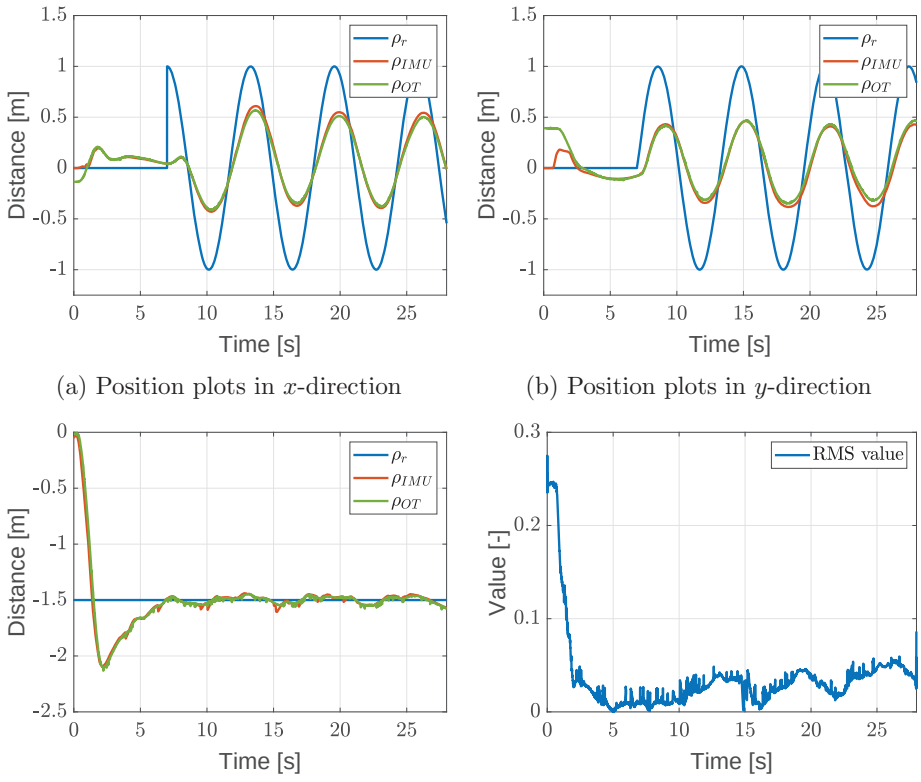
Now that we can conclude that the improved estimator, with the optical flow factor included, suffices for horizontal trajectories, we look into 3D trajectories with varying altitudes. First, we look at the reference trajectory

$$\rho_r = \begin{bmatrix} \sin(t) \\ \cos(t) \\ -1.5 - \sin(t) \end{bmatrix}, \tag{5.4}$$

which results in the behavior shown in Figure 5.11. It can be observed that the estimation of the position in x and z direction is sufficiently accurate. If we look at the y direction, however, there is an increasing error drift between the IMU estimate and the Optitrack measurement, which results in an increasing RMS value. The same sort of drift appears in similar experiments. When the trajectory in x and y direction is switched for example, the same drift appears in x direction. Therefore, we currently cannot conclude that the optical flow factor provides a good estimation for any trajectory. To further illustrate the problem, we conduct two more experiments. First we experiment with the reference trajectory

$$\rho_r = \begin{bmatrix} 0.5\sin(t) \\ 0 \\ -1.5 - 0.5\sin(t) \end{bmatrix}, \tag{5.5}$$

which physically is a trajectory going back and forth on a straight line. The results are shown



(a) Position plots in x-direction

(b) Position plots in y-direction

(c) Position plots in z-direction

(d) RMS value of the estimation error

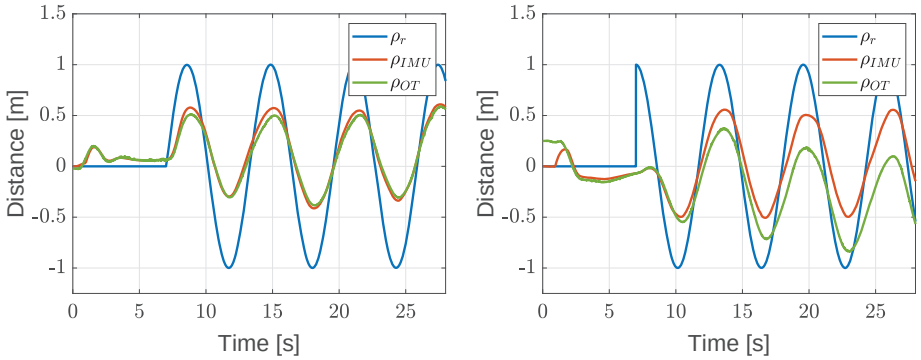
Figure 5.10: Experimental results of a circular trajectory with the optical flow factor included in the IMU estimator.

in Figure 5.12, where the plots in y direction are unnecessary. In this case no drift in the x direction is visible. If we now look at a vertical, circular trajectory as

$$\rho_r = \begin{bmatrix} 0.5\cos(t) \\ 0 \\ -1.5 - 0.5\sin(t) \end{bmatrix}, \tag{5.6}$$

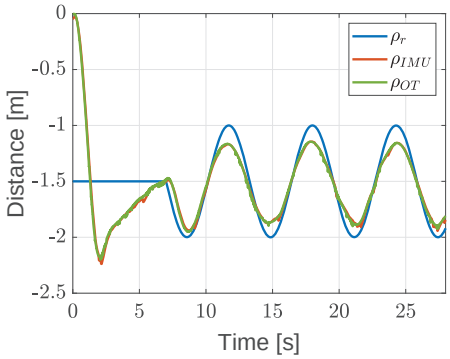
we obtain the results shown in Figure 5.13. Here we can observe that a similar drift as shown in Figure 5.11 occurs. Thus, the error is caused somehow during a vertical displacement. The fact that this error does not occur using a trajectory as (5.5) does not mean that the same error is not present, but it could be a recurring error that cancels itself on the way back. Possible causes for this error are the influence of the attitude, the rotational velocity or the velocity in z direction. To investigate if these are indeed the cause to the error, new experiments are done. These experiments are described in Appendix C and following the findings of these experiments, it can be concluded that these are not the causes of the error. However, these experiments have shown that there is a possible error in the optical flow factor (5.2). Therefore, new proposals are made and tested to include the altitude in the

optical flow, by making the optical flow factor Q also dependent on the altitude, both linearly and exponentially. Unfortunately, those proposals are rejected after some experiments as well.

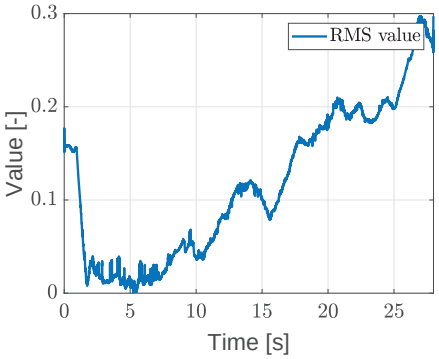


(a) Position plots in x-direction

(b) Position plots in y-direction

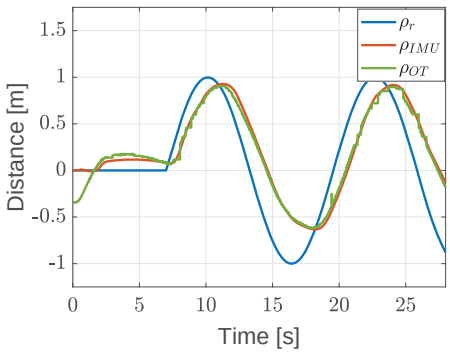


(c) Position plots in z-direction

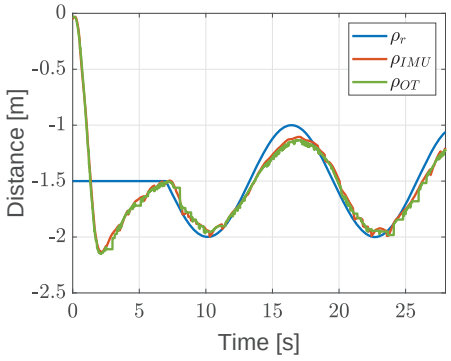


(d) RMS value of the estimation error

Figure 5.11: Experimental results of a 3D trajectory with the optical flow factor included in the IMU estimator.

