Circumnavigation with a group of quadrotor helicopters

JOHANNA ORIHUELA SWARTLING

KTH Electrical Engineering

Master's Degree Project
Stockholm, Sweden March 2014

XR-EE-RT 2014:007

Abstract

The primary goal of this thesis has been to implement Collective Circumnavigation. Given a group of unmanned aerial vehicles (UAVs) and the position of a target, the UAVs should approach the target and start circulating around it at a desired distance while forming a regular polygon. In this project quadrotor helicopters were used to conduct experiments and so a secondary goal was to study and implement control of quadrotors.

The overall control of a quadrotor can be divided into three layers; attitude control, position control and path planning. The attitude of a quadrotor describes the orientation, i.e. how much the quadrotor is tilted, with respect to a referance frame. The attitude controller represents the innermost layer of the controller since all translational motions are achieved by tilting the quadrotor. A position controller represents the middle layer and is used to determine how much the quadrotor needs to tilt to get to a certain desired position. The outermost layer is the path planner which decides the trajectory of the quadrotor. The path planner is in this project given by the Circumnavigation algorithm.

For both attitude and position control, the chosen controllers focus on the orientation of the thrust vector. The thrust vector points straight up from the center of mass of the quadrotor and is the sum of the thrust forces caused by the four rotors. The position controller calculates the desired direction of the thrust vector while the attitude controller aligns the real thrust vector with the desired one. Both controllers uses quaternions to describe the orientation of the thrust vector and to prioritize the alignment.

Simulation results show that it was possible to merge the chosen controllers and experimental results show the feasibility of Collective Circumnavigation.

Contents

Chapter 1

Introduction

A quadrotor is a vertical take off and landing vehicle consisting of four arms with a rotor attached to each arm. In recent years the quadrotor has gained much attention from researchers and applications in society are slowly emerging. From a control perspective point of view a quadrotor is a highly interesting object of research and the last years there have been several control methods presented. It is an interesting field of research since it is a quite new and undiscovered area where researches test different control approaches such as MPC [1][2], fuzzy control [3], event-triggered control [4][5], back-stepping control [6], adaptive control [7] etc.

Being small and agile, there has also been research about controlling groups of quadrotors. Interesting areas of research are for example coordinated acrobatics (aerobatics), machine learning, communication and decentralized control.

From the societies point of view there are more and more applications arising. Projects visible today are military applications, monitoring sports competition such as long distance biking and delivery. Example of current applications is Matternet[1] which is an interesting start up company that wants to autonomously deliver medicine and food to places that are hard to reach by ground.

The aim of this thesis has been to implement Collective Circumnavigation[2] at the Smart Mobility Lab[3] of KTH, Stockholm. Given a group of quadrotors and the position of a target, the quadrotors should approach the target and start circulating around it at a desired distance while forming a regular polygon.

In order to complete this task a position controller and an attitude controller have also been studied and are based on [8][9][10]. Both controllers focus on the orientation of the thrust vector in order to control the quadrotor. The thrust vector points straight up from the center of mass of the quadrotor and is the sum of the thrust forces caused by the four rotors. The position controller calculates the desired direction of the thrust vector while the attitude controller aligns the real thrust vector with the desired one.

What is special with these controllers is that both uses quaternions to de-

[1] `http://matternet.us/`

[2] developed by Iman Shames (`imansh@kth.se`), Dimos V. Dimarogonas (`dimos@kth.se`) and Karl H. Johansson (`kallej@kth.se`)

[3] `http://www.kth.se/ees/forskning/strategiska-forskningsomraden/intelligenta-transportsystem/smart-mobility-lab/`

scribe the orientation of the thrust vector. A quaternion is a number in 4D, similar to the complex number, which can describe a rotation about one axis and one angle. The traditional way of describing the orientation of a body with respect to a reference frame is using Euler angles, which describe rotations about the $x-$, $y-$ and z-axis. Quaternions are however less computationally demanding and avoid singularities [10][11]. In addition, the key of having a controller that prioritizes the alignment of the real thrust vector with the desired one lies in the ability of splitting a rotation quaternion into two rotations where the first one aligns the thrust vector with the desired one, and the second aligns the total orientation of the quadrotor with the desired one.

This report is ordered as follows, in chapter 2 a mathematical model of the quadrotor is presented. The relationship between the control signals and the motors, and a model describing the dynamics of the system is presented. In chapter 3 the full control of a quadrotor is presented. The full control is divided into three controllers that work in layers. These are the attitude controller, the position controller and the path planner. Chapter 4 describes the simulation results and that the path planner and the position controller are merged without changes in stability. Chapter 5 describes the SML, and the hardware and software used to conduct the experiments. In chapter 6 the experimental results are presented together with comments on what was done differently from theory. Finally, chapter 7 discusses improvements and future work.

Chapter 2

Mathematical model of a quadrotor

2.1 Presentation of the quadrotor

A quadrotor is a vertical take-off and landing vehicle that consists of a body with four arms. At the end of each arm a rotor spins a propeller, see figure 2.1. The rotors work in pairs so that pair A spins in a clockwise direction and pair B spins in a counter clockwise direction. However, because of the shape of the propellers both pairs create upward thrust forces. Each rotor also produces a torque about its center of rotation. All thrusts and torques produced by each rotor can be summed up to one thrust vector $\mathbf{T} = [0\ 0\ T_z]^T$ acting on the center of mass of the vehicle and a torque vector $\boldsymbol{\tau} = [\tau_x\ \tau_y\ \tau_z]^T$ acting on the center of rotation of the vehicle which in this case also corresponds to the center of mass.

Let the frame B be attached to the vehicle as in figure 2.1. The thrust force is then what causes the quadrotor to lift and move along the z^B-axis. The torque τ_x is what causes the quadrotor to rotate about the x^B-axis, this movement is referred to as *roll*. Similarly, τ_y and τ_z cause rotations about the y^B-axis (*pitch*) and z^B-axis (*yaw*) respectively.

The reason for why the two pairs of rotors spin in opposite directions is the third law of motion. The force making the rotors spin in one direction also produces a counter force that would make the body spin in the other direction. By letting one pair of rotors spin in opposite direction of the other (but with the same velocity), the net force acting on the body is zero.

By tuning the speed of the rotors one can control four kinds of motions. These are regarded as the control input $\mathbf{U} = [U_1\ U_2\ U_3\ U_4]^T$. U_1 corresponds to the total thrust and is controlled by letting pair A and pair B spin with the same velocity. When the thrust $T_z > mg$ the quadrotor lifts and when $T_z = mg$ the quadrotor hovers at a fixed height. U_2 corresponds to the roll movement and is achieved by letting one rotor belonging to pair B increase its velocity with a certain Δv while letting the other rotor decrease its velocity with the same Δv. In this way the total thrust remains the same but the arm with the rotor spinning faster will elevate while the other lowers, in this way the quadrotor tilts towards the lower arm. U_3 corresponds to the pitch movement

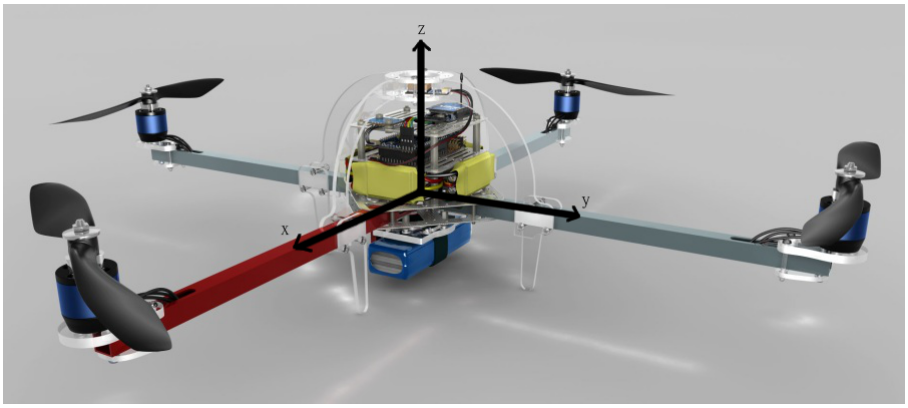

Figure 2.1: A quadrotor and the body frame B which moves and rotates along with the vehicle and with its origin attached to the center of mass. The pair of rotors along the x-axis is referred to as pair A, and pair B is the pair along the y-axis.

and is achieved by the same way as for U_2 but with pair A. U_4 corresponds to the yaw movement and is achieved by letting one pair increase its velocity and the other decrease its velocity with the same amount. In this way the total thrust remains the same but, as described above, the net force acting on the body is no longer zero and the quadrotor will rotate about its z^B-axis [12].

A quadrotor has six degrees of freedom, meaning that it can perform six kinds of movements. Three are translational movements (along the x-, y- and z-axis of a world reference frame) and three are rotational movements (roll, pitch and yaw as described above). As stated one can control four kinds of movements through the rotors. Such a system is referred to as an underactuated system since the degrees of freedom are more than the ones that can be controlled directly. The two degrees of freedom that cannot directly be controlled are the lateral movements along the x- and y-axis of the world reference frame. These are instead controlled by letting the quadrotor tilt, i.e. by adjusting the roll and pitch angles.

