Technische Universiteit
Eindhoven
University of Technology

Department of Mechanical Engineering
Control Systems Technology

System identification and control of a quadrotor

Internship

P. Rooijakkers

Supervisor:
dr. D.J. Guerreiro Tomé Antunes

version 1.1

Eindhoven, Sep 2016

Abstract

In this work, a linear controller is developed for the AR.Drone 2.0. The AR.Drone 2.0. is a multirotor aerial vehicle, with four rotors and therefore also denoted by quadrotor. A model-based approach is followed. The dynamic model is first obtained by means of grey box system identification. The model parameters are experimental identified. In particular, the moment of inertia is experimentally measured using a Bifilar Pendulum. The relation between thrust, torque and motor controller input is modeled based on experimental data, and accounts for parameter uncertainty. Then, a method for state estimation is described. The attitude of the quadrotor is estimated by fusing the sensor data from the gyroscope and from the acceleration meter. In addition, an alternative switching based sensor fusion method is proposed and evaluated. For the x-y position estimation of the quadrotor, several, camera based, methods are described. An Aruco marker based method and an optical flow algorithm expanded with an estimation of global motion, is implemented and reviewed. Moreover, an external camera method using Aruco markers is implemented. For the controller design, the non-linear equations of motion are linearized, and based on this model a linear controller is developed. A polytope-based method is proposed for checking robustness of the controller against rotor parameter variations. Finally, the linear controller is implemented on the AR.Drone 2.0 and the experimental obtained data is evaluated.

Contents

Chapter 1

Introduction

The development of multirotor areal vehicles knows a long history. However in recent years, the platform gained considerably more attention. Applications in film, agriculture, military, and transport industries becoming more and more visible.

Crucial in these applications is the flight controller. Several solutions for flight controllers are available in the literature [1], [7], [10], [9], [11], [12]. This work follows the research developed at the Control Systems Technology Group at the Technical University Eindhoven (TU/e). Among others, in previous work, a PID controller is investigated [14] [16], and a non-linear back stepping controller [17] is implemented on a AR.Drone 2.0. The AR.Drone 2.0 is a commercial available quadrotor produced by the company Parrot.

The objective of this work is to identify, in more detail, the system dynamics of the AR.Drone 2.0. and based on these identification data, design a flight controller which can be implemented in a trajectory tracking algorithm for autonomous flights.

In this work a dynamical model of the quadrotor is obtained by grey box modeling, in which a theoretical model is optimized with experimentally obtained parameters. The mass of the quadrotor is measured using a weighting scale and the moment of inertia is determined with a bifilar pendulum. The input-output relation between the motor input signal and the static thrust and torque are experimentally measured. A weighted least-squares method is used to fit a quadratic model to the experimentally obtained thrust and torque data, while perceiving an expression for uncertainty. Furthermore, a state-estimation method is developed. A drift free attitude estimation is obtained by fusing the high-frequent content from the gyroscope sensor and the low-frequent content from the acceleration sensor. In addition, a switching-based method for attitude estimation is proposed and evaluated. For the x-y position estimation, several, camera based, methods are introduced. The bottom camera of the quadrotor is used to estimate its position while hovering above a grid of Aruco markers. In a second approach, a fast optical flow method, based on local histogram matching, is extended with global motion and implemented. In a final approach, an external camera is used to track the quadrotor, using Aruco markers mounted on the quadrotor. Based on the dynamical model and the state estimation, a linear controller is designed that stabilizes the quadrotor. By introducing a pilot frame, a yaw rotation invariant controller is obtained. The stability of the controller, under model uncertainties, is checked using a polytope-based method. Finally, the controller is experimentally implemented on the AR.Drone 2.0. using the Simulink environment and the flight results are discussed.

The outline of this report is as follows. In Chapter 2 the non-linear model of a quadrotor is described, the system parameters are experimentally identified, and a rotor thrust-torque model is proposed based on experimental data. Chapter 3 discusses state-estimation including sensor fusion for attitude control, an alternative switching based sensor fusion algorithm, and camera based x-y position estimation. In Chapter 4, the non-linear model is linearized and a stabilizing controller is designed. Further, a polytope-based method for checking the robustness against parameter variations is evaluated and the experimental flight results are published. In final Chapter 5 a conclusion is given.

Chapter 2

Physical model

In this work, a model based approach for designing the controller is followed. Therefore, a mathematical description of the dynamics is needed. A so called grey-box model is obtained by refining a theoretical parameterized model in such a way that it closely approximates the quadrotor dynamics. The parameters are experimentally identified.
This chapter starts by introducing the quadrotor dynamical model in Section 2.1 and then addresses parameter identification in Section 2.2.

2.1 Equations of motions

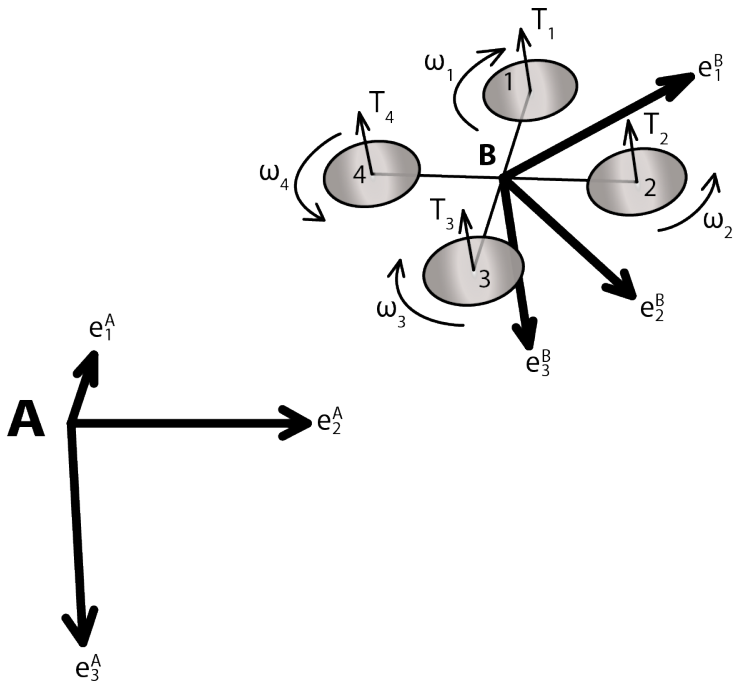

Figure 2.1: Schematic representation of the quadrotor. Indicated is the body-frame B and the inertia-frame A. T_i represents the thrust and ω_i the rotation direction of rotor i.

Dynamical model

The quadrotor in this work, is modeled as a rigid-body, equipped with four coplanar aligned rotors. Each rotor can be controlled independently and produces a thrust and a torque. In Figure 2.1 a quadrotor is sketched. The dynamical equations of the quadrotor can be derived from the Newton-Euler equations,

as described in [11]. The differential equations are

$$\dot{\xi}_A = \nu \tag{2.1}$$
$$m\dot{\nu}_A = -R(\lambda)\, T\, e_3^B + m\, g\, e_3^A \tag{2.2}$$
$$\dot{\lambda} = Q(\lambda)\, \omega \tag{2.3}$$
$$J\dot{\omega} = \tau + \omega \times J\omega \tag{2.4}$$

where $\xi_A = [x_A \quad y_A \quad z_A]^T$ and $\nu_A = [u_A \quad v_A \quad w_A]^T$ represents respectively the position and velocity of the quadrotor's center of mass, expressed in the inertia-frame (A). The rotation matrix $R(\lambda)$ converts the vector-orientations expressed in the body-frame (B) into inertia-frame (A) orientations, and is parameterized by the three Euler angles: roll (ϕ), pitch (θ), and yaw (ψ) stored in the vector $\lambda = [\phi \quad \theta \quad \psi]^T$.

$$R(\lambda) = R_z(\psi)R_y(\theta)R_x(\phi) \tag{2.5}$$
$$R(\lambda) = \begin{bmatrix} \cos\theta\cos\psi & \cos\psi\sin\theta\sin\phi - \cos\phi\sin\psi & \sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\theta\sin\phi\sin\psi & cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{2.6}$$

The time derivative of the Euler angels $\dot{\lambda}$ is related via matrix $Q(\lambda)$ to the angular body velocity's $\omega = [\omega_x \quad \omega_y \quad \omega_z]^T$. The derivation of matrix $Q(\lambda)$ is for example discussed in [4] and is given by

$$Q(\lambda) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}. \tag{2.7}$$

Furthermore, g represents the gravitational acceleration constant, m, the quadrotor's mass, and J the inertia matrix in the form

$$J = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.8}$$

where I_{ii} is the angular moment of inertia over the i axis. The gyroscopic effects $\omega \times J\omega$ are neglected since the angular body velocities are assumed to be low.
Finally, the system inputs are the total thrust T in the e_3^B direction, and the torques over the different axis represented by the vector $\tau_B = [\tau_x \quad \tau_y \quad \tau_z]^T$. The full written out equations of motion can be found in Appendix A.

In addition, the quadrotor can be mounted on a gimbal stand. This leads to a slightly different dynamical system. This system is described in detail in Appendix C.

Propulsion model

Four electric powered propellers drive the AR.Drone 2.0. Each propeller delivers thrust and torque. The static relations between motor input signal (pwm_i) and thrust (T_i) and torque (τ_i) is given by

$$T_i = c_i p\bar{w}m_i \tag{2.9}$$
$$\tau_i = (-1)^i d_i p\bar{w}m_i \tag{2.10}$$

where $p\bar{w}m_i = (pwm_i + k)^2$ and c_i, d_i and k are constants for motor i, as will be defined in detail in Section 2.2.2. Combining the trusts and torques from the four rotors gives the expression for the total applied force and torques on the quadrotor's body. This relation is described by

$$u = \Gamma\, p\bar{w}m \tag{2.11}$$
$$\begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 \\ d\,c_1 & -d\,c_2 & -d\,c_3 & d\,c_4 \\ d\,c_1 & d\,c_2 & -d\,c_3 & -d\,c_4 \\ -d_1 & d_2 & -d_3 & d_4 \end{bmatrix} \begin{bmatrix} p\bar{w}m_1 \\ p\bar{w}m_2 \\ p\bar{w}m_3 \\ p\bar{w}m_4 \end{bmatrix} \tag{2.12}$$

where d is the distance from the motor center to the body frame axis, (not the diagonal distance), $u = [T \quad \tau_x \quad \tau_y \quad \tau_z]$ and $p\bar{w}m = [p\bar{w}m_1 \quad p\bar{w}m_2 \quad p\bar{w}m_3 \quad p\bar{w}m_4]$. One can concluded that the matrix Γ couples the independent motor contributions. The inverse is also true; Γ^{-1} decouples the control signals T and τ_i in pwm_j as will be of importance in the controller design in Chapter 4.
Another effect, besides the static input-output relation, is that the internal motor controllers require time to change the propellers speed. To model this effect, a first order dynamical model with time constant τ_c can be introduced in the form

$$p\dot{\bar{w}}m_i = \frac{1}{\tau_c}\left(p\hat{w}m_i - p\bar{w}m_i\right) \tag{2.13}$$

where $p\hat{w}m_i = (pwm_i + k)^2$ is the new (intermediate) input. The time constant is assumed to be small and therefore this model is not taken into account.

2.2 Experimental parameter indentification

2.2.1 Mass, Center of Gravity, and Moment of Inertia

Quadrotor mass

The mass of the AR.Drone 2.0 is experimentally determined using an adequate weighing scale. The total and component mass is presented in Table 2.1.

Complete mass (including protective indoor hull)	0.456kg
Base	0.292kg
Battery	0.104kg
Protective indoor hull	0.060kg

Table 2.1: Total and component mass of the AR.Drone 2.0 (TU/e Parrot Quad number 3).

Center of gravity

The location of the center of gravity (CoG) is obtained by balancing the quad on a cord. The cord is repositioned until the quadrotor hangs horizontal or vertical. It was determined that the CoG in $x-y$ plane is located at the point where the motor support arms cross each other. The CoG in z-direction is measured at $48mm$ above the ground supports, which is approximately equal to the lens center height of the front camera.