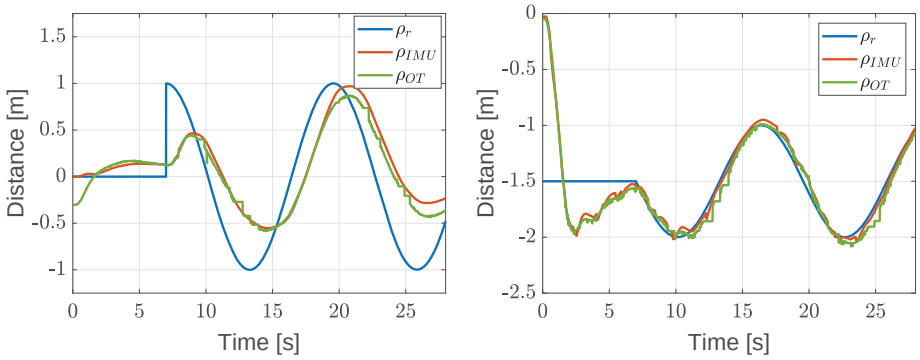


(a) Position plots in x-direction



(b) Position plots in z-direction

Figure 5.12: Experimental results of a trajectory following a straight line where no visible drift of the IMU data w.r.t. the Optitrack data in x-direction occurs.



(a) Position plots in x-direction (b) Position plots in z-direction

Figure 5.13: Experimental results of a vertical, circular trajectory where a visible drift of the IMU data w.r.t. the Optitrack data in x-direction occurs.

5.6 Concluding remarks

This chapter has first shown how the states of the Mambo are estimated using the default estimator of the Simulink support package provided by Parrot. This estimator is tested and compared to measurements using the Optitrack motion capture system. These experiments have shown that the altitude estimation using the gyroscope and the ultrasound sensor is sufficiently accurate. However, for the horizontal position estimation, which mainly uses optical flow data, it does not suffice. Therefore, the sensor data is analyzed and an improvement is found by introducing an optical flow factor, which includes the altitude in the optical flow data. Experiments have shown that this factor made significant improvement in the horizontal position estimation. However, on certain trajectories with varying altitudes a drift has occurred. This drift is studied but unfortunately this recurring problem is not resolved. Therefore, we conclude that the improved estimator, including the optical flow factor, can be used for experiments on the Mambo for trajectories with a constant altitude. Note that a perfect horizontal position estimation is nearly impossible with the available sensors, since the position is found by filtering and integrating acceleration and velocity measurements, thus not compared to a ground truth.

Chapter 6

Simulations and Experiments

The main goal of this thesis, as described in Chapter 1, is to acquire a simulation model that has coincident results with experimental results. First we have described a simulation model in Chapter 3 where we have also discussed the drag forces acting on the drone. However, before we can compare the simulation results with experimental results the default estimator of the drone has had to be improved in order to acquire proper data. The previous chapter has shown that for certain trajectories the experimental results are trustworthy and can, therefore, be used in a comparison with simulation results. In this chapter we first show the influence of the empirically found thrust ratio and mass moment of inertia, compared to the values initially given in the Simulink support package. After we assume that the new thrust ratio is correct, we continue with our final step: including drag forces in the simulation model to reduce the gap between the simulations and experimental results. The individual drag forces are not determined separately, but they are included as lumped parameters that act on the drone depending on the horizontal velocity as described in section 3.2.

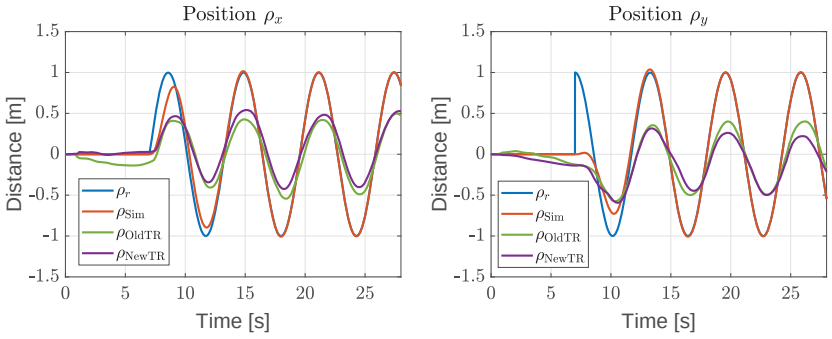
6.1 PWM to thrust ratio

In section 4.4.2 we have described how the PWM signal for the required thrust per motor is determined. This has resulted in the ratio

$$X = \left(\frac{T}{4.29 * 10^{-5}} \right)^{\frac{1}{1.47}}, \tag{6.1}$$

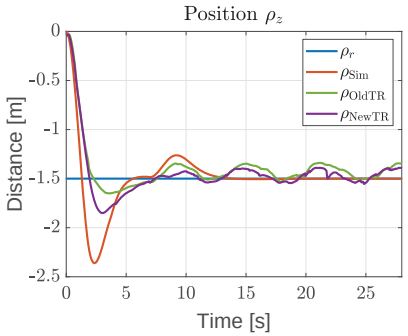
instead of the linear ratio $X = 1530.7 * T$ that is given in the Simulink support package. To evaluate this new thrust ratio, we want to compare simulation results with both the old and new thrust ratio in experiments. In Chapter 5 we have stated that the improved estimator is trustworthy using the optical flow factor when flying at a constant altitude. Therefore,

we first test with the circular reference (5.3). For the simulation model we use the model described in Chapter 3. Figure 6.1 shows the simulation results and both the experimental results using the provided and the determined thrust ratio. The experiments are much alike and both do not match the simulation. However, the experiment with the new thrust ratio shows a higher overshoot during takeoff compared to the old ratio, which is more in line with the simulation. This trajectory requires a constant thrust after take-off and due to the integration action in the controller a constant error in the thrust ratio should be eliminated over time. Thus, these experiments cannot conclude whether or not the new thrust ratio gives a better approximation of the actual ratio than the old, given ratio. Therefore, we experiment with a more aggressive reference in z direction as $\rho_r = \begin{bmatrix} 0 & 0 & -2 - \sin(2t) \end{bmatrix}^T$. Note that during this experiment the estimation in x and y direction is not very trustworthy due to the varying altitude. However, since the horizontal movement is minimal in this case, we assume that it does not influence this experiment. The results are shown in Figure 6.2 where there is a clear difference between the used thrust ratios. The new thrust ratio shows a better tracking behavior and is more coincident with the simulation. This can be observed during the high accelerations: both at the lowest altitude where the required thrust is very high and at the top where the required thrust is relatively low. This experiment uses a wide range of thrust and, therefore, we can conclude that the new thrust to PWM ratio is an improvement compared to the provided, linear ratio. From here on out we only use the new thrust ratio in experiments.



(a) Position plots in x-direction

(b) Position plots in y-direction



(c) Position plots in z-direction

Figure 6.1: Experimental and simulation results of a circular trajectory with both the old and the new thrust ratio.