In the following a mathematical model of the quadrotor is presented in order to understand how the quadrotor moves depending on the control inputs.

2.2 The Newton-Euler Model

What one wants to obtain when putting up a model are equations of motions that describes the dynamics of the system. Within robotics there are two common formalisms used to describe the dynamics of a plant. One is the Euler-Lagrange formalism that uses work and energy to describe the systems behaviour. The other one is the Newton-Euler formalism that uses force and torque to describe the dynamics of the system [13].

The model given in this section is a common model derived from the Newton-Euler formalism. Through Newton's second law of motion, equations describing the dynamics are derived by summing together all external forces and moments acting on the center of mass (which also is the center of rotation) of the quadro-

tor. Of course, the more accurate model the easier to control the real plant. Many more or less detailed models have been presented, e.g. in [14], [15]. In this thesis the model presented in [12] stands as a reference. The difference is that a more simple model will be provided here with only the main acting forces and torques included.

2.2.1 The Newton part of the Newton-Euler model

With respect to a body-fixed frame B let $\mathbf{V}^B = [V_x^B \; V_y^B \; V_z^B]^T$ be the linear velocity of the quadrotor and let $\boldsymbol{\omega}^B = [\omega_x^B \; \omega_y^B \; \omega_z^B]^T$ be the angular velocity of the quadrotor. Another way of interpreting $\mathbf{V}^B$ and $\boldsymbol{\omega}^B$ is how the coordinate system B changes with respect to a fixed world frame W. The Newton-Euler model is given by

$$\begin{bmatrix} m\mathbf{J} & \mathbf{0} \\ \mathbf{0} & \mathbf{J} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{V}}^B \\ \dot{\boldsymbol{\omega}}^B \end{bmatrix} + \begin{bmatrix} \boldsymbol{\omega}^B \times (m\mathbf{V}^B) \\ \boldsymbol{\omega}^B \times (\mathbf{J}\boldsymbol{\omega}^B) \end{bmatrix} = \begin{bmatrix} \mathbf{F}^B \\ \boldsymbol{\tau}^B \end{bmatrix} \tag{2.1}$$

where

$$\mathbf{J} = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix}$$

is the matrix of inertia of the quadrotor and $\mathbf{F}^B$ and $\boldsymbol{\tau}^B$ are the forces and torques acting on the center of mass of the quadrotor and given in the body frame B. The next step is to identify $\mathbf{F}^B$ and $\boldsymbol{\tau}^B$.

2.2.2 The Euler part of the Newton-Euler model

The external force acting on the quadrotor is caused by the gravity force, $\mathbf{F}_g^W = [0 \; 0 \; -mg]^T$ given in the world frame. If the z^B-axis of the quadrotor is parallel to the z^W-axis of the world frame, then $\mathbf{F}_g^B = \mathbf{F}_g^W$, but what about when the quadrotor is tilted? This is where the Euler part comes in.

Euler angles are the angles describing how a rigid body is orientated with respect to a reference frame. In the case of the quadrotor this corresponds to the angles that describe the orientation of the body frame B with respect to the world frame W. These angles can be identified as the roll (ϕ), pitch (θ) and yaw (ψ) angles described in section 2.1 .

To relate a vector in B to W one can use rotation matrices. Using the Euler angles a rotation matrix projects a vector from one frame to the other. Let R_{B2W} be the rotation matrix used to transform from B to W

$$R_{B2W} = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\phi s_\theta s_\phi & s_\psi s_\phi + c_\psi s_\theta c_\phi \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta s_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}$$

where s_x, c_x stands for $\sin x$ and $\cos x$ respectively. To do the inverse transform (from W to B) the rotation matrix is given by

$$R_{W2B} = (R_{B2W})^T$$

Hence the gravity force vector in the body frame is given by

$$\mathbf{F}_g^B = (R_{B2W})^T \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} = \begin{bmatrix} mgs_\theta \\ -mgc_\theta s_\phi \\ -mgc_\theta c_\phi \end{bmatrix}$$

2.3 Identifying the Control Inputs

As described in section 2.1 there are four control inputs to the system. These relate to the speed of the rotors by (see [12], [14]).

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} b(\Omega_{R1}^2 + \Omega_{R2}^2 + \Omega_{R3}^2 + \Omega_{R4}^2) \\ bl(\Omega_{R3}^2 - \Omega_{R1}^2) \\ bl(\Omega_{R4}^2 - \Omega_{R2}^2) \\ d(-\Omega_{R1}^2 + \Omega_{R2}^2 - \Omega_{R3}^2 + \Omega_{R4}^2) \end{bmatrix} \tag{2.2}$$

where Ω_{Ri} is the velocity of the i:th rotor, l is the distance from a rotor to the center of mass of the quadrotor, and b and d are constants called the thrust factor and the drag factor respectively. In this thesis neither l, b nor d are identified. What is of importance in (2.2) are the following three observations:
i) The values of the control inputs are directly related to the squared speed of the rotors.
ii) As concluded in section 2.1 U_1 corresponds to an upward thrust force $\mathbf{F}_U^B = \begin{bmatrix} 0 \\ 0 \\ U_1 \end{bmatrix}$ acting on the center of mass of body.
iii) As concluded in section 2.1, $U_2 - U_4$ correspond to the torque $\boldsymbol{\tau}_U^B = \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix}$ acting on the center of mass of the body.

In this way the total external forces and torques acting on the body are

$$\begin{bmatrix} \mathbf{F}^B \\ \boldsymbol{\tau}^B \end{bmatrix} = \begin{bmatrix} mgs_\theta \\ -mgc_\theta s_\phi \\ -mgc_\theta c_\phi + U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

These are the main dominating forces. In more detailed models one can take into account the drag force (or air resistance) which is a kind of friction force acting on the opposite direction of the motion and gyroscopic effects from the motors which is a torque depending of the moment of inertia of the rotors and the rotor speeds [12] [16].

2.4 Controlling the Quadrotor

What is left now to complete the model is to put every part of the equation (2.1) together, obtaining a model that describes the dynamics in the world frame

W. Introduce the state space variables $\mathbf{x}^W = [x \ y \ z \ \phi \ \theta \ \psi]^T$ which represents the (x, y, z)-position of the quadrotor in the world frame and describes how the quadrotor is tilted by (ϕ, θ, ψ) with respect to the world frame.

From (2.1) one can see that the Newton-Euler model contains the linear velocity, angular velocity and angular acceleration *in the body frame* B. From section 2.2.2 one can identify

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R_{B2W}\mathbf{V}^B \tag{2.3}$$

and

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = R_{B2W}\dot{\mathbf{V}}^B \tag{2.4}$$

The next question is how $\dot{\boldsymbol{\phi}} = [\dot{\phi} \ \dot{\theta} \ \dot{\psi}]^T$ should relate to $\boldsymbol{\omega}^B$. The easiest way to think is to see $\dot{\boldsymbol{\phi}}$ as the rate of change of roll, pitch and yaw respectively, while $\boldsymbol{\omega}^B$ represents the velocity vector in the body frame pointing along the axis of rotation. When roll, pitch and yaw are small (which in general is the case since the quadrotor flies near hovering state) then $\dot{\boldsymbol{\phi}} \approx \boldsymbol{\omega}^B$, but if not, they are related by a transfer matrix T_{B2W} [15]

$$T_{B2W} = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi/c_\theta & c_\phi/c_\theta \end{bmatrix}$$

where $t_x = \tan x$, and so

$$\dot{\boldsymbol{\phi}} = T_{B2W}\,\boldsymbol{\omega}^B$$
$$\ddot{\boldsymbol{\phi}} = T_{B2W}\,\dot{\boldsymbol{\omega}}^B$$

Calculating for small angles (T_{B2W} =identity matrix) and using (2.1), (2.3) and (2.4) the following state space equations are obtained (see [12], [14], [15])

$$\begin{cases} \ddot{x} = -(c_\phi s_\theta c_\psi + s_\phi s_\psi)\frac{U_1}{m} \\ \ddot{y} = -(c_\phi s_\theta s_\psi - s_\phi c_\psi)\frac{U_1}{m} \\ \ddot{z} = g - (c_\phi c_\theta)\frac{U_1}{m} \\ \ddot{\phi} = \dot{\phi}\dot{\psi}\,\frac{J_z - J_x}{J_y} + \frac{U_2}{J_y} \\ \ddot{\theta} = \dot{\theta}\dot{\psi}\,\frac{J_y - J_z}{J_x} + \frac{U_3}{J_x} \\ \ddot{\psi} = \dot{\theta}\dot{\phi}\,\frac{J_x - J_y}{J_z} + \frac{U_4}{J_z} \end{cases}$$

From these equations one can make two observations. First is that all translational motions depend on U_1 and how the quadrotor is tilted and second that, clearly, the system is non-linear and coupled. Such properties generally makes it harder to find a good control law. In this thesis however, the chosen controllers are based on another parametrization (through quaternions) which has the benefit of not having to deal with rotation matrices.