Experimentally obtained Moment of inertia

The moment of inertia is experimentally determined by using the Bifilar pendulum principle, as described in Appendix D and as is shown in Figure D.1 and D.2. By determining several easy to measure constants, such as the distance between the cords ($2r$), the cord length (L), the gravitational acceleration constant (g) and the body mass (m), the moment of inertia (I) can be determined by measuring the oscillation frequency ($f = 2\pi\omega$). The relation between the measures is given by

$$I_{ii} = \frac{mgr^2}{4\pi^2 L f^2}. \tag{2.14}$$

A complete derivation of this equation is given in the Appendix D. The experimental results for the AR.Drone 2.0 with indoor hull are shown in Table 2.2. The theoretical approximations used in [7] and [10] are added for comparison reasons. It can be concluded that the theoretical approximations are close to our experimental measured values. In addition, the results for the AR.Drone 2.0 without indoor hull are given in Appendix D.

	Theoretical [7]		Theoretical [10]		Experimental	
	$m\,[kg]$	$I\,[kg \cdot m^2]$	$m\,[kg]$	$I\,[kg \cdot m^2]$	$m\,[kg]$	$I\,[kg \cdot m^2]$
I_{xx}	0.429	0.0022	0.436	0.0045	0.456	0.0033
I_{yy}	0.429	0.0029	0.436	0.0051	0.456	0.0037
I_{zz}	0.429	0.0048	0.436	0.0095	0.456	0.0068

Table 2.2: Experimental measured moment of inertia using a Bifilar pendulum (AR.Drone 2.0 with indoor hull). For validation purpose the theoretical approximations in [7] and [10] are also included.

2.2.2 Propulsion model

In this work, the static thrusts and torques are measured experimentally. The direct relation from input (*pwm* signal) to output (thrust/torque) is measured. In other works, the thrust and torque is often measured as a function of the angular velocity of the rotor [7]. In our setup we omit the intermediate step of measuring the angular velocity because it is not direct measurable and it appears to be redundant for the final quadrotor control.

Experimental setup

From eight motors of two different quadrotor's (AR.Drone 2.0) the static trust and torque is measured, using a weighing scale as measuring instrument. In steps of 10% in the range $10\%, 20\% \ldots 100\%$ the trust and torque is measured. The experimental thrust measurement setup is shown in Figure 2.2 and in Figure 2.3 the setup for measuring the torque is shown. For the torque measurement the quad is placed on a stand that allows the quatrotor only to rotate around the z-axis. The torque is transformed by a lever in a force working on the weighing scale.

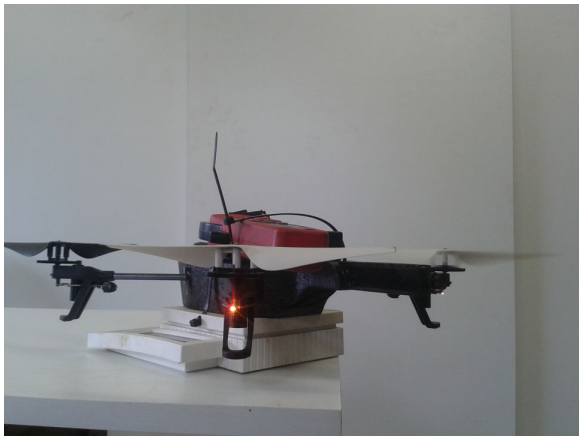

Figure 2.2: Experimental setup measuring static thrust using a weighingscale. With the active rotor placed over the edge of the desk to avoid the influence of ground effects [9].

Figure 2.3: Experimental setup for measuring the torque using a radial grove bearing stand, lever and weighing scale.

Experimental results

The experimentally obtained results are shown in Figure 2.4. As can be observed, the deviation between the motors is relative large. It is even not guaranteed that when a motor delivers more thrust, relative to another motor, it also delivers more torque. As becomes clear by analyzing the differences between motor 1 and 2 of quad 4 (q4) for example. Another observation is that motor 2 (q3) delivers significant less thrust, the cause of this reduction is a slightly bended propeller shaft, probably caused by crashes. The deviation between motors emphasizes the urge to a certain robustness against motor characteristics variations in the final controller design.
Additional experimental measurements can be found in Appendix F.

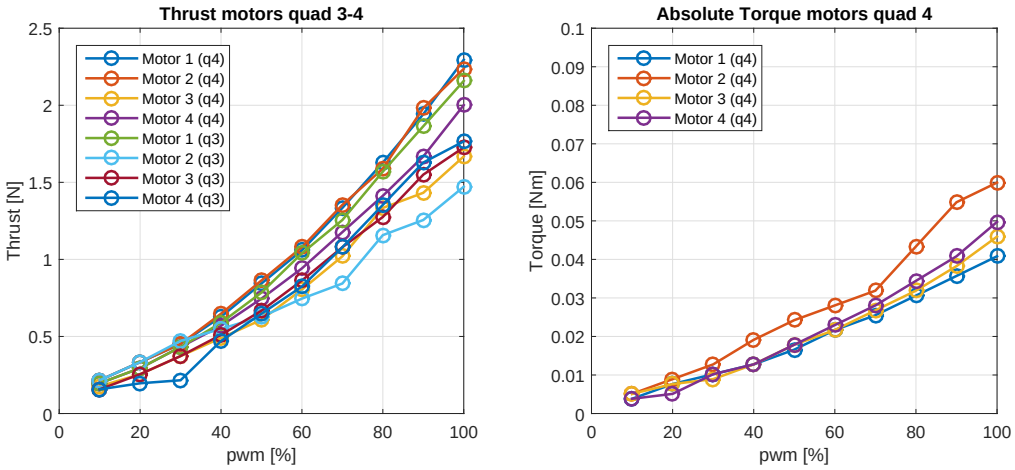

Figure 2.4: Measurement results, thrust and torque. Absolute torque because motor 1 and 3 deliver in practice negative torque due to the different rotation directions.

Thrust, torque, input model

The following mathematical model is proposed to describe the experimental data in Figure 2.4

$$T_{mi} = c_{mi}\left(pwm_i + k\right)^2 \tag{2.15}$$

$$\tau_{mi} = d_{mi}\left(pwm_i + k\right)^2 \tag{2.16}$$

Where T_{mi} and τ_{mi} are the trust and torque produced by motor i respectively, c_{mi} and d_{mi} are the thrust and torque coefficients respectively, and k is a shared shift. Each motor characteristic is thus described by only two parameters (c_{mi} and d_{mi}), and all motors share a common shift (k). The shift is introduced because the motor characteristics are not close to zero when the pwm value is close to zero. A weighted least square method is used to fit the model to the experimental data, as is described in Appendix E. The data fitting procedure yields the mean values with corresponding standard deviation as presented in Table 2.3 and Figure 2.5.

	$c_{mi}\left[\frac{N}{\%^2}\right]$	$d_{mi}\left[\frac{Nm}{\%^2}\right]$	$k\,[\%]$
Mean	$0.1318 \cdot 10^{-3}$	$0.8722 \cdot 10^{-5}$	25.19
Standard deviation	$1.349 \cdot 10^{-5}$	$1.039 \cdot 10^{-6}$	0

Table 2.3: Estimated motor characteristics accourding to the model given by (2.15) and (2.16).

It must be taken into account that the model (2.15) and (2.16) only holds for the range $0\% < pwm_i \leq 100\%$. Thus,

$$T_{mi} = \begin{cases} 0 & if\, pwm_i \leq 0. \\ c_{mi}\left(pwm_i + k\right)^2 & otherwise. \end{cases} \tag{2.17}$$

$$\tau_{mi} = \begin{cases} 0 & if\, pwm_i \leq 0. \\ d_{mi}\left(pwm_i + k\right)^2 & otherwise. \end{cases} \tag{2.18}$$

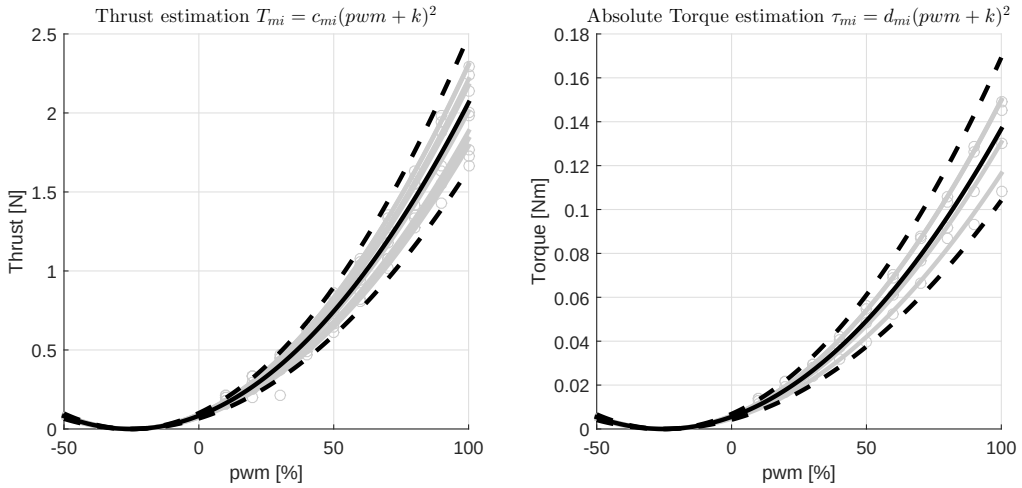

Figure 2.5: Estimation of motor characteristics each individual experimental motor measurement is shown in grey. In solid black is shown the characteristic with the mean value. The dashed black lines represent the thrust and torque characteristic when deviating twice the standard deviation from the mean value.

Chapter 3

State Estimation

As it is typical the case in model-based control design, the controller is composed of a state estimator and a state-base compensation. This chapter describes the state estimation. Section 3.1 discuses the attitude estimation and Section 3.2 discuses position estimation.

3.1 Attitude Estimation

Attitude control is one of the crucial components of stabilizing a quadrotor. The attitude of the quadrotor is determined by its roll (ϕ), pitch (θ), and yaw (ψ), but only the control of ϕ and θ are relevant for x, y plane stabilization. Therefore, the control of ϕ and θ is referred to as attitude control. It is essential for proper control to have a valid estimation of the current quadrotor's attitude. The AR.Drone 2.0 has two sensors available for measuring its attitude:

1. An acceleration sensor, measuring the acceleration ($a = [a_x \quad a_y \quad a_z]^T$) in three directions.
2. A gyroscope sensor, measuring the angular velocity ($\omega = [\omega_x \quad \omega_y \quad \omega_z]^T$) over three axises.

Several methods for attitude estimation are discussed next.

3.1.1 Attitude estimation from accelerometer.

Using only the acceleration sensor is in principle sufficient for measuring the roll (ϕ) and pitch (θ) angles in case of steady state body motions, i.e. no body accelerations. In this case only the gravity vector ($g\, e_z^3$) is measured. The roll (ϕ) and pitch (θ) angles can then be calculated using the inverse map of the rotation matrix (2.6) (only the bottom row elements of the rotation matrix are required here). The equations are

$$\phi_a = \tan^{-1}\left(\frac{a_y}{a_z}\right), \tag{3.1}$$

$$\theta_a = -\sin^{-1}\left(\frac{a_x}{\|a\|}\right). \tag{3.2}$$

These equations can be written in vector form, as $\lambda_a = [\phi_a \quad \theta_a]^T$. However, using the above equations in the presence of body accelerations result in an incorrect estimation of the attitude, and even more important in instability. An intuitive example is given in Figure 3.1. This figure sketches the cause of instability under body accelerations. In both, sub figures A and B, the same angles are estimated from the accelerometer output. In case A, in the absence of body accelerations, the estimated angle is correct. A logic control action is then to apply a clockwise torque. As opposed to A, in B the attitude is estimated incorrectly, due to body accelerations. Applying the same control action yields an even larger acceleration and deviation from the horizontal plan, resulting in unstable behavior.