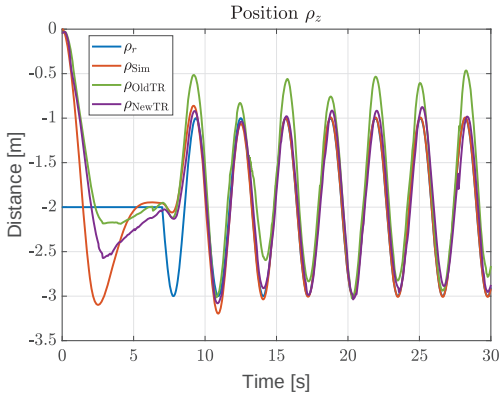


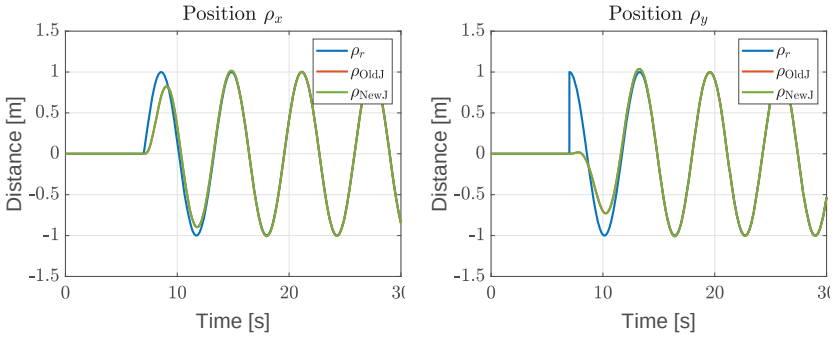
Figure 6.2: Experimental and simulation results of a sine trajectory in z direction.

6.2 Mass moment of inertia

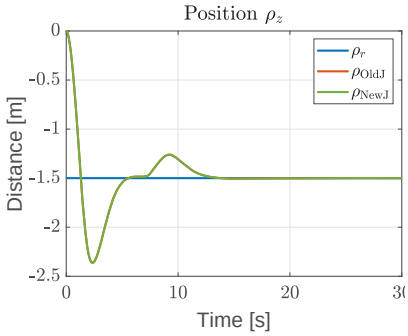
Another part of the system identification is the mass moment of inertia that is determined in section 4.4.1. The results have shown that the determined inertia is 19, 8 and 50 percent higher for the J_{xx}, J_{yy} and J_{zz} direction, respectively, compared to the values stated in the support package. The inertia is used in both the controller and the system (3.1) of the simulation, but when we experiment to find the correct inertia we only need to change the inertia in the system (3.1) of the simulation model. We would compare these simulation results to experimental results and hope to find the new inertia to have better matching results. However, this change in inertia does not have a large impact on the behavior, as shown in Figure 6.3 where the resulting behavior of both simulations is nearly identical. Only when the aggressiveness of the trajectory is significantly increased, there are bigger differences in the behavior. However, if we would test this influence in aggressive experiments, more drag-like parameters become influential and, therefore, it would be difficult to identify what inertia is a better approximation. Consequently, for now we cannot conclude whether or not the new inertia better represents the experimental setup. However, we use the new inertia from here on out, because we suspect that the provided inertia is measured without the bumpers.

6.3 Including drag in simulation model

So far we have only used the frictionless model (3.1) in the simulation which obviously does not match the experimental results. To include this damping, we can identify all different drag forces individually as described in section 3.2. However, we can also use the lumped parameter K_c acting on the drone for horizontal velocities as stated in (3.13) to calculate the total drag force acting on the drone F_D. This includes the effects of blade flapping, induced drag, translational drag and profile drag. Assuming that the rest of the model is correct, this means that we only need to tune the lumped parameter K_c in order to reduce the gap between the simulation results and experimental results. This is done similarly to the tuning of the optical flow factor, thus by trial and error and checking this value in other simulations and



(a) Position plots in x-direction (b) Position plots in y-direction

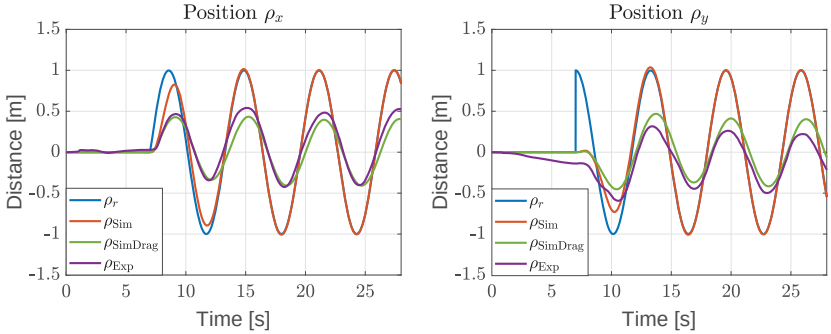


(c) Position plots in z-direction

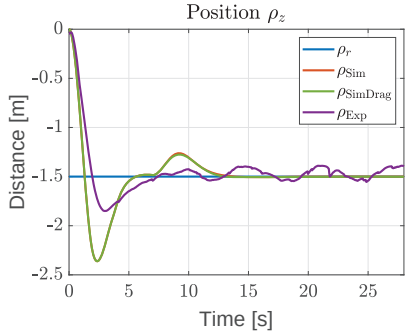
Figure 6.3: Simulation results of a circular trajectory with both the old and the new inertia. The red (old inertia) line is nearly identical to the green (new inertia) line.

experiments. If we apply this to the experiments with reference (5.3) as in Figure 6.1 and we start increasing the value for k_c in K_c, we eventually get the results shown in Figure 6.4 for a value of $k_c = 0.22$. Herein the drag included model has, compared to the frictionless model in red, a high resemblance with the experimental values. This improvement is only visible in horizontal direction, as the drag forces in F_D are currently only formulated in horizontal direction of the body-fixed frame. The results show a great improvement in terms of the simulation representing the experimental results. In order to verify the value of $k_c = 0.22$ we experiment with other trajectories. Figure 6.5 shows the results using a step function on the x reference. Herein, we can once again observe a major improvement in the similarities between the simulation and the experiment in horizontal direction when the drag is included. Finally, we experiment with the same circular trajectory as before, only at double velocity, resulting in the behavior shown in Figure 6.6. Apart from small irregularities in the horizontal tracking behavior, it can be observed that also for higher velocities the constant value of $k_c = 0.22$ as lumped parameter for modeling drag forces provides a good approximation. In all experiments that are shown in this section we do not observe a significant improvement in z direction, which makes sense because we have modeled the drag forces only in the body fixed directions b_1 and b_2. Experiments on the thrust ratio show, in Figure 6.2, that a trajectory in z direction can be flown with a minimal tracking error. Only during takeoff, or in other words a step function, the simulations show a higher overshoot. During a vertical trajectory the error after takeoff is minimal, which is most likely the results of the integration action of the controller.

Another cause can be the unmodeled ground effect. During the circular trajectory, we can also observe small periodic oscillations in the vertical direction, where the first excitation is also present in the simulations. Other simulations have shown that this first excitation is caused by the integration action in the controller. For the rest of the oscillations in experimental flights this could be a recurring error with this integration action, but we cannot give a definitive cause for this event.