Chapter 3

Control of the Quadrotors

The full control of a quadrotor can be divided into three layers; attitude control, position control and path planning. The innermost layer is the attitude control, which is about controlling the orientation of the quadrotor with respect to a reference frame and stabilizing it. The position controller makes sure that the quadrotor tracks a certain desired position and the path planner decides where the robot should go while, in the general case, fulfilling certain requirements such as formation and/or obstacle avoidance.

Figure 3 describes an ideal and general model of the control system during one iteration. First the collected sensor data which gives information about the quadrotors position is sent to the path planner. Based on the quadrotors current position, $\mathbf{x}(k)$, the path planner calculates the next desired position, $\mathbf{x}_d(k+1)$, which is sent as a reference to the position controller. The position controller then calculates how much the quadrotor needs to tilt and the magnitude of the thrust vector in order to reach $\mathbf{x}_d(k+1)$. Recalling from section 2.3, the magnitude of the thrust vector is regarded as the control input U_1 in (2.1) and is sent directly to the speed controllers of the motors. The desired tilt (e.g. in terms of roll, pitch and yaw) of the quadrotor with respect to the inertial frame is sent to the attitude controller, which then sends the control inputs U_2, U_3 and U_4 to the motors. Recall from (2.1) that the control inputs $U_1 - U_4$ are proportional to the speed of the motors and that U_1 controls the total thrust, and U_2, U_3 and U_4 control roll, pitch and yaw respectively.

As mentioned, figure 3 represents an ideal model of the control system. How this control system has been implemented at the SML differs somewhat. This will be explained more in detail in chapter 6, but the main difference is that the position controller and path planner can easily be implemented in computers at the lab, while the attitude controller already is programmed on a control board on the quadrotor. To modify the set up of the control board is hard (it would at least be a time consuming task) and therefore the attitude control described in this section will only be used in simulation.

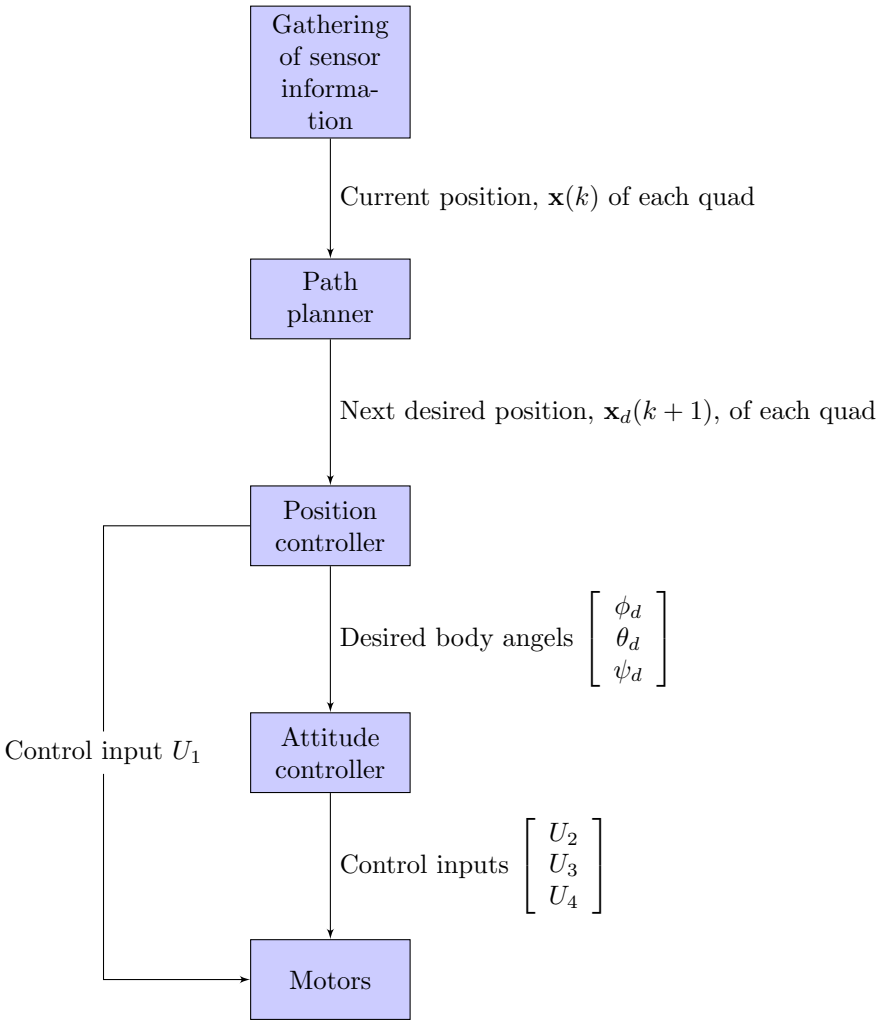

Figure 3. An ideal model of the control process during one iteration.

3.1 Reference Frames

In order to present the controllers three reference frames need to be defined. First define a global frame W. A point $(x, y, z)^W$ in this frame describes the global position of the quadrotor. Next, define the body frame B which has its origin attached to the center of mass of the quadrotor as in figure 3.1. The x^B-axis always points along the "nose" arm (or front arm) and the y^B-axis always points along the left arm of the quadrotor. The z^B-axis points straight up from the center of mass of the quadrotor. Note that B always tilts, rotates and moves along with the quadrotor. Last, define the inertial frame I. This frame will always share the same origin as B but remains fixed with respect to the world frame W. This means that I also moves along with the quadrotor

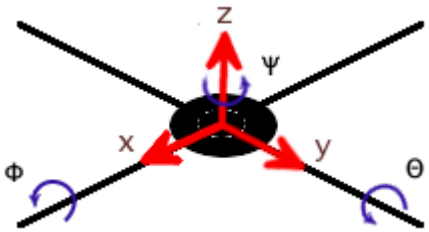

Figure 3.1: A model of a quadrotor describing the body angles ϕ =roll, θ =pitch and ψ =yaw. There are two coordinate systems attached to the center of mass of the quadrotor; the body-fixed frame B which rotates along with the quadrotor, and the inertial frame I that is fixed (does not rotate) and works as a reference when describing the tilt of frame B.

but never tilts nor rotates together with it as B does. I will therefore always have its x-, y- and z-axis parallel to the x-, y- and z-axis of the world frame.

3.2 Attitude Control

3.2.1 What is attitude control?

Attitude control is about controlling a vehicles orientation, that is controlling the body angles roll, pitch and yaw with respect to the inertial frame I. In the case of a quadrotor, any translational motion is achieved by tilting the quadrotor and therefore an attitude controller is the innermost layer in the control of a quadrotor. Being the innermost layer the attitude control is also what stabilizes the quadrotor and the most essential part in the control of a quadrotor [17]. Basically its task is to track the desired body angles roll, pitch and yaw, described by figure 3.1.

The attitude controllers control the orientation of the quadrotor through the torques acting on the it since they both directly affect the angular acceleration (see equation 2.1) and can be seen as the control input to the speed controllers of the motors, as described in section 2.4.

3.2.2 State of the art

Being the most essential part in the control of a quadrotor several different methods have been developed to control the attitude of a quadrotor. Different techniques involve different versions of PID control [21][22], back-stepping control [17], adaptive control [24][25] and much more.

Most attitude controllers, despite using different techniques, are designed to directly control the body angles roll, pitch and yaw. The concept is shown in figure 3.2.

These controllers are usually based on a model similar to the Newton-Euler model presented in chapter 2 which leads to a state space representation that is non-linear and coupled. The attitude controller chosen for this thesis uses however quaternions to describe the quadrotors orientation in 3D space. This means

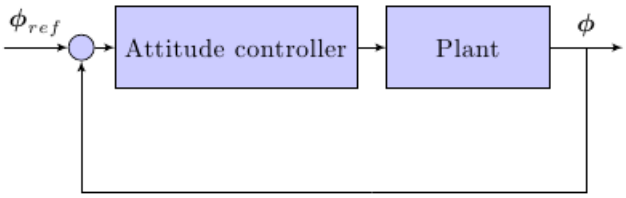

Figure 3.2: Most attitude controllers are designed to track a desired body angles $\boldsymbol{\phi} = [\phi \ \ \theta \ \ \psi]^T$.

that the attitude controller controls the quaternion describing the orientation rather than the body angles roll, pitch and yaw. Even though quaternions might conceptually be a bit hard to grasp, they are not as computationally demanding as rotation matrices used for Euler angles and in [10] the translational dynamics and yaw dynamics are decoupled from each other thanks to the quaternion parametrization.

3.2.3 Quaternions

The attitude controller implemented in this project describes both the dynamics of the quadrotor and the attitude controller in quaternion space. Therefore some background theory about quaternions will be described in this section.