3.1.2 Attitude estimation from gyroscope.

Another approach for calculating the roll (ϕ) and pitch angle (θ) is by integrating the angular velocities measured by the gyroscope, according to the non-linear equation of motion (2.3)

$$\dot{\lambda}_g = Q(\lambda)\,\omega \tag{3.3}$$

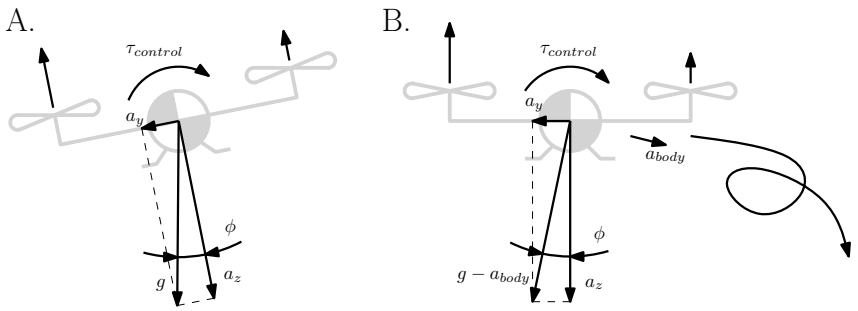

Figure 3.1: The same angle is estimated from the acceleration sensor in both cases, A and B. The incorrect estimation in case B, due to body accelerations, results in instabillity.

where we select only the roll (ϕ) and pitch (θ) part. We denote the estimation of ϕ, θ according to (3.3) as $\lambda_g = [\phi_g \quad \theta_g]^T$. The problem with the angular velocity signal is an inevitable bias. This bias causes drift when integrated. Therefore using only the gyroscope for attitude control is also insufficient.

3.1.3 Sensor fusion.

In the following approach, the high frequent content of the gyroscope is merged with the low frequent content of the acceleration sensor. The gyroscope signal is integrated and a correction term is added to compensate for drift, in the form

$$\dot{\hat{\lambda}} = \dot{\lambda}_g + c\left(\lambda_a - \hat{\lambda}\right) \tag{3.4}$$

where $\hat{\lambda} = [\hat{\phi} \quad \hat{\theta}]^T$ is the estimated attitude and c is in its simplest case a positive real scalar constant. The equivalent schematic overview of (3.4) is given in Figure 3.2. Laplace transform of (3.4) yields

$$\hat{\Lambda}(s) = \frac{s}{s+c}\Lambda_g(s) + \frac{1}{\frac{1}{c}s+1}\Lambda_a(s) \tag{3.5}$$

which is equivalent to a low-pass filter for the acceleration measurements and a high-pass filter for the gyroscope measurements, and is known as a complementary filter, as is shown in Figure (3.3). In other words, the gyroscope is used to estimate the fast dynamics (during for example body accelerations), while the acceleration sensor is used to estimate the slow dynamics. When c is a simple positive gain then the drift is bounded to a steady-state error. In principle c can be chosen more complex to remove this steady-state error, however, experimental results have shown that using only a gain is sufficient since the complete controlled system uses velocity feedback to remove this bias.

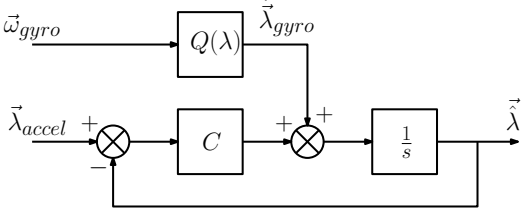

Figure 3.2: Schematic representation of sensor fusion.

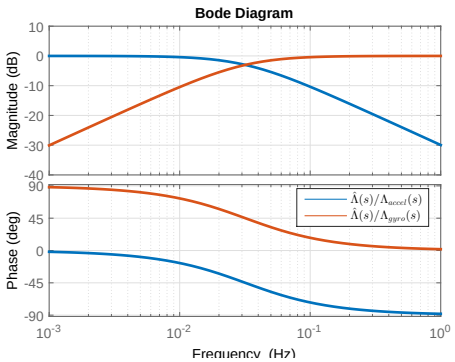

Figure 3.3: Input output relation in the form of a bode plot. In blue is shown the amplification ratio $\hat{\Lambda}(s)/\Lambda_{accel}(s)$, and in red is shown the amplification ratio $\hat{\Lambda}(s)/\Lambda_{gyro}(s)$

3.1.4 Experimental results

Shown in Figure 3.4 is the experimental result of estimating the roll (ϕ) angle under the influence of random body accelerations. It can be observed that the acceleration sensor signal (ϕ_{accel}) and the integrated gyroscope (ϕ_{gyro}) signal differ significantly, mainly due to the body accelerations. Also can be observed that the signal ϕ_{gyro} is contaminated with drift. Nevertheless, the final estimated signal $\hat{\phi}$ is free from drift and follows the high frequent (accurate) signal from ϕ_{gyro}.

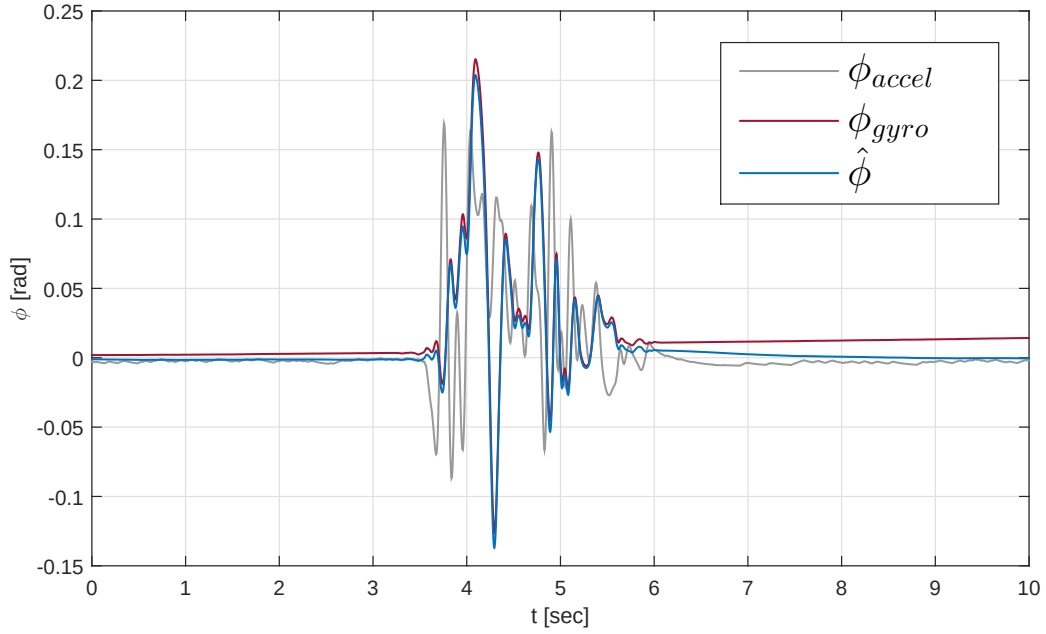

Figure 3.4: Experimental estimation of the roll (ϕ) angle under influence of body accelerations. One can observe that the estimated value tracks the highfrequent content from the gyroscope, and the low-frequent content from the acceleration sensor.

3.1.5 Alternative proposed switching based method for sensor fusion.

In this subsection we propose and discuss a different attitude estimation method. This estimator directly uses the integrated angular velocity from the gyroscope. The acceleration sensor is only used to correct for drift, when measuring the pure gravity vector (thus in case of no body accelerations). The proposed rules, for verifying that only the gravity vector is measured are

$$a_z > 0 \tag{3.6}$$

$$|\|a\| - 9.81| < \delta_n \tag{3.7}$$

$$\left| \frac{d}{dt} \|a\| \right| < \delta_d \tag{3.8}$$

where δ_n and δ_d are small bounds. When all the above equations hold then we make the estimated angle $\hat{\lambda}$ equal to the current estimate from the acceleration sensor λ_a, otherwise the integrated signal of the gyroscope λ_g is used to update the estimated angle $\hat{\lambda}$. In other words, the integrator that integrates the gyroscope signal is (re)set to λ_a when (3.6), (3.7), and (3.8) holds. Thus

$$\hat{\lambda} = \lambda_a \tag{3.9}$$

$$\lambda_g = \lambda_a \tag{3.10}$$

otherwise

$$\hat{\lambda} = \lambda_g. \tag{3.11}$$

where λ_g is given by (3.3).
The experimental results are shown in Figure 3.5. One can observe that indeed the estimation follows λ_g without drift. Furthermore, from experimental flights it can be concluded that the method works quite well. However, besides an observable shocking behavior due to the switching mechanism, it incidentally occurs (according to Murphy's law) that the wrong angle is estimated, this causes a direct crash of the quadrotor. It can therefore be concluded that implementing this method for attitude estimation, in this form, is too naive.

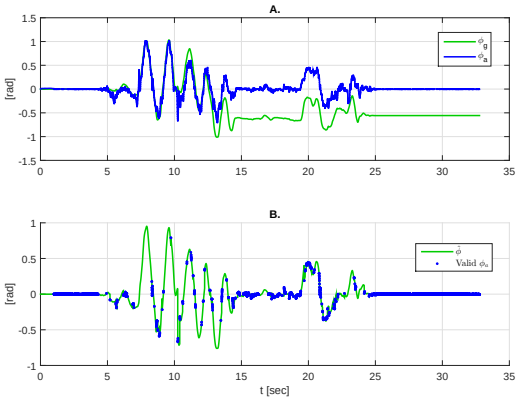

Figure 3.5: Experimental results of our proposed method. In A is shown the estimated ϕ angle direct from the sensors, the drift can be observed in the gyroscope signal. In B the estimation of the gyroscope is corrected when the acceleration sensor measures only the gravity vector.

3.2 Position estimation

3.2.1 Height estimation

The AR.Drone 2.0 has an ultrasound and an air pressure sensor available for estimating its height above the ground. In this work, only the ultrasound sensor is used. The height can be estimated in a piecewise linear sense from the sensor output

$$u_{sens} = \begin{cases} 0 & z < c_z \\ \frac{z - b_z}{a_z} & c_z \leq z < b_z \\ 0 & b_z \leq z \end{cases} \tag{3.12}$$

where z is the depth, u_{sens} is the sensor output, $b_z = -0.2[m]$ is the smallest measurable depth, $c_z = -2.75[m]$ is the largest measurable depth, and $a_z = -0.0076[m/-]$ is a linear scaling factor (notice the sign convention, height is negative depth, since the north-east-down (NED) convention is used). The height is in principle undetermined, with no further information, for the case $u_{sens} = 0$. In this work low fly heights are assumed. Therefore, the height can be estimated according to

$$z = \begin{cases} b & u_{sens} = 0 \\ a_h \, u_{sens} + b_h & b \leq u_{sens} \end{cases} \tag{3.13}$$

3.2.2 X-Y positon estimation

Bottom camera

The AR.Drone 2.0 is according to the documentation equipped with a HD front camera and a $60FPS$ 320×240 pixel bottom camera. As described in [1], the bottom camera is in the original software used for the estimation of the X-Y position and U-V velocity. The methods that where used are a feature detection method combined with an optical flow method. In this work, a feature detection method and a lightweight optical flow method are experimentally implemented in the Simulink environment library

[6]. The advantage of using an on-board camera, instead of an external camera, is that the quadrotor can fly autonomously (independent from its environment).

Aruco marker grid

In this first approach, of determining the quadrotor position, our Matlab Aruco Marker detection implementation is used, to determine the quadcopter's position, while hovering over a known grid of Aruco markers. The Aruco marker implementation is described in Appendix G. An overview of the experimental setup is shown in Figure 3.6. The detection algorithm locates the separate markers. Once the markers are detected, the two dimensional marker corner locations (image-points) are known, and the corresponding three dimensional (real-world points) locations can be looked up in the on-board map. Then, a GaussNewton iteration optimization method is used to find the parameterized projective map, describing the bottom camera location in the scene. This method is described in detail in [15] 6.1.3. The parameters in our case are selected such that they correspond directly to the states $[x \quad y \quad z \quad \phi \quad \theta \quad \psi]$.

Figure 3.6: Experimental setup, position estimation from marker grid. The method is first tested using a pc, Matlab, and a regular webcam.

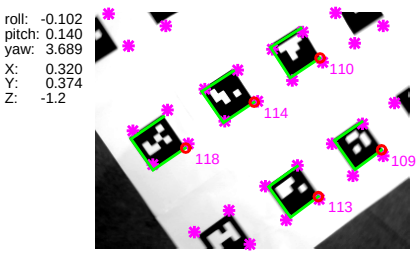

Figure 3.7: Detected Markers and estimated quadrotor position from bottom camera image.