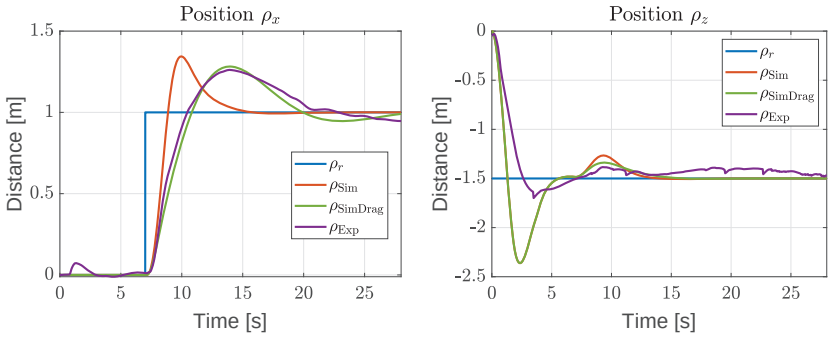


(a) Position plots in x-direction (b) Position plots in y-direction



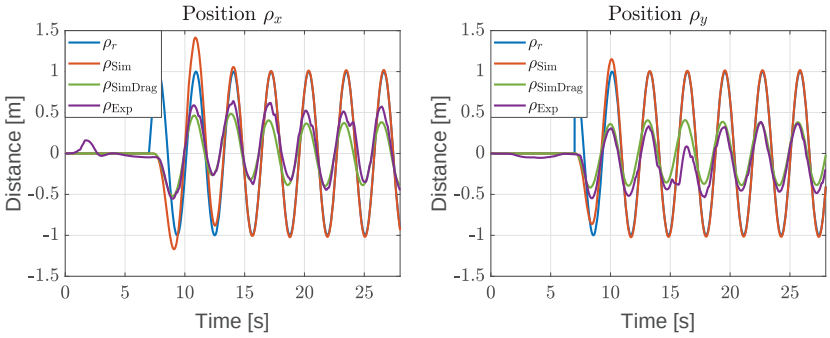
(c) Position plots in z-direction

Figure 6.4: Experimental and simulation results of a circular trajectory with and without the drag included in the model, with $k_c = 0.22$.



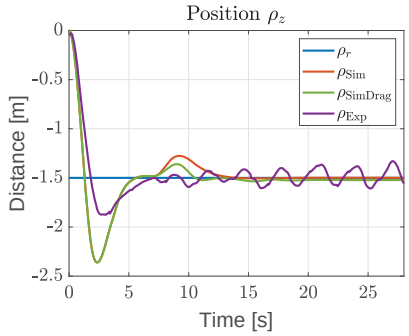
(a) Position plots in x-direction (b) Position plots in z-direction

Figure 6.5: Experimental and simulation results of a step response in x direction with and without the drag included in the model, with $k_c = 0.22$.



(a) Position plots in x-direction

(b) Position plots in y-direction



(c) Position plots in z-direction

Figure 6.6: Experimental and simulation results of a circular trajectory with and without the drag included in the model, with $k_c = 0.22$.

6.4 Concluding remarks

This chapter contains the description on how the empirically determined thrust ratio and mass moment of inertia have been validated. For the thrust ratio we can conclude that the new, determined thrust ratio provides a better approximation of the actual ratio. Regarding the mass moment of inertia, no significant improvement is found in the simulations. However, since the actual mass is also higher than claimed in the support package, we assume that the mass moment of inertia should be higher as well. Therefore, we assume the new, measured, inertia as a better approximation of the Mambo. Finally, the drag forces as described in section 3.2 are included in the simulation model. After some tuning of the lumped parameter k_c we have observed how the resemblance of the simulations and experiments in horizontal direction is increased, which is a great improvement towards achieving a representative simulation model of the drone. In the vertical direction the simulations and experiments show similar results when following a vertical trajectory. During takeoff or during a horizontal trajectory, however, the experiments show a recurring error in the vertical direction.

Chapter 7

Conclusions and Recommendations

In the path of reaching the goal to have multiple drones fly both autonomously and simultaneously, this thesis has described several steps in the process. The main goal of this thesis is to increase the resemblance of the simulation model and the experimental setup, which is crucial for further research. This is done mainly by including external forces into the simulation model. Issues with the state estimator of the Parrot Mambo Fly drone are first resolved. Within the given time frame, it has not been possible to investigate each of the potential influential parameters in depth. However, using a lumped parameter for the drag forces in horizontal direction a significant improvement is made. In this chapter we discuss the steps taken in the process and draw conclusions based on our findings. Subsequently, we present our recommendations for future work.

7.1 Conclusions

In this section we discuss the modeling of a quadrotor, the system properties and details of the Parrot Mambo Fly, and finally how the gap between the simulations and experiments is notably decreased.

Modeling

For the modeling part of the drone we have used the literature to derive the model of the quadrotor. A simple model has been constructed, using quaternions for the attitude representation. Even though this quaternion approach is not yet necessary since the use of rotation matrices suffices, for future work this can make a difference in terms of calculation efficiency. The claimed disadvantage of quaternions, being the ambiguity of a rotational expression, has not been a problem during this research, as a controller derived on $SO(3)$ is used. Subsequently, we have discussed the important external forces acting on the drone, mainly being

the effects of blade flapping, induced drag, translational drag and profile drag. Most of these effects depend on the geometry of the quadrotor and its blades. These effects are described with a lumped parameter, since they all mainly depend linearly on the horizontal velocity and the approximation of each effect individually would be very difficult.

For the control of the Mambo, a cascaded controller that uses a virtual input to follow a certain trajectory is used. Since this controller has already been proven to be UaGAS, the controller does not need improvement in terms of stability. However, even experiments with low velocity trajectories have shown that the tracking error dynamics are not nearly as good as the simulations. For the main goal of this thesis the performance of the controller is not very important, since we aim to have a representative simulation model.

The Parrot Mambo Fly

Previous research on quadrotors within the Dynamics & Control research group in Eindhoven has been conducted on the Parrot AR.Drone 2.0. This drone has been taken out of production and, therefore, we have started using the Parrot Mambo Fly. The Simulink support package that is used for experimenting with this drone includes parameters for the mass, mass moment of inertia and a thrust ratio of the drone. However, experiments have learned us that these values are not correct. The new, determined values are verified by comparing the simulations and experiments. In the new thrust to PWM ratio, which is exponential instead of linear, we have observed a significant improvement when a wide range of thrust is needed. Following these findings we have concluded that the new thrust ratio provides a better approximation of the actual ratio. Considering the mass of the drone, we have found that the value provided in the support package is actually lower than the weighted mass. For the mass moment of inertia, we have not found a significant difference/ improvement in the simulations using the determined values. However, also considering the difference in mass, we suspect that the values provided in the support package are approximated without the bumpers of the Mambo. Therefore, we assumed that the experimentally determined values for the mass and the mass moment of inertia provide a better approximation of the experimental setup.

During the first experiments with the Mambo, we found that the default estimator of the flight control system from the support package provides a poor state estimation. Using the Optitrack camera system we have confirmed that this estimator provides a good approximation of the altitude and attitude, but in the horizontal direction the position approximation is inadequate. Therefore, we have looked further into the sensor signals that are used in the estimator. The optical flow sensor is by far the most important sensor for the horizontal position estimation and by using the Optitrack system we have been able to compare the estimator output with the ground truth. Studying the approximation error and the possible solutions, we have introduced an optical flow factor. Using this factor we include the altitude in the optical flow data. Experiments on different altitudes have shown a significant improvement in the horizontal position estimation. However, during certain trajectories with a varying altitude a small drift in the estimation w.r.t. the ground truth occurs. This drift is unfortunately not taken care of, but compared to the initial estimator we have found a serious improvement. Therefore, we conclude that, using the optical flow factor in the estimator, the

Mambo can be used for experiments on a constant altitude. Consequently, we can compare the experimental results with simulation results.

Simulations and Experiments

In order to reach the final goal of this thesis, which is to acquire a representative simulation model of the Mambo, drag forces are included into the model. This is done by including a lumped parameter for the different external forces acting on the drone, mostly depending on the horizontal velocity. After we have included the parameter in the simulation model, we have tuned this parameter to reduce the gap between the simulation results and experimental results. When this determined parameter is used in other simulations and experiments, i.e., with other trajectories, we have found a significant improvement in the resemblance of the results there as well. However, this improvement is found mainly in the horizontal direction. In vertical direction there remains to be a mismatch between the simulations and experiments. During takeoff, i.e., the first part of a flight, the experimental velocity is lower than the velocity during simulation. This can be caused by an error in the thrust ratio or by other unmodeled behavior in vertical direction. However, after takeoff the Mambo can follow an aggressive trajectory in vertical direction. This implies that this error is taken care of by the integrating action of the controller.