Quaternions are generally used to describe rotations in 3D space. Common areas of usage are control of robot arms, control of satellites and within the field of computer graphics. When it comes to attitude control, it is some times more simple and effective to use quaternions over Euler angles (pitch, roll and yaw). The main reason for this is how simple quaternions can describe a 3D rotation with only one vector and one angle. This leads to simplified models, which in turn leads to a lower computational cost [11]. Another popular reason to use quaternions is to avoid the problem of *Gimbal lock* [1] which would be of great use when for example performing advanced flight manoeuvres or acrobatics.

However for the controller described in this project (presented in [8]), the main usage of quaternions is the benefit that comes along with some of the properties of quaternions which enables the controller to align its thrust direction with some desired thrust direction. What aligning of thrust directions means is described in section 3.2.4.

Definition

Historically, the search for quaternions started when the existence of the complex number had been presented. Mathematicians asked themselves if there could be a 3D equivalent. W. R. Hamilton is the scientist recognised as the inventor of quaternions. A quaternion $\mathbf{q}$ is defined as

$$\mathbf{q} = \begin{bmatrix} \mathbf{v} \\ w \end{bmatrix} \tag{3.1}$$

[1] When rotating three axis simultaneously and when, at a certain time, two of these axis align, then there is a loss of one degree of freedom. This is called a Gimbal lock.

where $\mathbf{v} \in \mathbb{R}^3$, $w \in \mathbb{R}$, but also as

$$\mathbf{q} = w + xi + yj + zk,$$

where $w, x, y, z \in \mathbb{R}$ and $i^2 = j^2 = k^2 = ijk = -1$. This might seem a bit confusing with scalars, vectors and imaginary numbers mixed together, but one can compare it with the complex numbers. The complex number $z = a + ib$ has one real part a and one imaginary part b. Similarly a quaternion has a real part w and an imaginary part x, y, z.

To get a feeling of why the imaginary part also can be represented as a vector one can analyse the imaginary numbers i, j, k. Let $\mathbf{i}, \mathbf{j}, \mathbf{k}$ be the unit Cartesian vectors (commonly written as $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$), then i, j, k are defined as the *quaternion units*

$$i = [0 \quad \mathbf{i}], \qquad j = [0 \quad \mathbf{j}], \qquad k = [0 \quad \mathbf{k}].$$

Just like a vector in any Cartesian coordinate system can be expressed as $\mathbf{a} = x\mathbf{e}_x + y\mathbf{e}_y + z\mathbf{e}_z$, then similarly $\mathbf{v}$ in (3.1) describes a vector $\mathbf{v} = xi + yj + zk$ in the 3D complex room. For more information about quaternions check [19] and [20].

Rotation with quaternions

As Euler stated in his theorem of rotation, a rigid body that rotates in 3D space can be described by *one* vector and *one* angle if there is some point of the body that remains fixed. Another way of thinking is that if any two (Cartesian) coordinate systems share the same origin, then they are related by a rotation about some fixed axis.

A *rotation quaternion* or *unit-norm quaternion*[2] is a quaternion of length 1, i.e. $\| \mathbf{q} \| = 1$, and what it does is to describe a rotation about a fixed axis.

A quaternion is a vector in 4D, but it is used to rotate any vector in 3D. To get an intuitive feeling for this rotation, compare with the complex numbers. It is known that a complex number $\mathbf{z} = a + ib$ can be seen as a vector in the complex plane with length $|\mathbf{z}|$ and argument θ. One can also see it as multiplying the vector $|\mathbf{z}|$ (lying on the real axis) with the unit vector $e^{i\theta}$, and the result is a rotation by an angle θ. So even if the complex number $\mathbf{z}$ lies in the two-dimensional complex plane it can describe a one-dimensional rotation. A rotation quaternion does exactly this, but since it has three imaginary components it describes a rotation in three dimensions.

The rotation quaternion

$$\mathbf{q} = \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix} = \begin{bmatrix} \mathbf{v} \sin \frac{\alpha}{2} \\ \cos \frac{\alpha}{2} \end{bmatrix} \tag{3.2}$$

where $\mathbf{v}$ is a unit vector, describes a rotation by the angle α about the axis $\mathbf{v}$.

The case with quaternions is similar to the case with complex numbers, but not exactly the same. In 1D (rotation about one axis), the rotation angle is given by some point on the unit circle $C = \{w, x \in \mathbb{R}^2 | w^2 + x^2 = 1\}$, just like

[2]Not to be confused with a unit quaternion, i.e. $\mathbf{q} = [0 \quad \hat{\mathbf{v}}]$, where $\hat{\mathbf{v}}$ is a unit vector.

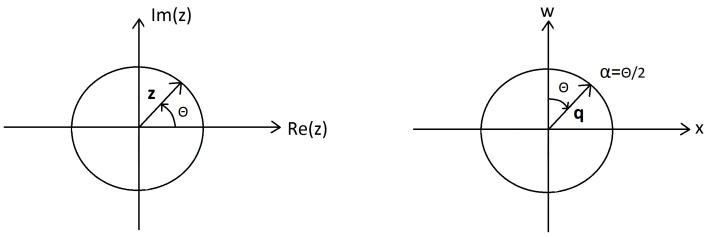

Figure 3.3: Comparing complex numbers with quaternions. For the complex numbers, a unit vector lying on the unit circle describes a rotation by the angle θ, which is measured as the counter clockwise angle between the positive real axis and the unit vector. A quaternion with only two components $\mathbf{q} = w+ix+j0+k0$ also describes a rotation. The difference is that angle θ, which describes the angle between the positive real axis (w) and the unit vector ($\mathbf{q}$) describes half the angle of rotation. This means that the quaternion in this figure describes a rotation by the angle $\alpha = 2\theta$ and about the axis $\mathbf{v} = [1\ \ 0\ \ 0]'$.

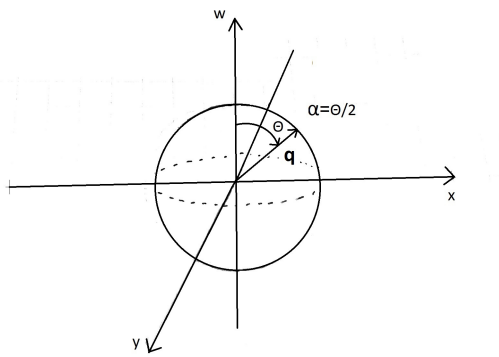

Figure 3.4: A rotation quaternion describing a rotation about two axis, $\mathbf{q} = w + ix + jy + k0$, is a unit vector on the unit sphere $S = \{w, x, y \in \mathbb{R}^3 | w^2 + x^2 + y^2 = 1\}$. $w = 1 \Leftrightarrow \alpha = 0$ describes a rotation by 0 rad, $w = -1 \Leftrightarrow \alpha = 2\pi$ describes a rotation by 2π rad.

in the case with complex numbers, see figure 3.3. To keep things apart, let α be the actual rotation angle and let $\theta = \frac{\alpha}{2}$ be the angle between the $w-$axis and the unit-norm quaternion $\mathbf{q}$. The difference, comparing with the complex numbers, is that $\theta = \frac{\pi}{2}\, rad$ represent a rotation of $\alpha = \pi\, rad$ about the x-axis, and $\theta = \pi\, rad$ represent a rotation of $\alpha = 2\pi\, rad$ about the x-axis.

This way of representing the rotation angle is easier to motivate in the 2D case (rotation about two axis). Here the rotation is described by a point on the unit sphere $S = \{w, x, y \in \mathbb{R}^3 | w^2 + x^2 + y^2 = 1\}$ as seen in 3.4. Any point on the unit sphere, pointed by a rotation quaternion describes a unit vector $\mathbf{v}$ (the axis of rotation) and an angle θ (half the angle of rotation).

One can notice that if the rotation angle was to be represented as it is for the complex numbers (θ = the angle between the unit vector $e^{i\theta}$ and the positive real axis = the rotation angle), then the south pole ($\theta = \pi$) would represent a rotation of $\alpha = \pi\, rad$ about the x-axis but also a rotation of $\alpha = \pi\, rad$ about the y-axis. Since every point on the unit sphere represent a rotation about *one* axis and by *one* angle, the south pole would be equivalent to saying "a rotation of $\pi\, rad$ about the x-axis"="a rotation of $\pi\, rad$ about the y-axis", which is not true. When instead letting θ describe half the angle of rotation, $\theta = \frac{\alpha}{2}$, then the south pole ($\theta = \pi$, $\alpha = 2\pi$) is equivalent to saying "a rotation of $2\pi\, rad$ about the x-axis"="a rotation of $2\pi\, rad$ about the y-axis", which is true. This is more a motivation than an explanation to why θ describes half the rotation angle.

Summing up and as seen if figure 3.4, the north-pole represents a rotation of $0\, rad$ about both the x-axis and y-axis, and the south-pole represents a rotation about $2\pi\, rad$ about both the x-axis and y-axis. Notice that $w = 1$ at the north-pole and $w = -1$ at the south-pole and that these are the only two points on the sphere describing a rotation where the orientation remains the same as before the rotation[3].