This method is experimentally implemented in the AR.Drone 2.0 using the Simulink interface. It can be concluded that the method works as expected, as is shown in Figure 3.7. A draw back of the Simulink implementation is that the achieved frame rate is around the 2.2 $[sec]$ per frame, against the $\approx 20ms$ per frame in the compiled Matlab pc implementation (using a mex compiler). Because of this low frame rate, the method in this implementation is not further used for real-time control.

Optical Flow

In [8] the authors propose a lightweight method for estimating the optical flow. In this work, the method is implemented with a few minor modifications (the Sobel filter is omitted). In this method, the rows and columns are block-wise (40×40 pixels) summed such that two (an x and an y) vector are obtained. This is done for two consecutive frames from the camera. In a next step, the corresponding vectors are correlated by convolution, resulting in a shift representing the best correspondence. The shift, represents directly the x-y components of the flow-vector. All the flow vectors from all the blocks together represent an estimation of the complete flow field.

To go from a (estimated) flow-field to global motion, the center of instantaneous rotation is calculated as is described in Appendix H. From the center of instantaneous rotation the translation and rotation between the two consecutive images is determined. Since the frame rate ($f = 1/Ts$), the height of the quadrotor, and the intrinsic parameters of the camera are known, the quadrotor's U-V ground velocity can be estimated.

In Figure 3.8, the experimental results for a 1.25 square trajectory over a 6×6 meter square, over a grass field, is shown. In A two consecutive images are shown, left the image at time $t - Ts$ and on the right at time t. In both images is shown the same optical flow field with the center of instantaneous rotation indicated with a marker (in this case the center of instantaneous rotation is located outside the image, therefore only the marker for the left image is visible). In C is shown the estimated velocity, and in D the integrated version is shown, representing the position. Finally, in B, the estimated trajectory is

shown. It can be observed that there is drift.

The implementation of this method suffers from the same problem as was already pointed out in the Aruco marker grid method in Section 3.2.2, the computation power of the AR.Drone 2.0 using the Simulink interface is not sufficient, with only an frame rate of 1.3[*sec*]. This method is therefore not further implemented.

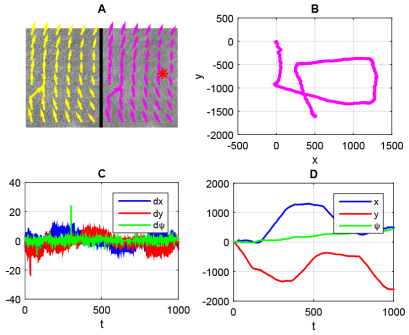

Figure 3.8: Experimental results of estimating U-V velocity and X-Y position from optical flow.

External top camera

Another method for measuring the quadcopter's X-Y position and yaw angle (ψ) is using an external camera mounted on the ceiling of the building. The camera is connected to a pc, the pc calculates the quadrotor's X, Y, and yaw position and sends it in real time to the quadrotor, as is proposed in [17]. With this solution, the autonomously and environmental independency is scarified. Which, in this research case, raises no restrictions.
In this work we use an Aruco marker on top of the quadrotor, as is shown in Figure 3.9, to make the quadrotor identifiable, instead of the LED coding used in [17]. The Aruco marker is used, because the used quadrotor is not yet equipped with LED's. Besides the missing LED's there is no reason why the LED's method wouldn't work (the same UDP communication channel between quadrotor and pc is used). Shown in Figure 3.10 is an example of the detected quadrotor in an image from the top camera.
This method measures only the absolute position. For estimating the velocity, a Luenberg observer of the form

$$\begin{bmatrix}\dot{\hat{x}}\\ \dot{\hat{u}}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\hat{x}\\ \hat{u}\end{bmatrix} + \begin{bmatrix}L_{11}\\ L_{21}\end{bmatrix}(x_{cam} - \hat{x}) \qquad \begin{bmatrix}\dot{\hat{y}}\\ \dot{\hat{v}}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\hat{y}\\ \hat{v}\end{bmatrix} + \begin{bmatrix}L_{12}\\ L_{22}\end{bmatrix}(y_{cam} - \hat{y}) \tag{3.14}$$

is used. Where $\hat{x}$, $\hat{u}$, $\hat{y}$, $\hat{v}$ are the estimated states, x_{cam}, y_{cam} are the measured positions received from the top camera, and L are gains determined by pole placement.

Figure 3.9: Marker with a rib-size of 10 cm is mounted with head pins on the quadrotor.

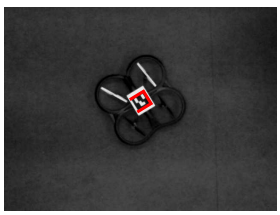

Figure 3.10: Example of detected quadcopter with marker in an image from the top camera.

Chapter 4

Controller design

In the previous chapters we introduced a dynamical model and a state estimation method. In this chapter, a linear controller will be designed with the objective to stabilize and steer the quadrotor to a programmed location. Although that there are several non-linear control approaches such as dynamic feedback linearization [12], backstepping [17], and sliding mode control, a linear controller is justified since we assume near hovering flight conditions.

4.1 Linearization

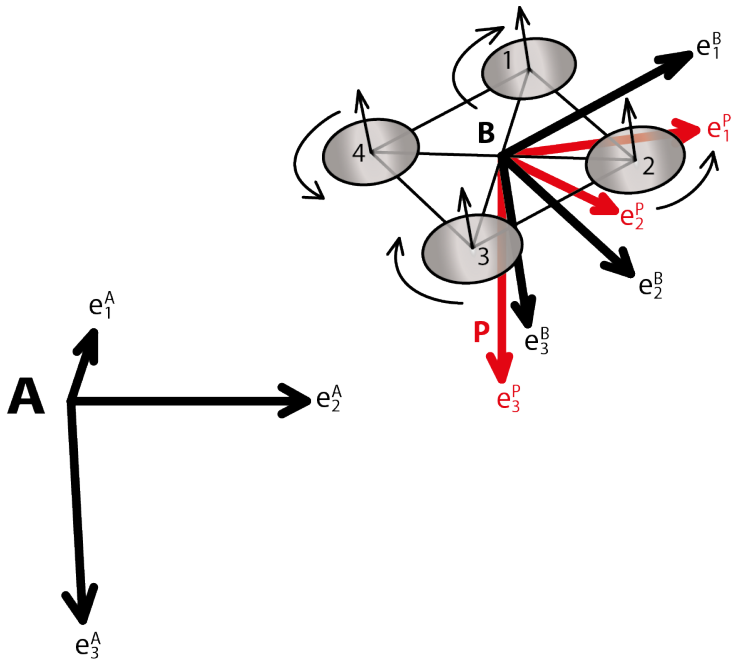

Figure 4.1: Schematic representation of the quadrotor. Indicated is the body-frame B, pilot-frame P and the inertia-frame A.

The non-linear equations of motion described in Section 2.1 can be linearized around the equilibrium position $\phi = \theta = \psi = \omega_x = \omega_y = \omega_z = 0$. However, the yaw angle ψ (heading of the quatrotor) can not assumed to be approximately zero. To avoid this problem a *Pilot frame* is introduced, as is sketched in Figure 4.1. The orientation of the pilot frame is equal to the orientation of the inertia frame except for the yaw angle (ψ), i.e. in the pilot frame the x-axis (e_1^P) is aligned with the heading of the quadrotor. Further, the origin of the pilot frame is located at the same location as the origin of the body frame (quadrotor center of gravity). The transformation from inertia-frame coordinates to pilot-frame coordinates is given

by the following homogeneous transformation matrix

$$T_{PA} = \begin{bmatrix} R_\psi^T & -R_\psi^T \xi_A \\ 0 & 1 \end{bmatrix} \tag{4.1}$$

where

$$R_{psi}(\psi) = \begin{bmatrix} cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.2}$$

With the system equations expressed in the pilot frame, the linear equations become

$$\begin{aligned} \dot{x}_P &= u_P & \dot{\phi} &= \omega_x & \dot{u}_P &= -\frac{T_0}{m}\theta & \dot{\omega}_{xx} &= \frac{1}{I_x}\tau_x \\ \dot{y}_P &= v_P & \dot{\theta} &= \omega_y & \dot{v}_P &= \frac{T_0}{m}\phi & \dot{\omega}_{yy} &= \frac{1}{I_y}\tau_y \\ \dot{z}_P &= w_P & \dot{\psi} &= \omega_z & \dot{w}_P &= -\frac{1}{m}T & \dot{\omega}_{zz} &= \frac{1}{I_z}\tau_z \end{aligned}$$

where $T_0 = m\,g$ is the thrust required to compensate for gravity, yielding $\dot{u}_P = -g\,\theta$ and $\dot{v}_P = g\,\phi$. When using the thrust-torque to motor input relation as, described by (2.12), the linear system can be expressed in the common form

$$\dot{x_P} = Ax_P + B\bar{u} \tag{4.3}$$

$$y_P = Cx_P + D\bar{u} \tag{4.4}$$

where, the full written out matrices are given in Appendix B.

4.2 Linear control

In Chapter 3 a full state estimation is developed. In Section 4.1 a linear expression is given of the system dynamics. It is therefore possible to design a linear controller using full state feedback. Where all non-linear effects are handled as disturbances. In this work, negative gain feedback of the form

$$u = -Kx \tag{4.5}$$

is used to stabilize the quadrotor, where $u = [T \quad \tau_x \quad \tau_y \quad \tau_z]^T$ is the system input, K a matrix with gains, and x is the system state. The final pwm motor signals are then calculated by inverting the $p\bar{w}m$ to u relation (2.12) such that

$$pwm = \sqrt{\Gamma^{-1}u} - k. \tag{4.6}$$

The LQR method could be used for finding the optimal gains in matrix K. LQR requires a weighting function. However, the exact weights are unknown and can be used as tuning knobs. By analyzing the linearized system, it can be observed that the linear dynamics are decoupled, and therefore, from a practical point of view, tuning the gains directly in the matrix K, is of equivalent difficult as tuning the weights in the LQR approach. Therefore the regular method of loopshaping is used for finding the gains in matrix K.
In the next sections is shown that applying simple gain feedback on the full state indeed stabilizes the system.

4.2.1 Attitude control

The control objective of attitude control is to stabilize the quadrotor around zero roll (ϕ) and pitch (θ) angle. As can be observed from the linear model, the roll (ϕ), pitch (θ), and yaw (ψ) angles can be

manipulated by the torques τ_x, τ_y, and τ_z respectively. The relation between the torques and angles is a simple double integrator

$$\Lambda(s) = \frac{1}{J\,s^2} U(s) \tag{4.7}$$

The system, with two poles at zero, is critically stable. In the bode plot in Figure 4.2, the system is indicated with the letter P (Plant). The system can be stabilized by applying a PD controller or a lead filter. In this work a lead filter is used of the form

$$U(s) = k \frac{\tau_1 s + 1}{\tau_2 s + 1} \Lambda(s) \tag{4.8}$$

since $\Omega(s) = s\Lambda(s)$ the equation above can be rewritten as

$$U(s) = \frac{1}{\tau_2 s + 1} (k_1 \Omega(s) + k_2 \Lambda(s)) \tag{4.9}$$

from which it becomes clear that proportional state feedback can be equivalent to a PD controller with low-pass filter. The low-pass filter is already embedded as part of the state estimation. The controller is indicated with the letter C (Controller) in Figure 4.2. Also is shown the open-loop (CP) and stable closed-loop (CP/(1+CP)).

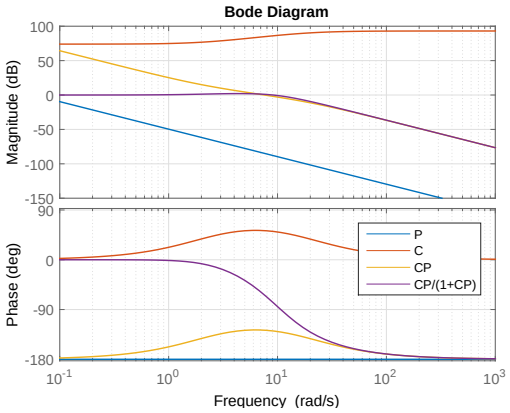

Figure 4.2: Bode diagram of attitude control (roll). Plant (P) is a double integrator. Controller (C) is a lead filter.