To summarize, we can conclude that the goal of this thesis is achieved to a certain extend. The simulation model now provides a good approximation of the reality in horizontal direction. In vertical direction, however, a small mismatch remains.

7.2 Recommendations

In this final section recommendations are presented for future work. It is very important to note that Parrot has recently stopped the production of the Mambo and other small drones [29]. Consequently, some of these recommendations need consideration if they are only applicable for this specific drone.

- One of the main issues encountered during this research is the poor estimation of the horizontal velocity. Using the proposed optical flow factor a major improvement is found. However, for a trajectory with a varying altitude there still is a drift occurring in the horizontal position estimation. Since the most logical causes/ solutions in the optical flow data are already covered in this thesis, we recommend looking for a solution in another direction. A possible solution would be to implement the Optitrack data live onto the drone. In this research the Optitrack data is used after the experiment to compare with the estimated values of the IMU. However, this data can also be integrated into the flight control system for a very accurate position estimation. Another possible solution is to use the image data from the bottom camera. This data is easily converted into RGB values and can then be used with specific markers on the ground,

which the drone can identify and use to estimate its position. However, since Parrot stopped the production of the Mambo it can be a bad decision to improve this specific estimator. Especially since the use of the optical flow factor seems to be a solution for this specific case rather than a general solution for other drones as well. Thus, we do not recommend having major changes in the estimator anymore, unless these solutions are also applicable to other drones.

- During the experiments we found some errors in the calibration of the drone. On every flight, it claims to calibrate the sensors by checking the values of the first time steps. However, experimental data shows that the gyroscope has a constant error, where another drone of the same type has a different constant error. In this research we have manually treated this problem by identifying the error value to compensate this problem. In future research the software can include a (continuous) calibration algorithm to not have this problem on a new drone.

- For the modeling of a quadrotor we have shown how the use of a lumped parameter for the horizontal drag forces can make a significant improvement in the accuracy of the model, compared to experimental results. Most of these forces depend on the geometry of the rotors and are, therefore, difficult to approximate individually. Hence, the use of experimental data and the lumped parameter K_c provides a simple method to approximate these forces all together, which should also work on other quadrotors. In vertical direction, however, a small mismatch remains. This mismatch could also be caused by drag/ aerodynamic forces and should be investigated in further research. An approach similar to the work presented in this thesis for the horizontal drag can be used for the investigation of the forces acting in the vertical direction. Hence, use the literature to find the influential parameters and their dependence on the velocities. Thereafter, this parameter can be added to the simulation model and tuned to reduce the gap between the simulation results and the experimental results.

- Now that we, up to a certain level, have modeled the drag forces in horizontal direction, we can use this knowledge of external forces in our controller in terms of feedforward. Since we can predict the external force acting on the drone horizontally we can include these forces in the thrust that needs to be delivered. The controller that is applied works with a virtual input to determine the desired thrust vector. Therefore, we can include the estimated drag forces into the control action, simply by including this drag function into the virtual input described in section 3.3. This should improve the tracking behavior of the Mambo.

Bibliography

[1] G. Allibert, D. Abeywardena, M. Bangura, and R. Mahony. Estimating body-fixed frame velocity and attitude from inertial measurements for a quadrotor vehicle. In *2014 IEEE Conference on Control Applications (CCA)*, pages 978–983. IEEE, oct 2014.

[2] M. Bangura. *Aerodynamics and Control of Quadrotors*. PhD thesis, Australian National University, Canberra, 2017.

[3] A. Briod, J.-C. Zufferey, and D. Floreano. Optic-Flow Based Control of a 46g Quadrotor. *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[4] P.-J. Bristeau, F. Callou, D. Vissière, and N. Petit. The Navigation and Control technology inside the AR.Drone micro UAV. *IFAC Proceedings Volumes*, 44(1):1477–1484, jan 2011.

[5] G. de Croon, H. Ho, C. De Wagter, E. van Kampen, B. Remes, and Q. Chu. Optic-Flow Based Slope Estimation for Autonomous Landing. *International Journal of Micro Air Vehicles*, 5(4):287–297, 2014.

[6] H. de Kleuver. Extended non linear dynamics for quadrotor control in aggressive maneuvering. Master's thesis, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, DC 2017.092, Eindhoven, 2017.

[7] J. Diebel. Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors. Technical report, Stanford University, Stanford. https://www.astro.rug.nl/software/kapteyn-beta/_downloads/attitude.pdf, 2006.

[8] D. Eberly. Rotation Representations and Performance Issues. Technical report, Geometric Tools, Redmond. https://www.geometrictools.com/Documentation/RotationIssues.pdf, jan 2002.

[9] M. Faessler, A. Franchi, and D. Scaramuzza. Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories. *IEEE Robotics and Automation Letters*, 3(2):620–626, 2017.

[10] I. Fantoni and G. Sanahuja. Optic Flow-Based Control and Navigation of Mini Aerial Vehicles. *AerospaceLab*, 1(8):8–3, 2015.

[11] E. Fresk and G. Nikolakopoulos. Experimental evaluation of a full quaternion based attitude quadrotor controller. In *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, volume 2015-Octob, pages 1–4. IEEE, sep 2015.

[12] L. R. García Carrillo, A. E. Dzul López, R. Lozano, and C. Pégard. *Quad Rotorcraft Control*. Advances in Industrial Control. Springer London, London, 2013.

[13] J. Gibson and O. Marques. *Optical Flow and Trajectory Estimation Methods*. SpringerBriefs in Computer Science. Springer International Publishing, Cham, 2016.

[14] R. Goldman. Rethinking Quaternions. *Synthesis Lectures on Computer Graphics and Animation*, 4(1):1–157, oct 2010.

[15] M. E. Guerrero-Sánchez, H. Abaunza, P. Castillo, R. Lozano, and C. D. García-Beltrán. Quadrotor Energy-Based Control Laws: a Unit-Quaternion Approach. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 88(2-4):347–377, 2017.

[16] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys. An Open Source and Open Hardware Embedded Metric Optical Flow CMOS Camera for Indoor and Outdoor Applications. *2013 IEEE International Conference on Robotics and Automation*, pages 1736–1741, 2013.

[17] M. R. Jardin and E. R. Mueller. Optimized Measurements of Unmanned-Air-Vehicle Mass Moment of Inertia with a Bifilar Pendulum. *Journal of Aircraft*, 46(3):763–775, may 2009.

[18] N. Jeurgens. Identification and Control Implementation of an AR.Drone 2.0. Master's thesis, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, DC 2017.013, Eindhoven, 2017.

[19] F. Kendoul, I. Fantoni, and K. Nonami. Optic flow-based vision system for autonomous 3D localization and control of small aerial vehicles. *Robotics and Autonomous Systems*, 57(6-7):591–602, jun 2009.

[20] E. Lefeber, S. van den Eijnden, and H. Nijmeijer. Almost global tracking control of a quadrotor UAV on SE(3). In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1175–1180. IEEE, dec 2017.

[21] J. Leishman. *Principles of Helicopter Aerodynamics*. Cambridge University Press, 2002.

[22] R. Mahony, V. Kumar, and P. Corke. Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor. *IEEE Robotics & Automation Magazine*, 19(August):20–32, 2012.

[23] A. Nair. Extended Modeling and Control of Quadrotors using System Identification. Master's thesis, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, DC 2019.057, Eindhoven, 2019.