Continuing this logic, for a rotation in 3D (rotation about three axis), the rotation angle is given by a point of the unit sphere in 4D, $S = \{w, x, y, x \in \mathbb{R}^4 | w^2 + x^2 + y^2 + z^2 = 1\}$. For obvious reasons S cannot be presented in a figure, but it is enough to understand that, analogue to the 2D-case, $w = \pm 1$ represents no change of orientation, and $w = 0$ represents a rotation of $\alpha = \pi\, rad$.

Some properties

In order to understand the control objective some properties of the quaternions will be presented. Let $\mathbf{q}$ represent the rotation between a coordinate system X and a coordinate system Y.

$$X \overset{q}{\curvearrowright} Y$$

First, by multiplying (3.2) with -1 one can understand that $-\mathbf{q}$ also is a mapping from X to Y since it also is a rotation about the axis $\mathbf{v}$ with the angle $\alpha \pm (2k+1)2\pi$, $k \in \mathbb{N}_0$ [9].

$$X \overset{-q}{\curvearrowright} Y \tag{3.3}$$

[3] A coordinate system rotated about an axis by $0\, rad$ or by $2\pi\, rad$ will not change its orientation.

Recalling $q_w = \cos(\frac{\alpha}{2}), \quad q_w \in [-1, 1], \quad \alpha \in [0, 2\pi]$ and using the property described in (3.3), one can define the rotation

$$\hat{\mathbf{q}} = \text{sgn}(q_w)\mathbf{q} \tag{3.4}$$

where sgn is the sign-function[4], $\hat{q}_w = \cos(\frac{\hat{\alpha}}{2})$ and $\hat{\alpha} \in [0, \pi]$ since $\hat{q}_w$ always will be positive. The interpretation of $\hat{\mathbf{q}}$ is that it will represent the same mapping as $\mathbf{q}$ but with the smallest possible angle $\hat{\alpha}$.

Second, the conjugate of a quaternion is obtained by changing the sign of the imaginary part (compare with the complex numbers)

$$\mathbf{q}^* = [-q_x \quad -q_y \quad -q_z \quad q_w]^T$$

A conjugate quaternion describes the reverse mapping from Y to X.

$$Y \overset{q^*}{\curvearrowright} X$$

Third, two rotations described by two unit-norm quaternions can be described as one rotation obtained by multiplying the two quaternions. It is important to know that the product of two quaternions is non-commutative, meaning that the order in which they are multiplied matters. A multiplication of two quaternions can be calculated by the matrix multiplication

$$\mathbf{r} = \mathbf{q} \circ \mathbf{p} = \begin{bmatrix} r_x \\ r_y \\ r_z \\ r_w \end{bmatrix} = \tag{3.5}$$

$$= \begin{bmatrix} p_w & p_z & -p_y & p_x \\ -p_z & p_w & p_x & p_y \\ p_y & -p_x & p_w & p_z \\ -p_x & -p_y & -p_z & p_w \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix} = \mathbf{W}(\mathbf{p})\mathbf{q}$$

Fourth, if $\boldsymbol{\omega} = [\omega_x \quad \omega_y \quad \omega_z]^T$ is the angular velocity of X, then the derivative $\dot{\mathbf{q}}$ can be expressed as

$$\dot{\mathbf{q}} = -\frac{1}{2}\mathbf{W}(\mathbf{q}) \begin{bmatrix} \boldsymbol{\omega} \\ 0 \end{bmatrix} = -\frac{1}{2}\mathbf{W}_R(\mathbf{q})\boldsymbol{\omega}$$

where $\mathbf{W}_R$ is the same as $\mathbf{W}$ without the last column [8][18].

Fifth, a vector in Y can be described as a vector in X through

$$\mathbf{a}^X = \mathbf{R}(\mathbf{q})\mathbf{a}^Y \tag{3.6}$$

where $\mathbf{a}^X, \mathbf{a}^Y$ describe the same vector but in different frames and

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_z q_w) & 2(q_x q_z + q_y q_w) \\ 2(q_x q_y + q_z q_w) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_x q_w) \\ 2(q_x q_z - q_y q_w) & 2(q_y q_z + q_x q_w) & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{3.7}$$

[4] $\text{sgn}(a) = \begin{cases} 1 & a \geq 0 \\ -1 & a < 0 \end{cases}$

3.2.4 Problem statement

Now that a relevant background information has been presented a full attitude control problem definition can be formulated. The controller implemented in this project is designed (in [8]) so that it prioritizes the alignment of the quadrotors total thrustvector $\mathbf{T}$. This is done by exploiting the properties of quaternions presented in the previous section.

Let B and I be the body-fixed and inertial frames described earlier with the origins fixed at the center of mass of the quadrotor. Recall that the thrust vector is the sum of all the thrust forces coming from the four rotors and always points straight up from the center of mass of the quadrotor. This means that it will always point along the positive z-axis of the body frame B. To determine the direction of the thrust vector with respect to the inertial frame I is a way of describing how the quadrotor is tilted. Since all translational motions are acquired by tilting the quadrotor, then the desired thrust direction can for example be the output of a position controller and so the aim of the attitude controller is to align the quadrotors thrust direction with the desired one [8].

Let the desired tilt be represented by a desired body frame D. Recalling that a unit-norm quaternion describes a rotation between two different frames, define the following quaternions:

$$\mathbf{q}_b : \; I \overset{q_b}{\curvearrowright} B$$

thus, recalling that the conjugate describes the reverse mapping:

$$\mathbf{q}_\mathbf{b}^* : \; B \overset{q_b^*}{\curvearrowright} I$$

and define

$$\mathbf{q}_d : \; I \overset{q_d}{\curvearrowright} D \; .$$

Using the property described in equation (3.5), define $\mathbf{q}$ as

$$\mathbf{q} = \mathbf{q}_\mathbf{b}^* \circ \mathbf{q}_\mathbf{d}$$

$$\mathbf{q} : \; B \overset{q}{\curvearrowright} D \; .$$

Thus $\mathbf{q}$ describes the rotation error between the frames B and D, and a first control criteria can be presented. It is desired[5] that

$$\mathbf{q} \to \begin{bmatrix} 0 & 0 & 0 & \pm 1 \end{bmatrix}^T, \quad \text{as } t \to \infty$$

Using (3.5) again, $\mathbf{q}$ can be decomposed to

$$\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \hat{\mathbf{q}} \, sgn(q_4) = (\mathbf{q}_{xy} \circ \mathbf{q}_z) \, \mathrm{sgn}(q_4) \tag{3.8}$$

[5]Recalling that $w = 1$ represents a rotation of $0\,rad$ and $w = -1$ represents a rotation of $2\pi\,rad$

where

$$\mathbf{q}_{xy} = \begin{bmatrix} q_x & q_y & 0 & q_p \end{bmatrix}^T$$

$$\mathbf{q}_z = \begin{bmatrix} 0 & 0 & q_z & q_w \end{bmatrix}^T$$

and

$$\mathbf{q}_{xy}: \ B \overset{q_{xy}}{\curvearrowright} A$$

$$\mathbf{q}_z: \ A \overset{q_z}{\curvearrowright} D$$

where A is an auxiliary frame.

The idea of dividing $\mathbf{q}$ into two rotations is to first prioritize the alignment of the z^B-axis with the z^D-axis, which is accomplished by rotating B about its x- and y-axis. This rotation is described by $\mathbf{q}_{xy}$ and the "in-between" frame is called A. The second rotation then aligns the x^A- and y^A-axis with the x^D- and y^D-axis, which is accomplished by a rotating A about its z-axis. The corresponding angles of rotation are

$$\phi = 2\arccos(q_p) \quad \text{and} \quad \gamma = 2\arccos(q_w).$$

It follows from (3.8) that $\phi, \gamma \in [0, \pi]$, ϕ describing the smallest possible error angle between B and A, and γ describing the smallest possible error angle between A and D. It is desired that $\phi = \gamma = 0$, meaning the frames are perfectly aligned. In other words, the desired equilibrium points are

$$\mathbf{q} = [0 \ 0 \ 0 \ \pm 1]^T \tag{3.9}$$

which by (3.8) means

$$\mathbf{q}_{xy} = \mathbf{q}_z = [0 \ 0 \ 0 \ 1]^T$$

and

$$\phi = \gamma = 0.$$

3.2.5 Plant model

The model in [8] used to describe the dynamics of the quadrotor is given by

$$\dot{\mathbf{q}} = -\frac{1}{2}\mathbf{W}_R(\mathbf{q})\boldsymbol{\omega} \tag{3.10}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}(\mathbf{J}\boldsymbol{\omega} \times \boldsymbol{\omega}) + \mathbf{J}^{-1}\boldsymbol{\tau} \tag{3.11}$$

where $\dot{\mathbf{q}}$ describes how $\mathbf{q}$ evolves with time, and $\dot{\boldsymbol{\omega}}$ is the same as in the Newton-Euler model, (2.1), and describes how the angular acceleration of the quadrotor evolves with time. $\mathbf{J}$ is the matrix of inertia of the quadrotor and can be regarded as a diagonal matrix. $\boldsymbol{\tau} = [\tau_2 \ \tau_3 \ \tau_4]'$ is the torque vector acting on the quadrotor, which components τ_2, τ_3, τ_4 cause the rotations roll, pitch and yaw respectively. As stated in section 2.4, $\boldsymbol{\tau}$ corresponds to $[U_2 \ U_3 \ U_4]'$. In other words, by changing the speed of the motors the torque acting on the quadrotor is controlled, therefore it is through $\boldsymbol{\tau}$ that the attitude is controlled.