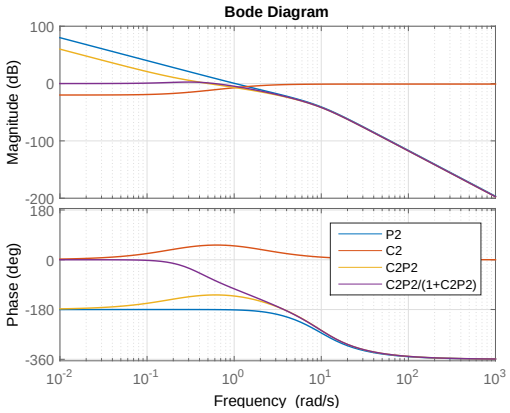

Figure 4.3: Bode diagram of position control (x). Plant (P2) is the double integrated attitude controlled closed-loop . Controller (C) is a lead filter.

4.2.2 Position control

From the linear equations it can be observed that by integrating the roll (ϕ) and the pitch (θ) angles twice the $x - y$ position is obtained. This means that the twice integrated attitude controlled system, described in Section 4.2.1, is also critically stable, as is shown in Figure 4.3. Again a PD or lead filter can stabilize the system, and again the controller can be rewritten in the form

$$T_x(s) = \frac{1}{\tau_4 s + 1} (k_3 U(s) + k_4 X(s)) \tag{4.10}$$

$$T_y(s) = \frac{1}{\tau_4 s + 1} (k_5 V(s) + k_6 Y(s)) \tag{4.11}$$

The controller and related open-loop and closed-loop are shown in Figure 4.3. One can observe that the bandwidth of the position controlled system is limited by the bandwidth of the attitude controlled system.

The height of the quadrotor can be manipulated by the thrust (T) input. The input output relation can again be modeled by a second order system, therefore a lead filter will stabilize the system. During the linearization process in Section 4.1 a gravitation compensation term ($T_0 = m\,g$) was introduced. This term is added as a static value to the thrust input, such that the height controller only handles the body accelerations and disturbances in e_3^A direction.

4.2.3 Integral action

Proportional and differential control does not correct for steady state errors. Steady state errors occur due to for example a bias in the roll, pitch angles or an incorrect gravity compensation. Integral control is able to remove steady state errors [3]. Integral action is only applied to the final control objectives. The control objectives in this work are position (ξ) and heading (yaw (ψ)) control. Integral control can be added by introducing extra states containing the integrated error. The additional control contribution for the integral control is then the integrated error times a constant gain K_I. A schematic representation of the control implementation is given in Figure 4.4

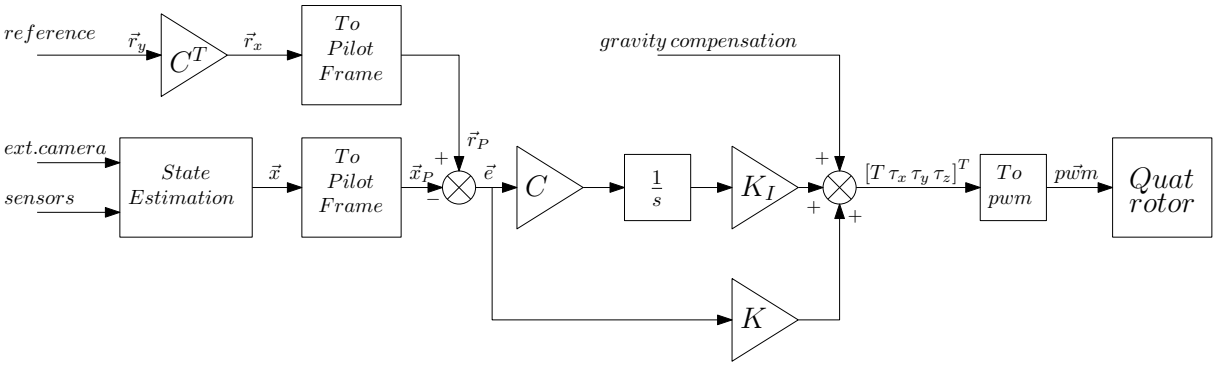

Figure 4.4: Schematic representation of the implemented controller.

4.3 Robustness against parameter variations

Once the gains are chosen, one might want to determine whether the system is still stable under parameter variations. In Section 2.2.2 was indicated that there is a variation in thrust and torque constants between the different motors. These variations can be described by

$$c_{mi} = \bar{c} + r\delta_c \tag{4.12}$$

$$d_{mi} = \bar{d} + s\delta_d \tag{4.13}$$

where $\bar{c}$ and $\bar{d}$ represent the mean thrust and torque constants and δ_c, δ_d the corresponding standard deviation, r and s represents a ratio of the present standard deviation in the real thrust-torque constant c_{mi} and d_{mi}. By reformulating the B matrix by including the coupling matrix (2.12) in the form

$$B(\delta) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{m} & 0 & 0 & 0 \\ 0 & \frac{d}{I_{xx}} & 0 & 0 \\ 0 & 0 & \frac{d}{I_{yy}} & 0 \\ 0 & 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \left(\begin{bmatrix} \bar{c} & \bar{c} & \bar{c} & \bar{c} \\ \bar{c} & -\bar{c} & -\bar{c} & \bar{c} \\ \bar{c} & \bar{c} & -\bar{c} & -\bar{c} \\ -\bar{d} & \bar{d} & -\bar{d} & \bar{d} \end{bmatrix} + \begin{bmatrix} r_1\delta_c & r_2\delta_c & r_3\delta_c & r_4\delta_c \\ r_1\delta_c & -r_2\delta_c & -r_3\delta_c & r_4\delta_c \\ r_1\delta_c & r_2\delta_c & -r_3\delta_c & -r_4\delta_c \\ -s_1\delta_d & s_2\delta_d & -s_3\delta_d & s_4\delta_d \end{bmatrix} \right), \tag{4.14}$$

it can be observed that only the $B(\delta)$ matrix is affected with this parameter uncertainty. The system input is now $p\bar{w}m = [p\bar{w}m_1 \quad p\bar{w}m_2 \quad p\bar{w}m_3 \quad p\bar{w}m_4]$. The $B(\delta)$ matrix can therefore be rewritten in the form

$$B(\delta) = \bar{B} + B_1(r_1) + B_2(r_2) + B_3(r_3) + B_4(r_4) + B_5(s_1) + B_6(s_2) + B_7(s_3) + B_8(s_4) \tag{4.15}$$

Then the stability is guaranteed according to [13], 5.1 when all polytopes are stable. When taking $r_{1...4}$, $s_{1...4}$ to be the eight uncertain parameters varying between $-n < r_i < n$ and $-n < s_i < n$ where n is a constant representing n-times the standard deviation, then we have to check $2^8 = 256$ polytopes. This stability check is done by verifying that for all polytopes hold that

$$\lambda_{\max}|A - B(\delta)K| < 0 \tag{4.16}$$

for a continuous time system case.

4.4 Experimental results

In this work, the linear controller using the state estimation from Chapter 3 is implemented in the AR.Drone 2.0, utilizing the Simulink library provided by [6]. It turned out that the most efficient method for tuning the control gains, after a first theoretical obtained initial value, is by changing the gains online. Starting with attitude control and extending them with position control. It can be concluded that, by applying the above linear control law, the system can be stabilized with large recovery capability against disturbances. In Figure 4.5 is shown the quadrotor undergoing a rough push from a broom. The step response to several side-steps in y direction is shown in Figure 4.6.

Despite that with the linear controller good control performance can be achieved, some non-linear effects can be observed. For example, during the sidesteps, as shown in Figure 4.6, the quadrotor initially looses height due to the lost of thrust caused by the roll of the quadrotor. This relation is not explained by the linear model, it is a non-linear effect. Another effect that indicates non-linearity is that for large side-steps the initial roll is too large, causing the quadrotor to crash. This initial roll is caused by the large error amplified with a proportional gain. A working solution is to bound the control torques produced by large set-point errors. Another, more preferable, solution is to use trajectory tracking, as for example is investigated in [5].

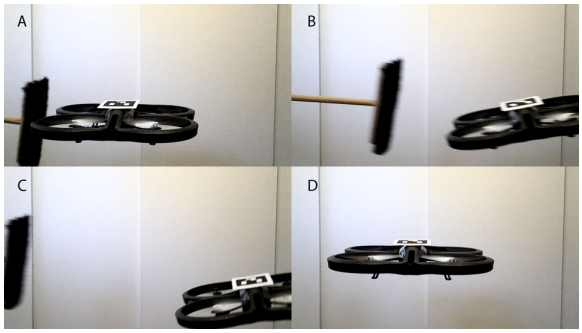

Figure 4.5: Experimental result, testing quadrotor robustness against side-push.

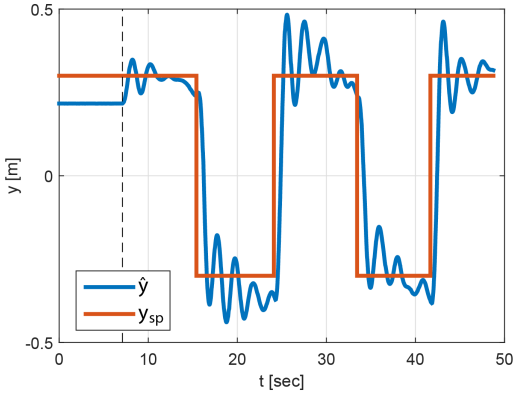

Figure 4.6: Response to side step. Takeoff at $t = 7.1\ sec$

Chapter 5

Conclusions

In this work, the dynamic parameters of the AR.Drone 2.0 are experimentally identified. The experimentally obtained moment of inertia agrees well with the theoretical approximations in the previous works [7] [10]. The gyroscope and the acceleration measurements are fused together as part of the state-estimation. The sensor fusion method combines the high frequent data from the gyroscope with the low frequent data from the acceleration sensor. An experimental implementation showed that sensor fusion increases the stability of the quadrotor due to the absence of drift. For estimating the x-y position of the quadrotor, a Aruco marker method using a grid of Aruco markers on the ground, and an optical flow method expanded with global motion was implemented. Both methods work successfully, however, since the implementation using the Simulink library is computationally too heavy, the methods are not further used. Instead, an external camera mounted on the ceiling is used to calculate the quadrotor position, by detecting a Aruco marker on top of the quadcopter. Further, a full state linear controller is designed and the robustness under parameter uncertainties is checked using a polytope method. The controller is successfully implemented in the AR.Drone 2.0 using the Simulink environment. Finally, it is shown by analyzing the step responses that the controller succesfully stabilizes the quadrotor.

Bibliography

[1] Pierre-Jean Bristeau, Franois Callou, David Vissire, and Nicolas Petit. The navigation and control technology inside the ar.drone micro uav. *IFAC World Congress Milano*, 2011. 1, 12

[2] S. Garrido-Jurado, R. Mu noz Salinas, F.J Madrid-Cuevas, and M.J. Marn-Jimnez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Department of Computing and Numerical Analysis. University of Cordoba*, -. 36

[3] Abbas Emami-Naeini Gene F. Franklin, J. David Powell. *Feedback Control of Dynamic Systems Sixth Edition*. Pearson, 2010. 18

[4] T. J. Bridges H. Alemi Ardakani. Review of the 3-2-1 euler angles: a yawpitchroll sequence. *Department of Mathematics, University of Surrey, Guildford GU2 7XH UK*, April 2010. 4

[5] Dave Kooijman. Trajectory planning and control of quadrotors. *Technische Universiteit Eindhoven*, 2015. 19

[6] Daren Lee. Simulink ar drone target (software). *MathWorks*, 2014. 13, 19

[7] Qianying Li. Grey-box system identification of a quadrotor unmanned aerial vehicle. Master's thesis, Delft Center for Systems and Control, August 2014. 1, 5, 6, 21

[8] Kimberly McGuirey, Guido de Croony, Christophe de Wagtery, Bart Remesy, Karl Tuylszyand, and Hilbert Kappen. Local histogram matching for efficient optical flow computation applied to velocity estimation on pocket drones. -, 2016. 13, 38

[9] Robert Mahony Moses Bangura. Nonlinear dynamic modeling for high performance control of a quadrotor. *Australasian Conference on Robotics and Automation*, 2012. 1, 6

[10] J. Pestana, J. L. Sanchez-Lopez, I. Mellado-Bataller, Changhong Fu, and P. Campoy. Ar drone identification and navigation control at cvg-upm. *Centre for Automation and Robotics, joint research centre CSIC-UPM*, -. 1, 5, 6, 21