[24] F. Oliva-Palomo, A. Sanchez-Orta, P. Castillo, and H. Alazki. Nonlinear ellipsoid based attitude control for aggressive trajectories in a quadrotor: Closed-loop multi-flips implementation. *Control Engineering Practice*, 77(June):150–161, 2018.

[25] Optitrack. Motion Capture System. https://optitrack.com/.

[26] C. Rucker. Integrating Rotations Using Nonunit Quaternions. *IEEE Robotics and Automation Letters*, 3(4):2979–2986, oct 2018.

[27] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics, Modelling , Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer London, London, 2009.

[28] A. Tayebi and S. McGilvray. Attitude stabilization of a VTOL quadrotor aircraft. *IEEE Transactions on Control Systems Technology*, 14(3):562–571, may 2006.

[29] The Verge. Parrot exits the toy drone market. https://www.theverge.com/2019/7/19/20699905/parrot-exit-toy-drone-market-dji-consumers.

[30] S. van Boheemen. Experiment-based Modelling of Aerodynamic Effects on a Quadrotor. Bachelor's thesis, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, DC 2018.068, Eindhoven, 2018.

[31] S. van den Eijnden. Cascade Based Tracking Control of Quadrotors. Master's thesis, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, DC 2017.012, Eindhoven, 2017.

Appendix A

Controller equations in quaternions

The work of [31] and [20] has described the dynamics and the controller design in terms of rotation matrices. Since the same controller is used in this thesis, only with a quaternion representation of the attitude, most of the equations can easily be rewritten according to the multiplication and rotation rules described in section 2.1.3. However, since the desired rotation matrix R_d of [20] depends on symbolic relations we want to write the corresponding quaternion in the most simple way possible. We start from

$$R_d = \begin{bmatrix} 1 - \frac{f_{d1}^2}{1+f_{d3}} & -\frac{f_{d1}f_{d2}}{1+f_{d3}} & f_{d1} \\ -\frac{f_{d1}f_{d2}}{1+f_{d3}} & 1 - \frac{f_{d2}^2}{1+f_{d3}} & f_{d2} \\ -f_{d1} & -f_{d2} & f_{d3} \end{bmatrix}, \tag{A.1}$$

and we use the rule for rewriting a rotation matrix in terms of a quaternion [7]

$$q_d = \frac{1}{2} \begin{bmatrix} 2 * \cos(\theta/2) \\ \text{sign}(R_{32} - R_{23}) * |\sqrt{1 + R_{11} - R_{22} - R_{33}}| \\ \text{sign}(R_{13} - R_{31}) * |\sqrt{1 - R_{11} + R_{22} - R_{33}}| \\ \text{sign}(R_{21} - R_{12}) * |\sqrt{1 - R_{11} - R_{22} + R_{33}}| \end{bmatrix}, \tag{A.2}$$

with $\theta = \cos^{-1}((\text{trace}(R) - 1)/2)$. Furthermore, we use the property $f_{d1}^2 + f_{d2}^2 + f_{d3}^2 = 1$ that follows from (3.21). First we calculate the angle

$$
\begin{aligned}
\theta =& \cos^{-1}(R_{11}+R_{22}+R_{33}-1)/2) \\
=& \cos^{-1}((1-\frac{f_{d1}^2}{1+f_{d3}}+1-\frac{f_{d2}^2}{1+f_{d3}}+f_{d3}-1)/2) \\
=& \cos^{-1}((1-\frac{f_{d1}^2}{1+f_{d3}}-\frac{f_{d2}^2}{1+f_{d3}}+f_{d3})/2) \\
=& \cos^{-1}((\frac{1+f_{d3}-f_{d1}^2-f_{d2}^2+f_{d3}+f_{d3}^2}{1+f_{d3}})/2) \\
=& \cos^{-1}((\frac{2f_{d3}+2f_{d3}^2}{1+f_{d3}})/2) \\
=& \cos^{-1}(f_{d3}).
\end{aligned}
\tag{A.3}
$$

We can calculate the second term of q_d as

$$
\begin{aligned}
q_d(2) =& \frac{1}{2}\text{sign}(R_{32}-R_{23}) * |\sqrt{1+R_{11}-R_{22}-R_{33}}| \\
=& \frac{1}{2}\text{sign}(-2f_{d2}) * |\sqrt{1+1-\frac{f_{d1}^2}{1+f_{d3}}-1+\frac{f_{d2}^2}{1+f_{d3}}-f_{d3}}| \\
=& \frac{1}{2}\text{sign}(-f_{d2}) * |\sqrt{\frac{1+f_{d3}-f_{d1}^2+f_{d2}^2-f_{d3}-f_{d3}^2}{1+f_{d3}}}| \\
=& \frac{1}{2}\text{sign}(-f_{d2}) * |\sqrt{\frac{2f_{d2}^2}{1+f_{d3}}}| \\
=& \frac{1}{2}\text{sign}(-f_{d2}) * |f_{d2}\sqrt{\frac{2}{1+f_{d3}}}| \\
=& -\frac{1}{2}f_{d2}\sqrt{\frac{2}{1+f_{d3}}},
\end{aligned}
\tag{A.4}
$$

where the last expression can be written due to the property $|f_{d3}| \leq 1$. The third term of q_d can be written similarly as

$$
\begin{aligned}
q_d(3) =& \frac{1}{2}\text{sign}(R_{13}-R_{31}) * |\sqrt{1-R_{11}+R_{22}-R_{33}}| \\
=& \frac{1}{2}f_{d1}\sqrt{\frac{2}{1+f_{d3}}}.
\end{aligned}
\tag{A.5}
$$

Finally, we can immediately see that the last term $q_d(4) = 0$ because $\text{sign}(R_{21}-R_{12}) = \text{sign}(0) = 0$. Thus, the desired attitude can be written as

$$q_d = \frac{1}{2}\begin{bmatrix} 2\cos(\cos^{-1}(f_{d3})/2) \\ -f_{d2}\sqrt{\frac{2}{1+f_{d3}}} \\ f_{d1}\sqrt{\frac{2}{1+f_{d3}}} \\ 0 \end{bmatrix}. \tag{A.6}$$

Now with the property of q_d being a unit quaternion because it describes a rotation, this can also be rewritten into

$$q_d = \begin{bmatrix} \frac{1}{2}\sqrt{2+2f_{d3}} \\ -\frac{f_{d2}}{\sqrt{2+2f_{d3}}} \\ \frac{f_{d1}}{\sqrt{2+2f_{d3}}} \\ 0 \end{bmatrix}, \tag{A.7}$$

or as $q_d = \frac{q_{d0}}{||q_{d0}||}$ with

$$q_{d0} = \begin{bmatrix} 1+f_{d3} \\ -f_{d2} \\ f_{d1} \\ 0 \end{bmatrix}, \tag{A.8}$$

which concludes the derivation of the desired attitude in quaternions needed for the controller described in section 3.3.

Appendix B

Parrot Minidrone Mambo Manual

This manual is a quick start guide to use the Parrot Minidrone Mambo with Simulink. Note that you need a Bluetooth CSR 4.0 compatible chip set. To check if your laptop is compatible, follow the instructions from
`https://bit.ly/2NrpFzS`
If your laptop is compatible, you can download the correct driver from
`https://bit.ly/2H21Z48`
If your laptop is not compatible, you will need a Bluetooth dongle.

First time connection

For the first time connection with the Parrot Minidrone, follow these instructions:

1. Install the Simulink support Package for the Parrot Minidrones according to
   `https://nl.mathworks.com/help/supportpkg/parrot/ug/install-support-for-parrot-minidrone.html`

Step 1 will automatically launch the setup wizard of connecting the drone. Otherwise this can be accessed clicking the gear icon from the 'Manage Odd-Ons' menu. The setup wizard will explain the steps to connect the drone with your laptop. Note: if the setup of the current drone is done before, the steps to install the firmware on the drone can be skipped. It will later check whether the firmware is installed correctly.