3.2.6 Control law

Since in this thesis, the attitude controller is only used for simulation, there will not be a detailed explanation of the design of the control law. If interested, the reader can refer to [9] and [8]. The control law is given by

$$\boldsymbol{\tau} = \mathbf{T}(\mathbf{q}) - \mathbf{D}(\mathbf{x})\boldsymbol{\omega} \tag{3.12}$$

The explicit expressions for $\mathbf{T}(\mathbf{q})$ and $\mathbf{D}(\mathbf{x})$ can be found in [8], in this section it is only the concept of the control law design that will be explained.

Let $V(\mathbf{x})$ describe the total energy of the closed-loop system:

$$V(\mathbf{x}) = E_{rot}(\boldsymbol{\omega}) + E_{pot}(\mathbf{q}) = \frac{1}{2}\boldsymbol{\omega}^T\mathbf{J}\boldsymbol{\omega} + E_{pot}(\mathbf{q}).$$

The total energy of the system is in fact given by E_{rot} which describes rotational energy, but a clever move is to add the artificial potential energy $E_{pot}(\mathbf{q})$. The fact of it being artificial comes from E_{pot} only depending on $\mathbf{q}$. More precisely, E_{pot} depend on the rotation angles ϕ and γ

$$E_{pot}(\mathbf{q}) = E_\phi(\phi) + E_\theta(\phi, \gamma).$$

E_ϕ and E_θ are designed so that they equal zero for $\phi = 0$ and $\gamma = 0$ respectively. According to (3.9) this means that $E_{pot} = 0$ for the desired equilibrium points, hence $V(\mathbf{x})$ behaves so that it equals zero for the desired equilibrium points and is greater than zero everywhere else.

In terms of Lyapunov stability $V(\mathbf{x})$ can be seen as a Lyapunov candidate to prove stability of the desired equilibrium points [26]. In order to guarantee stability $V(\mathbf{x})$ needs to satisfy
- $V(\mathbf{x}) > 0$ everywhere except for the desired equilibrium points,
- $V(\mathbf{x}) = 0$ for the equilibrium points, and
- $\dot{V}(\mathbf{x}) \leq 0$.

Analysing $\dot{V}(\mathbf{x})$ gives

$$\dot{V}(\mathbf{x}) = \boldsymbol{\omega}^T\mathbf{J}\dot{\boldsymbol{\omega}} + \frac{\partial E_{pot}(\mathbf{q})}{\partial \mathbf{q}}\dot{\mathbf{q}}$$

and using (3.10) and (3.11)

$$\dot{V}(\mathbf{x}) = \boldsymbol{\omega}^T\boldsymbol{\tau} - \frac{1}{2}\frac{\partial E_{pot}(\mathbf{q})}{\partial \mathbf{q}}\mathbf{W}_R\boldsymbol{\omega}$$

$$= \boldsymbol{\omega}^T\boldsymbol{\tau} - \mathbf{T}(\mathbf{q})^T\boldsymbol{\omega}$$

where $\mathbf{T}(\mathbf{q}) = \frac{1}{2}\frac{\partial E_{pot}(\mathbf{q})}{\partial \mathbf{q}}\mathbf{W}_R = [T_x\ T_y\ T_z]^T$ is the torque resulting from E_{pot} (since torque describes change in work-energy).

Inserting the control law given in (3.12) gives

$$\dot{V}(\mathbf{x}) = -\boldsymbol{\omega}^T\mathbf{D}(\mathbf{x})\boldsymbol{\omega} \leq 0$$

since $\mathbf{D}(\mathbf{x}) \geq 0$. $\mathbf{D}(\mathbf{x})$ is referred to as a damping matrix. Its purpose is to ensure that, when close to the equilibrium points, the system converges to those instead of oscillating about them.

3.3 Path Planning

The trajectory planning algorithm implemented in this thesis is called Collective Circumnavigation. It is designed so that if there are several quadrotors, they should collectively enclose a certain target and start rotating around it uniformly distributed on a circle with a certain radius d.

The position of the quadrotors are given in the global world frame W. Assume the target is on the ground ($z = 0$) and that the quadrotors fly at a certain fixed height. The trajectory planner does not consider the quadrotors height, this can instead be decided by the position controller. Therefore, only the x- and y-coordinates are of interest for the trajectory planner.

3.3.1 Case of one quadrotor

Let $\mathbf{x}_{target}(t) \in \mathbb{R}^2$ be the targets position, let $\mathbf{x}(t) \in \mathbb{R}^2$ be the position of the quadrotor in the (x, y)-plane, and let $C(\mathbf{x}_{target}, d)$, $d \in \mathbb{R}$ be the circle with radius d and center located at $\mathbf{x}_{target}(t)$. It is assumed that the quadrotors motion is described by

$$\dot{\mathbf{x}}(t) = \mathbf{v}(t)$$

where $\mathbf{v}(t)$ is the (virtual) control input. The control law that ensures that the quadrotor approaches the target and starts rotating around it on a distance d is given by:

$$\mathbf{v}(t) = (D(t) - d)\phi(t) + \alpha\bar{\phi}(t) \tag{3.13}$$

where the distance from quadrotor to target is

$$D(t) = \| \mathbf{x}_{target}(t) - \mathbf{x}(t) \|$$

the bearing to the target is

$$\phi(t) = \frac{\mathbf{x}_{target}(t) - \mathbf{x}(t)}{\| \mathbf{x}_{target}(t) - \mathbf{x}(t) \|}$$

the counter-clockwise rotation direction is

$$\bar{\phi}(t) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \phi(t)$$

and α is the angular speed with which the quadrotor rotates over the target. One can see that if the $D(t)$ is large, then the first term in (3.13) will dominate and the quadrotor will approach the target on an almost straight line. When $D(t) \approx d$ the first term will be very small and the quadrotor will start rotating around the target at a distance d.

The desired trajectory is then given by

$$\mathbf{x}_d(t) = \mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{v}(\tau)\, \mathrm{d}\tau \tag{3.14}$$

where $t > t_0$ and $\mathbf{x}_d(t)$ represents the desired trajectory.

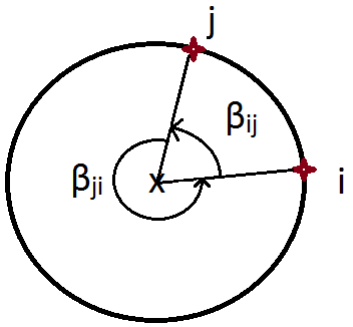

Figure 3.5: Two quadrotors circulating around a target marked with an "×". The counter clockwise angle β for quadrotor i and j is shown.

3.3.2 Case of several quadrotors

In the case of several quadrotors the control law is divided into two parts. In the first part the main task is to approach the target, but when the quadrotors are close to the target they also have to distribute themselves uniformly on the circle $C(\mathbf{x}_{target}, d)$.

Say that there are n quadrotors and that i:th quadrotor is within a specified range from the target $\parallel D_i(t) - d \parallel \leq \delta, \quad i = 1, 2, ..., n$. Then quadrotor i starts measuring which quadrotor j is the closest counter clockwise neighbour and the angle between them, the counter clockwise angle β_{ij}, as in figure 3.5. The second stage control law is given by

$$\mathbf{v}_i(t) = (D_i(t) - d)\phi_i(t) + (\alpha + \beta_{ij})\bar{\phi}_i(t) \tag{3.15}$$

where $D_i(t)$, $\phi_i(t)$ and $\bar{\phi}_i(t)$ are the same as in (3.13) but with $\mathbf{x} = \mathbf{x}_i$ and $\mathbf{x}_i$ being the position of the i:th quadrotor. The difference is an adaptable angular velocity that depends on the counter clockwise angle between quadrotor i and quadrotor j. A large counter clockwise angle gives a higher angular speed. When all counter clockwise angles are equal, the quadrotors rotate with the same angular velocity. As in the case of one quadrotor, the desired trajectory is given by

$$\mathbf{x}_{di}(t) = \mathbf{x}_i(t_0) + \int_{t_0}^{t} \mathbf{v}_i(\tau)\, \mathrm{d}\tau$$

3.4 Position Control

The position controller is what links the attitude controller with the path planner. It has the aim of calculating how much the quadrotor needs to tilt to get to a certain desired position $\mathbf{x}_d(t)$. The desired position is received by the path planner, and the desired tilt is sent to the attitude controller.