[11] Peter Corke Robert Mahony, Vijay Kumar. Multirotor aerial vehicles, modeling, estimation and control of a quadrotor. *IEEE Robotics & Automation Magazine*, September 2012. 1, 4

[12] Francesco Sabatino. Quadrotor control: modeling, nonlinear control design, and simulation. Master's thesis, KTH Electrical Engineering, 2015. 1, 15

[13] Carsten Scherer and Siep Weiland. *Linear Matrix Inequalities in Control*. Department of Electrical Engineering, Eindhoven University of Technology; Delft Center for Systems and Control, Delft University of Technology, 2004. 19

[14] Somanna Thapanda Suresh. Control of a quadcopter using pid. Master's thesis, Technische Universiteit Eindhoven, 2015. 1

[15] Richard Szeliski. *Computer Vision: Algorithms and Applications*. http://szeliski.org/Book/, 2010. 13, 37

[16] J.J.E. Unkel. Modeling and pid control of a quadrotor. Master's thesis, Technische Universiteit Eindhoven, 2015. 1

[17] B.C.M. van Aert. Control and coordination algorithms for autonomous multi-agent quadrotor systems. Master's thesis, Technische Universiteit Eindhoven, 2016. 1, 14, 15

Appendix A

Non-Linear equations of motion

The differential equations as introduced in Section 2.1 are

$$\dot{\xi}_A = \nu \tag{A.1}$$
$$m\dot{\nu}_A = -R(\lambda)\, T\, e_3^B + m\, g\, e_3^A \tag{A.2}$$
$$\dot{\lambda} = Q(\lambda)\, \omega \tag{A.3}$$
$$J\dot{\omega} = \tau + \omega \times J\omega. \tag{A.4}$$

The above equations full written out yields

$$\dot{x}_A = u_A \tag{A.5}$$
$$\dot{y}_A = v_A \tag{A.6}$$
$$\dot{z}_A = w_A \tag{A.7}$$
$$\dot{\phi}_A = \omega_x + \omega_y \sin\phi \tan\theta + \omega_z \cos\phi \tan\theta \tag{A.8}$$
$$\dot{\theta}_A = \omega_y \cos\phi - \omega_z \sin\phi \tag{A.9}$$
$$\dot{\psi}_A = \omega_y \sin\phi \sec\theta + \omega_z \cos\phi \sec\theta \tag{A.10}$$
$$\dot{u}_A = -\frac{1}{m}\left(\sin\phi \sin\psi + \cos\phi \cos\psi \sin\theta\right) T \tag{A.11}$$
$$\dot{v}_A = -\frac{1}{m}\left(\cos\phi \sin\theta \sin\psi - \cos\psi \sin\phi\right) T \tag{A.12}$$
$$\dot{w}_A = -\frac{1}{m}\left(\cos\theta \cos\phi\right) T + g \tag{A.13}$$
$$\dot{\omega}_{xx} = \frac{1}{I_x}\tau_x \tag{A.14}$$
$$\dot{\omega}_{yy} = \frac{1}{I_y}\tau_y \tag{A.15}$$
$$\dot{\omega}_{zz} = \frac{1}{I_z}\tau_z. \tag{A.16}$$

Appendix B

Linear system equations, expressed in pilot frame.

The full written out matrices as defined in Section 4.1 are

$$x = \begin{bmatrix} x_P & y_P & z_P & \phi & \theta & \psi & u_P & v_P & w_P & \omega_x & \omega_y & \omega_z \end{bmatrix}^T$$

$$u = \begin{bmatrix} p\bar{w}m_1 & p\bar{w}m_2 & p\bar{w}m_3 & p\bar{w}m_4 \end{bmatrix}^T$$

$$y = \begin{bmatrix} x_P & y_P & z_P & \psi \end{bmatrix}^T$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{c_1}{m} & -\frac{c_2}{m} & -\frac{c_3}{m} & -\frac{c_4}{m} \\ d\frac{c_1}{I_{xx}} & -d\frac{c_2}{I_{xx}} & -d\frac{c_3}{I_{xx}} & d\frac{c_4}{I_{xx}} \\ d\frac{c_1}{I_{yy}} & d\frac{c_2}{I_{yy}} & -d\frac{c_3}{I_{yy}} & -d\frac{c_4}{I_{yy}} \\ -\frac{d_1}{I_{zz}} & \frac{d_2}{I_{zz}} & -\frac{d_3}{I_{zz}} & \frac{d_4}{I_{zz}} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Appendix C

Physical model of quadrotor in stand

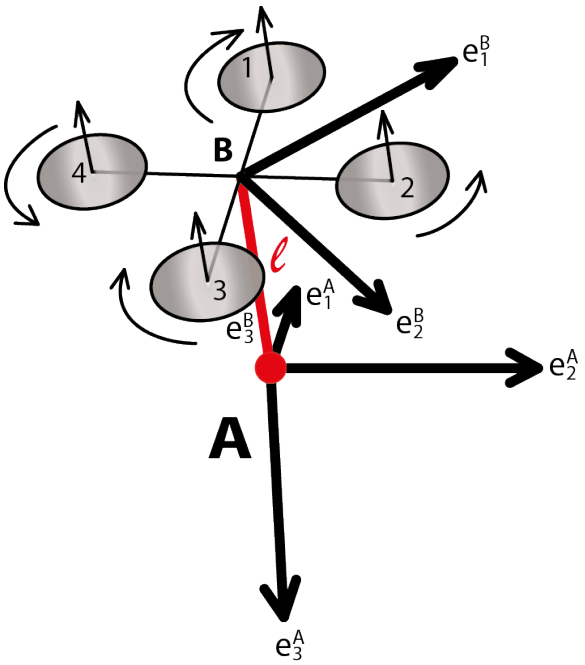

Figure C.1: Schematic overview quadrotor mounted on a stand.

Figure C.2: Detail overview of the stand, On the right, the quadrotor is mounted to a gimbal. On the left the quadrotor can be mounted to a radial grove bearing, the yaw angle is then the only degree of freedom.

When the quadrotor is mounted on a stand, as is shown in Figure C.1 and C.2, the corresponding equations of motion are

$$\dot{\lambda} = Q(\lambda)\,\omega_B \tag{C.1}$$

$$J\dot{\omega_B} = \tau_B - lR(\lambda)\; e_3^B \times mge_3^A + \omega_B \times J\omega_B \tag{C.2}$$

where l is the distance from quadrotor center of gravity to stand joint.
The full written out non-linear equations are

$$\dot{\phi}_A = \omega_x + \omega_y \sin\phi \tan\theta + \omega_z \cos\phi \tan\theta \tag{C.3}$$

$$\dot{\theta}_A = \omega_y \cos\phi - \omega_z \sin\phi \tag{C.4}$$

$$\dot{\psi}_A = \omega_y \sin\phi \sec\theta + \omega_z \cos\phi \sec\theta \tag{C.5}$$

$$\dot{\omega}_{xx} = \frac{1}{I_x + ml^2}\left(\tau_x + mgl\left(\cos\psi \sin\phi - \cos\phi \sin\theta \sin\psi\right)\right) \tag{C.6}$$

$$\dot{\omega}_{yy} = \frac{1}{I_y + ml^2}\left(\tau_y + mgl\left(\sin\psi \sin\phi - \cos\phi \sin\theta \cos\psi\right)\right) \tag{C.7}$$

$$\dot{\omega}_{zz} = \frac{1}{I_z}\tau_z \tag{C.8}$$

linearization yields a system of the form

$$\dot{x_B} = Ax + B\bar{u} \tag{C.9}$$

where

$$x = \begin{bmatrix} \phi & \theta & \psi & \omega_{x_B} & \omega_{y_B} & \omega_{z_B} \end{bmatrix}^T \tag{C.10}$$

$$u = \begin{bmatrix} T & \tau_x & \tau_y & \tau_z \end{bmatrix}^T \tag{C.11}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{glm}{I_x+ml^2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{glm}{I_y+ml^2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_x+ml^2} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y+ml^2} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix} \tag{C.12}$$

Appendix D

Bifilar pendulum and Moment of Inertia

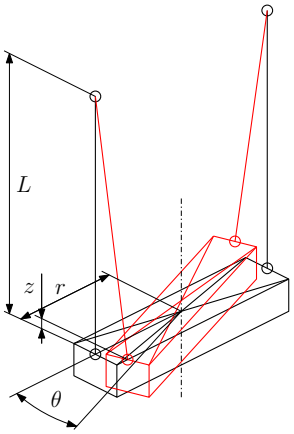

Figure D.1: Schematic overview Bifilar pendulum.

Figure D.2: Experimental setup for measuring the moment of inertia (I_{xx}).

The moment of inertia can be experimentally determined by use of a Bifilar pendulum. In this appendix, a derivation of (2.14) is given. The Bifilar pendulum is sketched in Figure D.1. The vertical displacement (z) due to body rotation (θ) is given by

$$L^2 = (r\theta)^2 + (L - z)^2 \tag{D.1}$$

$$z = L - \sqrt{L^2 - r^2\theta^2} \tag{D.2}$$

where L is the cord length, r the horizontal distance from the body center of gravity to the cord, i.e. half the distance between the cords, θ is the angle of rotation along the vertical axis, and z is the vertical displacement of the body due to rotation θ. Assumed is that after the release (where $\theta \neq 0$ and $\dot{\theta} = 0$) of the body the total energy in the system remains constant such that

$$E_{total} = E_{kinetic} + E_{potential} = c \tag{D.3}$$

$$= \frac{1}{2} I_z z \dot{\theta}^2 + mgz \tag{D.4}$$

The derivative to time is then zero

$$\frac{d}{dt}\left(E_{kinetic} + E_{potential}\right) = 0 \tag{D.5}$$

$$\dot{E}_{kinetic} = \dot{E}_{potential} \tag{D.6}$$

$$I_{zz}\dot{\theta}\ddot{\theta} = \frac{mgr^2\theta\dot{\theta}}{\sqrt{L^2 - r^2\theta^2}} \tag{D.7}$$

when θ is assumed small the equations can be linearized, the equations of motion become then

$$I_{zz}\ddot{\theta} = \frac{mgr^2}{L}\theta. \tag{D.8}$$

Since the equation is linear and undamped we can substitute a solution of the form,

$$\theta = a\cos\omega t + b\sin\omega t \tag{D.9}$$

$$\ddot{\theta} = \omega^2\left(a\cos\omega t + b\sin\omega t\right). \tag{D.10}$$

This yields the equation

$$I_{zz}\omega^2\left(a\cos\omega t + b\sin\omega t\right) = \frac{mgr^2}{L}\left(a\cos\omega t + b\sin\omega t\right) \tag{D.11}$$

$$I_{zz} = \frac{mgr^2}{L\omega^2} \tag{D.12}$$

$$I_{zz} = \frac{mgr^2}{4\pi^2 Lf^2} \tag{D.13}$$

where $f = 2\pi\omega$ is the oscillation frequency.

Experimental results
In Figure D.2 is shown the experimental setup for measuring the I_{xx} moment of inertia. The selected cord material for this experiment is fish-wire, which is mounted on a horizontal bar on the ceiling. The experiments are several times repeated with varying cord configurations, this is done to get a feeling for how reliable this method is. As the results show, this method can be adopted as reliable for relative large body masses compared to the low cord mass. In this case, the method gives good results for the quad as whole. The method is in this setup not suitable for determining the moment of inertia of a single propeller, these propeller measurements are therefore not presented in the following list with experimental obtained moment of inertias.
Nomenclature

m	Mass of object ($m = 0.456kg$).
g	Gravitational acceleration constant $\left(g = 9.81\frac{N}{sec^2}\right)$.
r	Radial distance from center of gravity to cord connection.
L	Length of cord.
t	Measure time.
n	Number of oscillations during measure time.
f	Calculated oscillation frequency.
I	Calculated moment of inertia.