2. Install the RNDIS for Parrot Minidrones according to
   `https://bit.ly/2Sv23eA`
3. The wizard will tell you how to connect with the drone via Bluetooth, otherwise follow the instructions from

https://bit.ly/2EhxS5t
Note: in Windows 10 the correct icon for the Minidrone is not a gamepad, so there should be two icons with identical names. Try the connecting steps for both and find the correct one.

The setup wizard from Simulink can now check whether the drone is correctly connected via Bluetooth to use the Simulink toolbox. Once the connection is setup for the first time, the drone can be connected simply by using the final actions of step 3: Join a Personal Area Network and connect to the drone.

Building flightControlSystem onto the drone

Now that the Minidrone is connected correctly, Simulink can be used to fly and experiment with the drone. To get used to building the software onto the drone, follow
https://bit.ly/2Ex7A0C for a quick starters guide on controlling the drone.

To start with a clean implementable prepared file for both simulating and experimenting, enter **asbQuadcopterStart** into the command window. This will open a Simulink model including a 3D simulation environment which can also be use to build the software on the drone. The input and output methods for the simulation can be chosen in the 'Commands' and 'Visualization' boxes. Build the system on the drone follow the instructions from
https://bit.ly/2Tl14C8
Note: make sure you build the model from the flightControlSystem (FCS) block.

Other useful links

Setup and configuration
https://bit.ly/2ExvY21

An extended tutorial for first flights.
https://youtu.be/qPVnweXMguE

Obtaining log files.
https://bit.ly/2SuUmW5

Possible Errors in the log files.
https://bit.ly/2GNDZCs

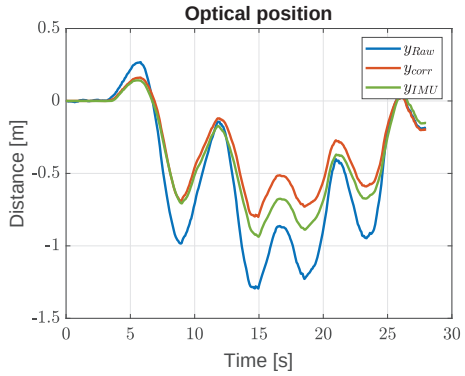
Appendix C

Possible solutions for the optical flow factor

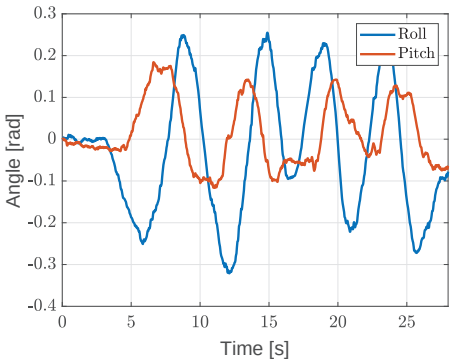
In section 5.5 we have shown that the proposed optical flow factor provides a significant improvement in the estimation accuracy. However, for certain trajectories with varying altitudes, there is a drift present in horizontal direction. Due to the varying altitude, possible causes are the influence of the attitude, the rotational velocity, the velocity in z direction and the altitude. Here we discuss the experiments done to verify if these could indeed be the causes to the drift occurring in optical flow.

Attitude and angular velocity of the quadrotor

A possible cause of the drift could be the influence of the attitude or the angular velocity of the drone. Therefore, an experiment is done where the drone is manually moved following a circular trajectory in x and z direction with two forward circles followed by two backward circles. During this movement the attitude of the drone is kept constant, i.e. $q \approx \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, \omega \approx 0$. Figure C.1 shows the results of this experiment. The estimated roll and pitch angles that are measured are a consequence of the sensor fusion between the gyroscope and the accelerometer. These angles are used in the IMU measurements. Therefore, Figure C.1a also shows the blue line representing the integrated raw optical flow data and the red line the integrated raw signal multiplied with the height according to (5.2). The results once again show that there is an error present, thus it can be stated that it is probably not caused by the attitude or an angular velocity. Note: during the experiments with a manual displacement the IMU estimates the height of the drone poorly with a lot of noise, probably caused by the accelerometer. Therefore, a number of experiments is done until a decent altitude estimation is achieved.



(a) Position plots in x-direction including the Optitrack measurements



(b) Estimated Roll and pitch angles, even though the drone is manually held at a constant angle

Figure C.1: Experimental results of the drone with a manually applied circular trajectory, twice in forward direction followed by two backward circles

Velocity in z-direction

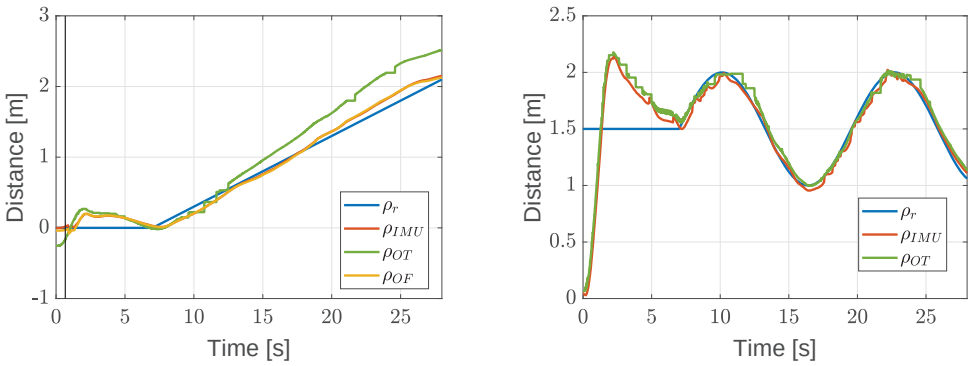
Because (5.2) has seemed to work properly for trajectories on different, constant altitudes, the error could be caused by a varying altitude that has an influence on the optical flow. Therefore, we can experiment on the influence of a velocity in z direction. Figure C.2 shows the results of a flight with trajectory $\rho_r = \begin{bmatrix} 0.1t & 0 & -1.5 - \sin(0.5t) \end{bmatrix}^T$. If the velocity in z direction would have an influence, we would expect a varying difference between the actual (Optitrack) position and the estimated position. There is a constantly increasing difference which indicates that the error is not caused by the velocity in z direction. Note that this constant error can be drawn to zero by adjusting the factor in (5.2). This indicates that there is a flaw in (5.2) rather than an influence of the attitude, angular velocity and the vertical velocity.

Revised optical flow actor

The section where the optical flow factor is calculated has shown small differences in the optimal factor, mostly depending on the altitude. The final value is a mean value of these factors and is used for the approximation of the horizontal velocity as

$$\dot{\rho}_{xy} = -\rho_z \dot{\rho}_{xy,s} Q, \tag{C.1}$$

where $\dot{\rho}_{xy,\mathrm{s}}$ is the velocity directly measured by the optical flow sensor and $Q = 1.06$ the determined optical flow factor. The small inconsistency in determined factors at different altitudes, which are used in calculating a mean value, could be the cause of the problem.



(a) Position plots in x-direction

(b) Position plots in z direction

Figure C.2: Experimental results of a trajectory with constant horizontal speed and a varying height.