The position controller implemented in this project is presented in [10]. Since all translational motions are acquired by tilting the quadrotor, a position controllers task is basically to determine the thrust direction of the quadrotor. Recall that the thrust direction is the sum of the thrusts at the center of mass of the quadrotor caused by the four rotors, hence it will always be pointing on

the positive z-axis of the body frame B. This body frame shares the same origin as the inertial frame I (attached to the center of mass of the quadrotor) but also rotates together with the quadrotor. The inertial frame I is however always fixed and its (x,y)-plane is always parallel to the (x,y)-plane of the world frame W.

3.4.1 Control aim

If z_B is tilted with respect to z_I, then this corresponds to a rotation quaternion that describes the rotation between I and B. The aim is to find the rotation quaternion describing the rotation between the inertial frame I and a desired body frame D.

$$\mathbf{q}_d : \ I \overset{q_d}{\curvearrowright} D$$

This rotation quaternion can then be regarded as the reference quaternion that the attitude controller described in section 3.2 should track.

The control objective is to have the quadrotor to follow a certain given desired trajectory, $\mathbf{x}_d(t)$. The translational dynamics[6] in frame I are

$$\dot{\mathbf{x}}(t) = \mathbf{v}(t) \tag{3.16}$$

$$\dot{\mathbf{v}}(t) = \mathbf{g} + \frac{\mathbf{T}}{m} \tag{3.17}$$

where $\mathbf{g} = [0 \quad 0 \quad -g]^T$ and $\mathbf{T}$ is the total thrust vector (given in frame I). The control objective can thus be re-formulated to find a desired thrust direction that drives the position error $\mathbf{e}_x(t) = \mathbf{x}(t) - \mathbf{x}_d(t)$ to zero in finite time.

3.4.2 Control law

Next the control design will be presented (see [10]). The Lyapunov function

$$V_x = \frac{1}{2}\mathbf{e}_x^T\mathbf{e}_x$$

is chosen to stabilize $\mathbf{e}_x = 0$, i.e. $V_x(\mathbf{e}_x) > 0$ if $\mathbf{e}_x \neq 0$ and $V_x(0) = 0$. In order to make $\mathbf{e}_x = 0$ a stable equilibrium point V_x also needs to obey $\dot{V}_x < 0$ for $\mathbf{e}_x \neq 0$.

Analyse $\dot{V}_x$:

$$\begin{aligned} \dot{V}_x &= \mathbf{e}_x^T\dot{\mathbf{e}}_x \\ &= \mathbf{e}_x^T(\dot{\mathbf{x}} - \dot{\mathbf{x}}_d) \\ &= \mathbf{e}_x^T(\mathbf{v} - \dot{\mathbf{x}}_d) \end{aligned} \tag{3.18}$$

where $\mathbf{v}$ is the current velocity of the quadrotor and $\dot{\mathbf{x}}_d$ is the first time derivative of the desired trajectory.

Define the desired velocity

$$\mathbf{v}_D = \dot{\mathbf{x}}_d + \mathbf{A}_x\mathbf{e}_x \tag{3.19}$$

[6]The translation model is intuitive when thinking that acceleration is equal to force/mass and that the acting forces are gravity and the total thrust.

where $\mathbf{A}_x < 0$. If $\mathbf{v} = \mathbf{v}_D$, i.e. the current velocity of the quadrotor equals the desired velocity, then $\dot{V}_x$ becomes

$$\dot{V}_x = \mathbf{e}_x^T \mathbf{A}_x \mathbf{e}_x < 0 \qquad \text{for} \quad \mathbf{e}_x \neq 0$$

and $\mathbf{e}_x = 0$ will be a stable equilibrium point. It is therefore desired that $\mathbf{v} \to \mathbf{v}_D$.

Define the velocity error $\mathbf{e}_v = \mathbf{v} - \mathbf{v}_D$ which gives

$$\mathbf{v} = \mathbf{e}_v + \mathbf{v}_D$$

and using (3.19)

$$\mathbf{v} = \mathbf{e}_v + \dot{\mathbf{x}}_d + \mathbf{A}_x \mathbf{e}_x \tag{3.20}$$

Inserting (3.20) in (3.18) gives

$$\dot{V}_x = \mathbf{e}_x^T (\mathbf{e}_v + \dot{\mathbf{x}}_d + \mathbf{A}_x \mathbf{e}_x - \dot{\mathbf{x}}_d)$$

$$\dot{V}_x = \mathbf{e}_x^T \mathbf{A}_x \mathbf{e}_x + \mathbf{e}_x^T \mathbf{e}_v \tag{3.21}$$

Choose the Lyapunov function

$$V_v = V_x + \frac{1}{2} \mathbf{e}_v^T \mathbf{e}_v$$

to stabilize $\mathbf{e}_v = 0$. The time derivative of V_v is then given by

$$\dot{V}_v = \dot{V}_x + \mathbf{e}_v^T \dot{\mathbf{e}}_v$$

with $\dot{V}_x$ given by (3.21) and $\dot{\mathbf{e}}_v = \dot{\mathbf{v}} - \dot{\mathbf{v}}_D$

$$\dot{V}_v = \mathbf{e}_x^T \mathbf{A}_x \mathbf{e}_x + \mathbf{e}_v^T (\mathbf{e}_x + \dot{\mathbf{v}} - \dot{\mathbf{v}}_D) \tag{3.22}$$

This expression contains the acceleration vector $\dot{\mathbf{v}}$ of the quadrotor which can be controlled by the thrust vector $\mathbf{T}$. One can therefore define a desired thrust vector

$$\mathbf{T}_D = m(-\mathbf{g} + \dot{\mathbf{v}}_D - \mathbf{e}_x + \mathbf{A}_v \mathbf{e}_v) \tag{3.23}$$

, where $\mathbf{A}_v < 0$. And so (3.22) becomes

$$\dot{V}_v = \mathbf{e}_x^T \mathbf{A}_x \mathbf{e}_x + \mathbf{e}_v^T \mathbf{A}_v \mathbf{e}_v < 0 \qquad \text{for} \quad \mathbf{e}_v \neq 0$$

Using (3.19) $\dot{\mathbf{v}}_D$ can be defined

$$\dot{\mathbf{v}}_D = \ddot{\mathbf{x}}_d + \mathbf{A}_x (\mathbf{v} - \dot{\mathbf{x}}_d) \tag{3.24}$$

where $\ddot{\mathbf{x}}_d$ is the second time derivative of the desired trajectory.

3.4.3 Connecting with the path planner

It is seen in (3.24) that the first and second time derivative of the desired trajectory are needed to calculate the desired thrust. Since $\dot{\mathbf{x}}_d(t)$ is already defined in (3.13), it remains to analyse $\ddot{\mathbf{x}}_d(t)$

$$\ddot{\mathbf{x}}_d = \dot{D}(t)\phi(t) + (D(t) - d)\dot{\phi}(t) + \alpha\dot{\bar{\phi}}$$

in the case of one quadrotor, and

$$\ddot{\mathbf{x}}_{di} = \dot{D}_i(t)\phi_i(t) + (D_i(t) - d)\dot{\phi}_i(t) + (\alpha + \beta_{ij})\dot{\bar{\phi}}_i$$

in the case of several quadrotors and where

$$\dot{D}(t) = -\frac{(\mathbf{x}_{target}(t) - \mathbf{x}(t))^T}{\sqrt{(\mathbf{x}_{target}(t) - \mathbf{x}(t))^2}}\dot{\mathbf{x}}_d$$

$$\dot{\phi}(t) = -\frac{\dot{\mathbf{x}}_d(t)D(t) - (\mathbf{x}_{target} - \mathbf{x}(t))\dot{D}(t)}{D(t)^2}$$

and where $\dot{D}(t)$ and $\dot{\phi}(t)$ is calculated for each quadrotor in the case of several quadrotors.

3.4.4 Connecting with the attitude controller

In order to connect the position controller with the attitude controller presented in the previous section, the next step is to find the quaternion that describes the rotation between the inertial frame I and the *desired* body frame D.

Equation (3.23) describes the desired thrust vector in frame I

$$\mathbf{T}_D^I = \begin{bmatrix} T_x^I \\ T_y^I \\ T_z^I \end{bmatrix}$$

Recalling that the thrust vector always points along the positive z-axis of the body frame B one can express the desired thrust $\mathbf{T}_D^I$ in a desired body frame D as

$$\mathbf{T}_D^D = \begin{bmatrix} 0 \\ 0 \\ \parallel \mathbf{T}_D^I \parallel \end{bmatrix}$$

Since $\mathbf{T}_D^D$ and $\mathbf{T}_D^I$ describe the same vector but in different frames they can be related to each other as in (3.6) by

$$\mathbf{T}_D^I = \mathbf{R}(\mathbf{q}_d)\mathbf{T}_D^D \tag{3.25}$$

where $\mathbf{q}_d$ describes the rotation from frame I to frame D. The rotation quaternion will be on the form

$$\mathbf{q}_d = \begin{bmatrix} q_x \\ q_y \\ 0 \\ q_w \end{bmatrix}$$

The third component being equal to zero corresponds to no rotation about the z-axis. With the only purpose of aligning the z-axis of frame I with frame D, a rotation about the z-axis will not affect the result.