AR.Drone 2.0 with Protective indoor hull
Experimental results for the moment of inertia I_{zz}

$m\,[kg]$	$r\,[m]$	$L\,[m]$	$t\,[sec]$	$n\,[-]$	$f\,[Hz]$	$I\,[kg\cdot m^2]$
0.456	0.1240	1.4250	120.0200	50	0.4166	0.0071
0.456	0.1240	1.4250	120.2500	50	0.4158	0.0071
0.456	0.0700	1.4250	81.9000	20	0.2442	0.0066
0.456	0.0700	1.4250	102.7800	25	0.2432	0.0066
0.456	0.1170	0.9180	19.4100	10	0.5152	0.0064
0.456	0.1170	0.9180	57.4900	30	0.5218	0.0062
0.456	0.1115	2.2670	133.5700	40	0.2995	0.0070
0.456	0.1115	2.2670	166.8600	50	0.2997	0.0070
0.456	0.1115	2.2670	50.1500	15	0.2991	0.0070
					Mean:	0.0068

Experimental results for the moment of inertia I_{xx}

$m\,[kg]$	$r\,[m]$	$L\,[m]$	$t\,[sec]$	$n\,[-]$	$f\,[Hz]$	$I\,[kg \cdot m^2]$
0.456	0.1260	1.8310	91.7600	50	0.5449	0.0033
0.456	0.1260	1.8310	55.4200	30	0.5413	0.0034
0.456	0.1255	0.9350	65.2000	50	0.7669	0.0033
					Mean:	0.0033

Experimental results for the moment of inertia I_{yy}

$m\,[kg]$	$r\,[m]$	$L\,[m]$	$t\,[sec]$	$n\,[-]$	$f\,[Hz]$	$I\,[kg \cdot m^2]$
0.456	0.1230	1.8340	100.6500	50	0.4968	0.0038
0.456	0.1230	0.9380	70.2200	50	0.7120	0.0036
					Mean:	0.0037

AR.Drone 2.0 No hull

	$m\,[kg]$	$r\,[m]$	$L\,[m]$	$t\,[sec]$	$n\,[-]$	$f\,[Hz]$	$I\,[kg \cdot m^2]$
I_{xx}	0.398	0.1270	1.825	48.25	30	0.6218	0.0023
I_{yy}	0.398	0.1250	1.825	52.07	30	0.5761	0.0026
I_{zz}	0.398	0.1725	1.815	50.06	30	0.5993	0.0045

AR.Drone 2.0 Protective indoor hull + markers (as is shown in Figure G.2)

	$m\,[kg]$	$r\,[m]$	$L\,[m]$	$t\,[sec]$	$n\,[-]$	$f\,[Hz]$	$I\,[kg \cdot m^2]$
I_{xx}	0.502	0.1370	1.750	62.16	30	0.4826	0.0057
I_{yy}	0.502	0.1250	1.750	68.38	30	0.4387	0.0058
I_{zz}	0.502	0.1725	1.815	71.67	30	0.4186	0.0117

Appendix E

Least squares fit of propulsion model

Assumed is that the model of static trust and torque can be described in the form

$$T_{mi} = c_{mi}\left(pwm + k\right)^2 \tag{E.1}$$

$$\tau_{mi} = d_{mi}\left(pwm + k\right)^2 \tag{E.2}$$

where subscript mi represent the specific motor, and k a shared common signal shift for all motors the same. In this form, each motor thrust characteristic can be described with one parameter c_{mi}, and for the torque characteristic by parameter d_{mi}. These parameters can be estimated from the experimentally obtained data in a least squares sense. Rewriting the equations (E.1) (E.2) gives

$$pwm = a_{mi}\sqrt{T_{mi}} - k \tag{E.3}$$

$$pwm = b_{mi}\sqrt{\tau_{mi}} - k \tag{E.4}$$

$$\tag{E.5}$$

where $a_{mi} = \sqrt{\frac{1}{c_{mi}}}$ and $b_{mi} = \sqrt{\frac{1}{d_{mi}}}$. These equations are linear in the parameters and can therefore be solved in a weighted least squares manner.

$$Ax = b \tag{E.6}$$

$$WAx = Wb \tag{E.7}$$

$$x = \left(A^T W A\right)^{-1} A^T W b \tag{E.8}$$

where

$$A = \begin{bmatrix} \sqrt{\vec{T}_{m1}} & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & -1 \\ 0 & \sqrt{\vec{T}_{m2}} & \ddots & 0 & 0 & \cdots & & 0 & -1 \\ \vdots & \ddots & \ddots & 0 & 0 & & \cdots & 0 & -1 \\ 0 & 0 & 0 & \sqrt{\vec{T}_{mi}} & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & \sqrt{\vec{\tau}_{m1}} & 0 & 0 & 0 & -1 \\ \vdots & \ddots & \ddots & 0 & 0 & \sqrt{\vec{\tau}_{m2}} & \cdots & 0 & -1 \\ \vdots & \ddots & \ddots & 0 & 0 & & \ddots & 0 & -1 \\ \vdots & \ddots & \ddots & 0 & 0 & & \ddots & \sqrt{\vec{\tau}_{mj}} & -1 \end{bmatrix} \tag{E.9}$$

$$x = \begin{bmatrix} a_{m1} & a_{m2} & \cdots & a_{mi} & b_{m1} & b_{m2} & \cdots & b_{mj} & k \end{bmatrix}^T \tag{E.10}$$

$$b = \begin{bmatrix} p\vec{w}m_{m1}^T & p\vec{w}m_{m2}^T & \cdots & p\vec{w}m_{mi}^T & p\vec{w}m_{m1}^T & p\vec{w}m_{m2}^T & \cdots & p\vec{w}m_{mj}^T \end{bmatrix}^T \tag{E.11}$$

$$W = diag\left(\begin{bmatrix} w & \cdots & w \end{bmatrix}\right) \tag{E.12}$$

$$w = diag\left(\begin{bmatrix} 1 & 1 & 1 & 5 & 5 & 50 & 100 & 100 & 50 & 1 \end{bmatrix}\right) \tag{E.13}$$

where the weight is large on the values involved in the hovering position.
The final estimated parameters are shown in Table E.1.

Quadrotor	Motor	$c_i \left[\frac{N}{\%^2}\right]$	$d_i \left[\frac{Nm}{\%^2}\right]$	$k\,[\%]$
3	1	$0.1387 \cdot 10^{-3}$		25.19
3	2	$0.1412 \cdot 10^{-3}$		25.19
3	3	$0.1172 \cdot 10^{-3}$		25.19
3	4	$0.1202 \cdot 10^{-3}$		25.19
4	1	$0.1469 \cdot 10^{-3}$	$0.9550 \cdot 10^{-5}$	25.19
4	2	$0.1473 \cdot 10^{-3}$	$0.9574 \cdot 10^{-5}$	25.19
4	3	$0.1143 \cdot 10^{-3}$	$0.7427 \cdot 10^{-5}$	25.19
4	4	$0.1283 \cdot 10^{-3}$	$0.8337 \cdot 10^{-5}$	25.19
	mean	$0.1318 \cdot 10^{-3}$	$0.8722 \cdot 10^{-5}$	25.19
	standard deviation	$1.349 \cdot 10^{-5}$	$1.039 \cdot 10^{-6}$	0

Table E.1: Thrust and torque parameters, estimated from the measurement data.

Appendix F

Additional thrust measurements

Shown in this appendix are the experimental thrust measurement which were not used in this work

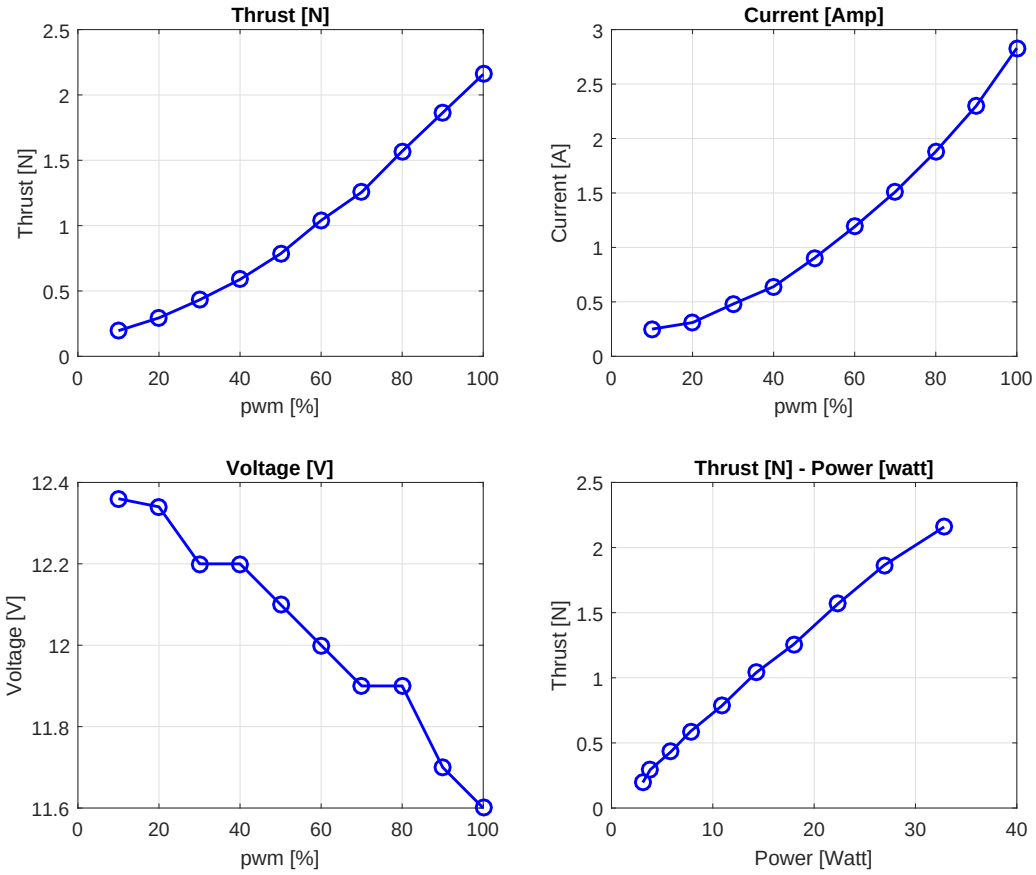

Figure F.1: Experimentally obtained thrust of motor 1 of quadrotor 3, where the power is $P = UI$ is calculated from the current and the voltage.

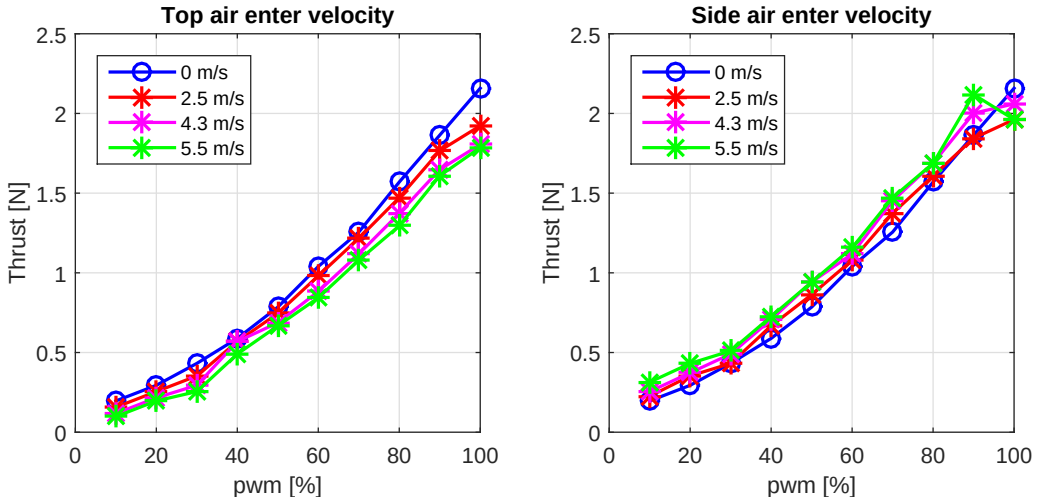

Figure F.2: Experimentally obtained influence of top and side wind on motor 1 of quadrotor 3. The side and top wind is produced by a ventilator for which the velocity is rougly estimated.

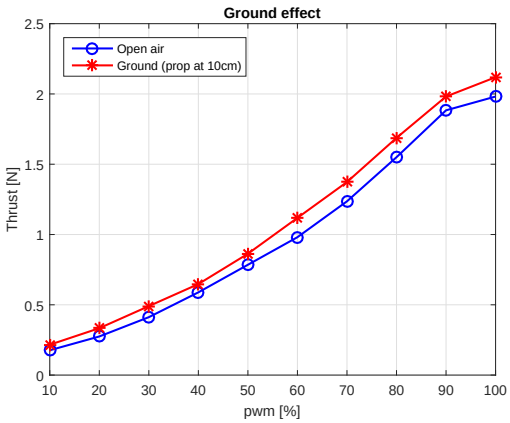

Figure F.3: Experimentally obtained ground effect of motor 1 of quadrotor 3.