Therefore, the optical flow factor could include the altitude itself, i.e., there is a different relation with the altitude. Unfortunately the experiments have only shown proper results for altitudes of 0.75, 1, 1.5 and 2.5 meters. For these altitudes we determine the best values to match the optical flow with the Optitrack measurements. Figure C.3 shows the data points of the renewed optimal optical flow factor in green. Because of the scarcity of data points there are multiple possible relations between the height and the optical flow factor. Therefore, both an exponential and a linear relation are fitted and tested in (5.2) as

$$Q_{\text{exp}} = 0.95 + 0.011 \exp(\dot{\rho}_{xy,\text{s}}) \tag{C.2a}$$

$$Q_{\text{lin}} = 0.92 + 0.07 \dot{\rho}_{xy,\text{s}}, \tag{C.2b}$$

which are displayed in Figure C.3 as the blue and red line respectively. These can now be tested in a varying height trajectory. Our starting point is the factor as in (C.1) with $Q = 1.06$. The results of a trajectory with $\rho_r = \begin{bmatrix}\cos(t) & 0 & -1.5 - \sin(t)\end{bmatrix}^T$ is shown in Figure C.4. It clearly shows the discussed drift with the true position, as measured with Optitrack. Thereafter, the same optical flow data is used with the renewed optical flow factors. The result for the exponential case is shown in Figure C.5a where the purple line represents the improved approximation. It shows a slight improvement. When the values of (C.2) are adjusted in this case it is possible to increase the resemblance with the Optitrack measurements as shown in Figure C.5b with a value of 0.02 for the coefficient instead of 0.011.

The same procedure is followed for the linear approximation. The results are shown in Figure C.6. The directional coefficient is changed to 0.1. Thus, for this trajectory there are two possible ways to get a good approximation. Now we test these propositions again for other trajectories. Figure C.7 shows the results for a 3D trajectory with the original optical flow factor of 1.06. It clearly shows the drift of the IMU/ optical flow with the true (Optitrack) measurements. Now this trajectory is used with the described optical flow factors. Figure C.8 shows the results for the linear optical flow factor with the directional coefficient of 0.1. The

resulting position tracking is slightly improved in x direction but in y direction however the position tracking shows a larger drift. Calculations with the exponential optical flow factor showed similar bad results. These results suggest that the optical flow error occurring at varying altitudes is not caused by the height factor included in the equation.

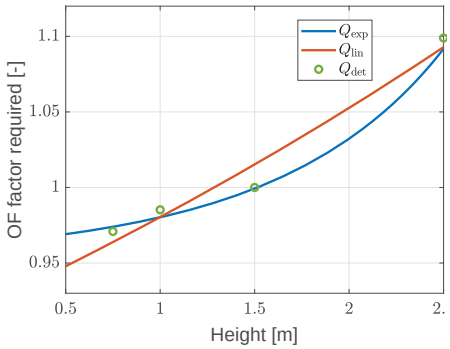
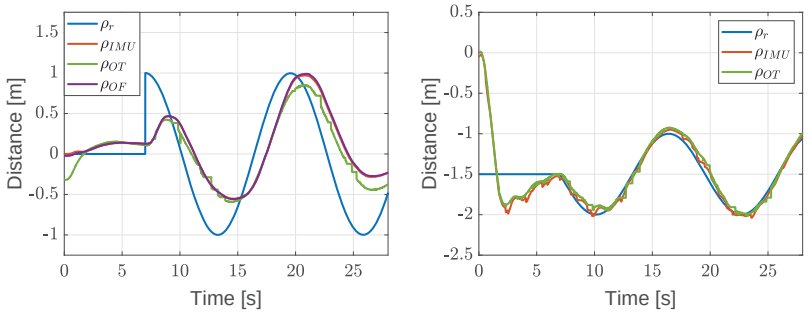


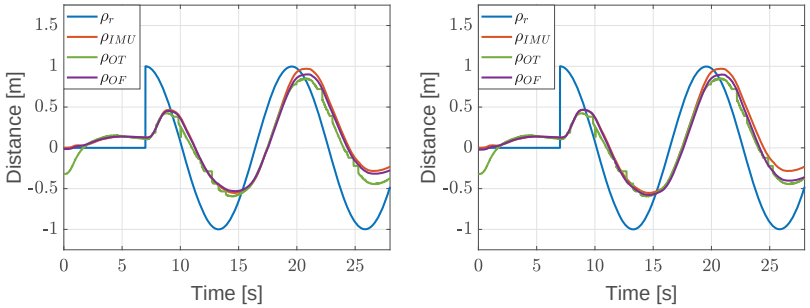
Figure C.3: The plots of (C.2) and the determined values found as described in section 5.4



(a) Position plots in x-direction with

(b) Position plots in z-direction

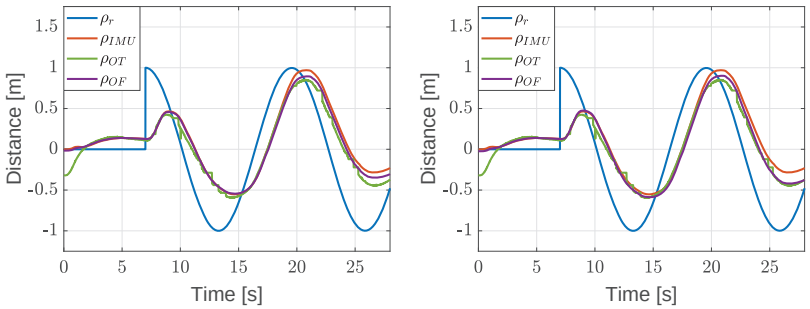
Figure C.4: Position plots for the circular trajectory in x and z direction for the original optical flow factor.



(a) Position plots in x-direction with (C.2a)

(b) Position plots in x-direction with the adjusted coefficient

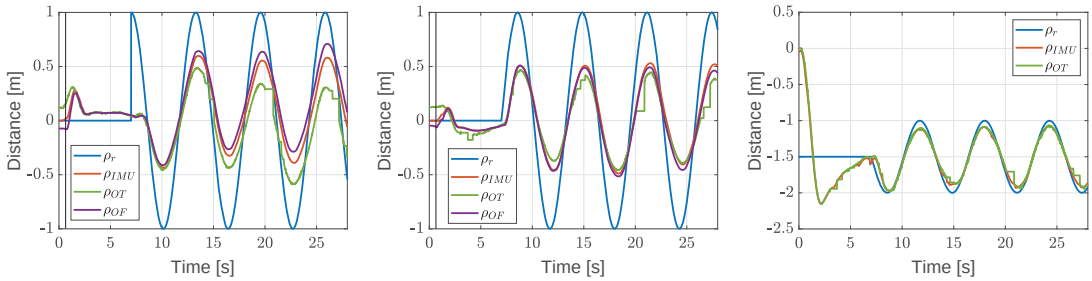
Figure C.5: Position plots for the circular trajectory in x direction for the exponential optical flow factors



(a) Position plots in x-direction with (C.2b)

(b) Position plots in x-direction with the adjusted coefficient

Figure C.6: Position plots for the circular trajectory in x direction for the linear optical flow factors

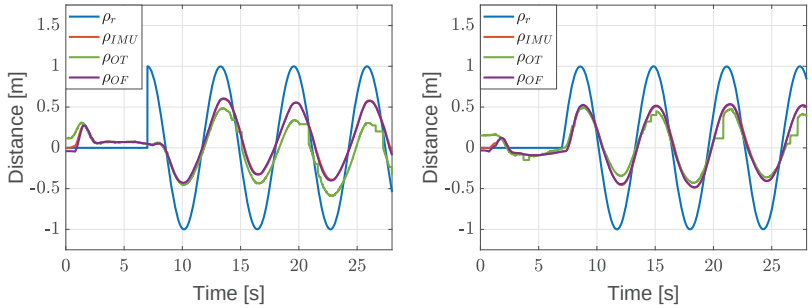


(a) Position plots in x-direction

(b) Position plots in y-direction

(c) Position plots in z-direction

Figure C.7: Position plots for the 3D trajectory using the original optical flow factor of 0.94.



(a) Position plots in x-direction (b) Position plots in y-direction

Figure C.8: Position plots for the 3D trajectory in x and y direction, the z direction is similar to Figure C.7c