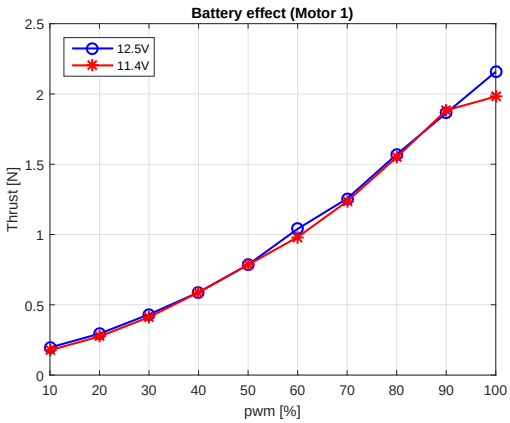

Figure F.4: Experimentally obtained effect of battery discharge on motor 1 of quadrotor 3.

Appendix G

Aruco marker detection and pose estimation from camera.

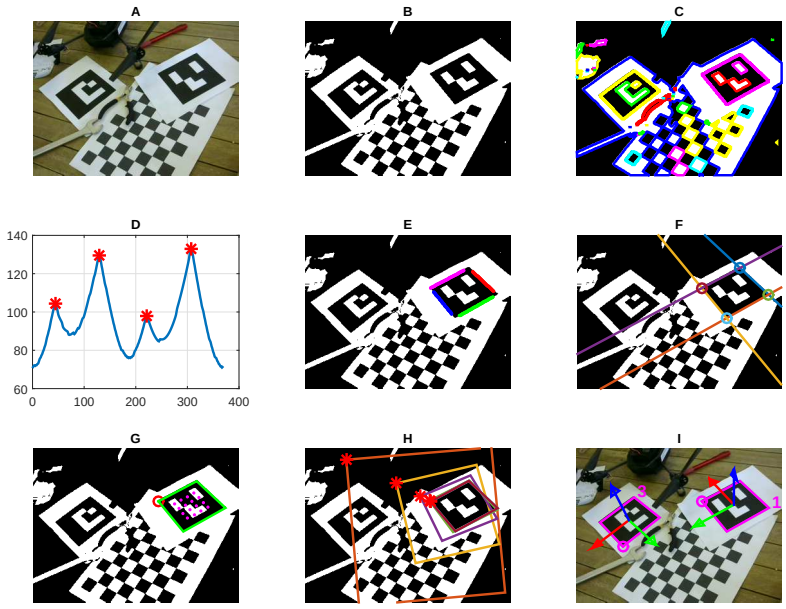

Figure G.1: Overview of steps taken to find Aruco Markers

As described in [2], Aruco markers are 2D markers that can be detected fairly simple with high confidence. Although, there is a method to include the original procedure in Matlab (using the OpenCV library), the method here is a variant that can be compiled and used in the Simulink interface.
In Figure G.1, an overview of our method for finding Aruco markers in an image, and estimating the camera location in the scene is shown.
In the first step, the original colored image is transformed into a gray-scale image. Secondly, to reduce noise in the image a small scale Gausian blur filter is applied and the image is by thresholding transformed into a binary image, as is shown in G.1.B.
From the binary image all the contours are determined, as is shown in G.1.C, by taking the difference between the binary image and its dilated version (using a 3×3 kernel). Then all closed contours are determined using a line following method. For each separate closed contour the center is approximated by taking the mean of the x and y pixels positions of the contour, then, for each pixel on the contour the euclidean distance to this center is calculated. For the remainder of this example we only show the results for the outer contour that determines the right marker, but mention that the same process holds

for all contours. The euclidean distance is shown in sub-image D.
After smoothing the calculated euclidean distance with a Gausian kernel, the most significant peaks can easily be detected. For a square we expect to find four peaks, where each peak represents a corner. These peaks are used to separate the contour-points in four groups. To avoid the effect of rounded corner, due to the earlier used Gausian blur in the second step, we select the data with a padding distance away from the peaks. The different groups are shown in G.1.E. For each group of contour-points we can estimate a line with a regular least squares method. To avoid singularities (lines where the inclination angle approximates infinity) we check in advance for vertical lines. This is done by calculating the mean of all the x positions of the contour-points in the group. When this mean value is below a certain small bound, then we assumed the line to be vertical.
The corner points at sub-pixel level are now given by the intersection-points of the estimated lines, as is shown in G.1.F. Since that the corner points of the potential marker are now known, we can check if the contour is a real Aruco marker. This is done by direct sampling the interior of the perspective transformed square, such that we obtain a set (matrix) of ones and zeros. When this set corresponds with a set in the database of known markers then we have found a Aruco marker.
To handle the marker in different orientations, the set is compared with the database in four orientations. The sample points are in a affine sense taken as is shown in sub-figure G.
Since the orientation of the marker is now known, each corner point can be coupled to a real world location. From this relation the position and orientation of the camera with respect to the marker can be estimated. To do this, an iterative GaussNewton iteration optimization method is used, which search for the rotation and translation parameters between the camera and marker frame. As described in [15] 6.1.3. To speed up the search process, four initial guesses are made in advance with each a different yaw rotation. The guess with the smallest error is used as the initial value for the iterative optimization process. In sub figure H the iterative steps of the process are shown, and in G.1.I the marker frames are projected on the image from the found rotation and translation parameters.

When the real world distance between a set of markers is known, the distance between different sets of markers can be estimated with a single camera. In Figure G.2 the experimental setup is shown, where it is clear that the distances between the markers mounted on the quadrotor are constant and therefore known. In Figure G.3 an image from the camera is shown. The distances between the set of markers mounted on the quadrotor are known, and the same holds for the set of markers mounted on the cube in the left bottom corner. In the above method we estimated the transformation matrix between marker-frame to camera-frame. In this case, we have a transformation from quadrotor-markers to camera frame and a transformation from cube-markers to camera frame. The transformations can be combined to estimate the location of the quadrotor with respect to the cube (inertia frame). This method can thus be used for measuring the quadrotor position and attitude. Experiments show that the method works well for low body velocities. For high velocities, motion blur can make this method less applicable. A possible solution to avoid motion blur is the use of a camera with a lower minimum exposure time, or a camera with a higher sample rate.

Figure G.2: Overview experimental setup for measuring position of the quadrotor in stand.

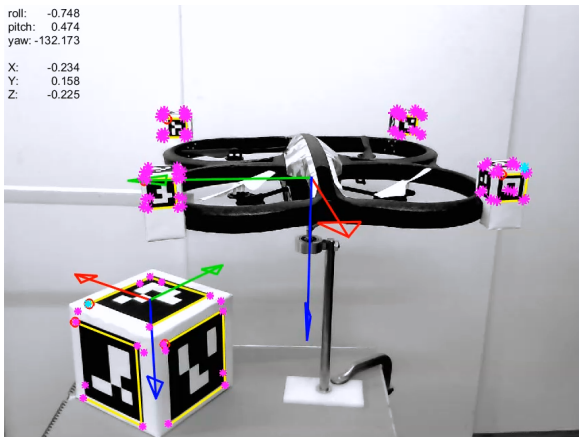

Figure G.3: Detected Markers and estimated quadrotor position with respect to the inertia-frame (marker cube).

Appendix H

Estimation of Global Motion.

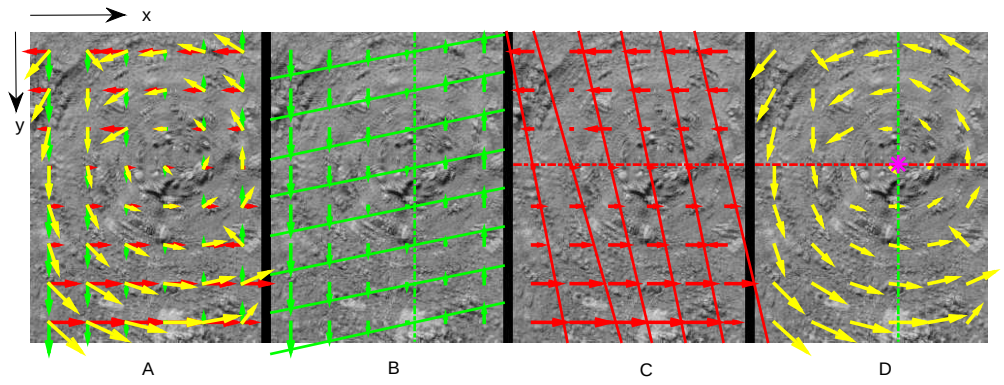

Figure H.1: Overview of steps taken find the center of instanteous rotation.

As already highlighted in 3.2.2 the optical flow is calculated according to [8]. In a second step, to go from optical flow to camera motion, the so called global motion is calculated from the optical flow-field. In this work, an equivalent representation of the optical flow-field is calculated in the form of a rotation around a single center of instantaneous rotation (three parameters). From this center of instantaneous rotation and the rotation angle, the displacement at the center of the image can be calculated, which represents the x-y body translation. The change in yaw angle corresponds to the angle of rotation of the flow-field around the center of instantaneous rotation.
The center of instantaneous rotation, from the $m \times n$ optical flow-field, is calculated by solving two systems of linear equations, using a least square method. As is shown in Figure H.1.A, the optical flow field can be split in its x and y components. As is shown in Figure H.1.B, for each row ($i \in 1 \dots m$), a line can be estimated representing the y-components of the vector field. Although each line has its own slope, all the lines assumed to have a common zero crossing indicated in the image with a dashed line. The solution for finding the slopes and zero crossing of a set of lines is

$$X = (A^T A)^{-1} A^T b \tag{H.1}$$

where

$$b = \begin{bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{1n} \\ x_{21} \\ \vdots \\ x_{2n} \\ \vdots \\ x_{m1} \\ \vdots \\ x_{mn} \end{bmatrix} \qquad A = \begin{bmatrix} 1 & dy_{11} & 0 & \dots & 0 \\ 1 & dy_{12} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & dy_{1n} & 0 & \dots & 0 \\ 1 & 0 & dy_{21} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & dy_{2n} & \dots & 0 \\ 1 & 0 & 0 & \ddots & 0 \\ 1 & 0 & 0 & \dots & dy_{m1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & \dots & dy_{mn} \end{bmatrix} \qquad X = \begin{bmatrix} c_x \\ a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \tag{H.2}$$

where x_{ij} represents the x position of the ij optical flow element in the image, dy_{ij} represents the length of the optical flow vector in y direction, a_i represents the slope of the i'th line, and c_x represents the x location of the center of instantaneous rotation. For calculating c_y the above procedure is repeated with inversed x - y notation, as is show in Figure H.1.C.
The final center of instantaneous rotation is shown in Figure H.1.D. The rotation angle (γ) around the center of instantaneous is estimated by

$$\vec{p}_{ij} = \begin{bmatrix} y_{ij} - c_y \\ x_{ij} - c_x \end{bmatrix} \tag{H.3}$$

$$\vec{v}_{ij} = \begin{bmatrix} dx_{ij} \\ dy_{ij} \end{bmatrix} \tag{H.4}$$

$$\gamma_{ij} = \tan^{-1} \frac{\vec{p}_{ij} \cdot \vec{v}_{ij}}{|\vec{p}_{ij}||\vec{p}_{ij}|} \tag{H.5}$$

$$\gamma = \frac{1}{m\,n} \sum_{i=1}^{m} \sum_{j=1}^{n} \gamma_{ij}. \tag{H.6}$$

i.e. for each flow field vector a rotation around the instantaneous center of rotation is estimated, thereafter, the average rotation is calculated.
The real world translation of the camera can then be calculated by estimating the translation at the center of the image and projecting on the ground, given by

$$dX = \frac{z}{f}(0.5h - c_y)\tan\gamma \tag{H.7}$$

$$dY = \frac{z}{f}(0.5w - c_x)\tan\gamma \tag{H.8}$$

where dX, dY are the camera displacements with respect to the ground, h, w are the image height and width respectively, f is the camera focal length, and z is the vehicle depth or height above the ground.