Plugging in $\mathbf{q}_d$ in the general expression for $\mathbf{R}(\mathbf{q})$ described in (3.7) gives

$$\mathbf{R}(\mathbf{q}_d) = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 & 2q_xq_y & 2q_yq_w \\ 2q_xq_y & q_w^2 - q_x^2 + q_y^2 & -2q_xq_w \\ -2q_yq_w & 2q_xq_w & q_w^2 - q_x^2 - q_y^2 \end{bmatrix} \tag{3.26}$$

Using (3.26) to evaluate the right hand side of (3.25) gives

$$\begin{aligned}\mathbf{T}_D^I &= \mathbf{R}(\mathbf{q}_d)\begin{bmatrix} 0 \\ 0 \\ \parallel \mathbf{T}_D^I \parallel \end{bmatrix} \\ &= \begin{bmatrix} 2q_yq_w \\ -2q_xq_w \\ q_w^2 - q_x^2 - q_y^2 \end{bmatrix} \parallel \mathbf{T}_D^I \parallel\end{aligned}$$

or

$$\begin{bmatrix} 2q_yq_w \\ -2q_xq_w \\ q_w^2 - q_x^2 - q_y^2 \end{bmatrix} = \frac{\mathbf{T}_D^I}{\parallel \mathbf{T}_D^I \parallel} = \begin{bmatrix} T_x^I \\ T_y^I \\ T_z^I \end{bmatrix} \frac{1}{\parallel \mathbf{T}_D^I \parallel} \tag{3.27}$$

Given that $\mathbf{q}_d$ is a rotation quaternion it must obey

$$\parallel \mathbf{q}_d \parallel = 1$$

which means

$$\begin{aligned} q_x^2 + q_y^2 + q_w^2 &= 1 \\ -q_x^2 - q_y^2 &= q_w^2 - 1 \end{aligned} \tag{3.28}$$

Inserting (3.28) in (3.27) gives

$$\begin{bmatrix} 2q_yq_w \\ -2q_xq_w \\ q_w^2 + q_w^2 - 1 \end{bmatrix} = \begin{bmatrix} 2q_yq_w \\ -2q_xq_w \\ 2q_w^2 - 1 \end{bmatrix} = \begin{bmatrix} T_x^I \\ T_y^I \\ T_z^I \end{bmatrix} \frac{1}{\parallel \mathbf{T}_D^I \parallel}$$

Solving first for the third row gives

$$\begin{aligned} 2q_w^2 - 1 &= \frac{T_z^I}{\parallel \mathbf{T}_D^I \parallel} \\ q_w &= \sqrt{\frac{\frac{T_z^I}{\parallel \mathbf{T}_D^I \parallel} + 1}{2}} \end{aligned}$$

and solving for the first and second row respectively gives

$$\begin{aligned} q_y &= \frac{T_x^I}{2q_w \parallel \mathbf{T}_D^I \parallel} \\ q_x &= -\frac{T_y^I}{2q_w \parallel \mathbf{T}_D^I \parallel} \end{aligned}$$

Hence the quaternion describing the rotation between the inertial frame I and the desired body frame D is given by

$$\mathbf{q}_d = \begin{bmatrix} -\frac{T_y^I}{2q_w \parallel \mathbf{T}_D^I \parallel} & \frac{T_x^I}{2q_w \parallel \mathbf{T}_D^I \parallel} & 0 & \sqrt{\frac{\frac{T_z^I}{\parallel \mathbf{T}_D^I \parallel} + 1}{2}} \end{bmatrix}^T$$

where $q_w = \sqrt{\frac{\frac{T_z^I}{\parallel \mathbf{T}_D^I \parallel} + 1}{2}}$. This can be regarded as the desired quaternion sent as a reference to the attitude controller.

Notice that the orientation of the thrust vector is described through rotations about the $x-$ and y-axis only. To include heading control define the quaternion

$$\mathbf{q}_z = \begin{bmatrix} 0 \\ 0 \\ \sin(\frac{\psi}{2}) \\ \cos(\frac{\psi}{2}) \end{bmatrix}$$

which describes a rotation about the axis $\mathbf{v} = [0\ 0\ 1]^T$ by the angle ψ, where ψ is the desired heading angle (yaw). The desired quaternion then reads

$$\mathbf{q}_d = \mathbf{q}_{xy} \circ \mathbf{q}_z$$

3.4.5 Height controller

As seen in figure 3, a height controller is embedded in the position controller. The magnitude of the thrust vector will vary depending on how much the quadrotor needs to tilt in order to keep the quadrotor on a constant height. This total thrust can be sent directly to the quadrotor as the input $U_1 = \parallel \mathbf{T}_D^I \parallel$.

Chapter 4

Simulation Results

Since each controller was presented in different papers, ([10], [8])[1], the simulations were performed with two aims. The first aim was to simulate each controller with obtained values for the quadrotors used at the SML and including possible modifications. The second aim was to show that these controllers could be merged without any changes in stability.

4.1 Attitude Controller

To simulate the attitude controller the mass moment of inertia had to be measured. The experimental setup was inspired by [27] and is referred to as a bifilar suspension system. Letting a body hang from two wires with the length L, and with center of mass right in between the wires, the moment of inertia can be calculated by measuring the period by which the body swings back and forth about the vertical axis going through the center of gravity of the body, see figure 4.1. Deriving from Newton's law, summing all moments acting on the center of gravity of the body, the equation describing the above mentioned swing by a small angel θ is given by

$$J\ddot{\theta} + \frac{mgb^2}{L}\theta = 0$$

where m is the mass of the body and b is the distance from where the wire is attached to the body to the center of mass of the body [28]. The period by which the body swings is constant (neglecting small friction forces) and given by

$$T = \frac{2\pi}{\omega}$$

where

$$\omega = \sqrt{\frac{mgb^2/L}{J}}$$

and so

$$J = \frac{mgT^2b^2}{4\pi^2 L}$$

[1] And the Collective Cirumnavigation controller developed by I. Shames, D. V. Dimarogonas and K. H. Johansson

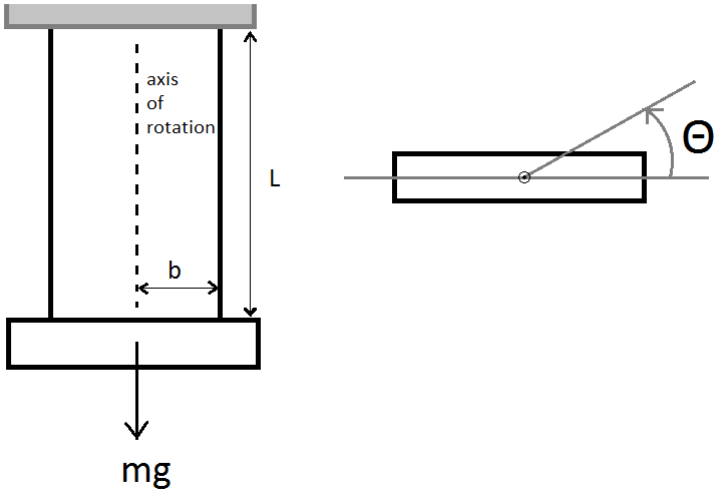

Figure 4.1: The bifilar suspension system, the left figure represents the system seen from above. A body hangs from two wires and rotates about an axis going through the center of mass of the body. As long as the angle describing the maximal deviation from initial state is small, an approximation describing the system can be obtained from where the inertia of the body can be measured.

First one can conclude that because of the symmetry of a quadrotor $J_x = J_y$, meaning that it is equally "hard" to rotate the quadrotor along its x-axis as it is along its y-axis. It is also expected from literature that $J_z \approx 2J_x$, [14] [8] [15]. Two experiments were conducted letting the vertical axis going through the center of mass of the quadrotor correspond to the x^B- and z^B-axis respectively. To calculate the J_x component, the obtained values were

$$\begin{aligned} m &= 1.129\,\text{kg} \\ b &= 0.075\,\text{m} \\ L &= 2.16\,\text{m} \\ T &= 4.5223\,\text{s} \end{aligned}$$

which gives $J_x = J_y = 14.9 \cdot 10^{-3}\,\text{m}^2\text{kg}$. To calculate the J_z component, the obtained values were

$$\begin{aligned} m &= 1.129\,\text{kg} \\ b &= 0.2175\,\text{m} \\ L &= 2.32\,\text{m} \\ T &= 2.467\,\text{s} \end{aligned}$$

which gives $J_z = 34.8 \cdot 10^{-3}\,\text{m}^2\text{kg}$. The reliability of these results is however unknown. The accuracy of the result can be improved by repeating the experiments several times or using parameter identification programs through, for example, CAD systems. Since the attitude controller in this project only was used for simulation, there was never an experimental verification of the obtained values for the mass moment of inertia of the quadrotor.

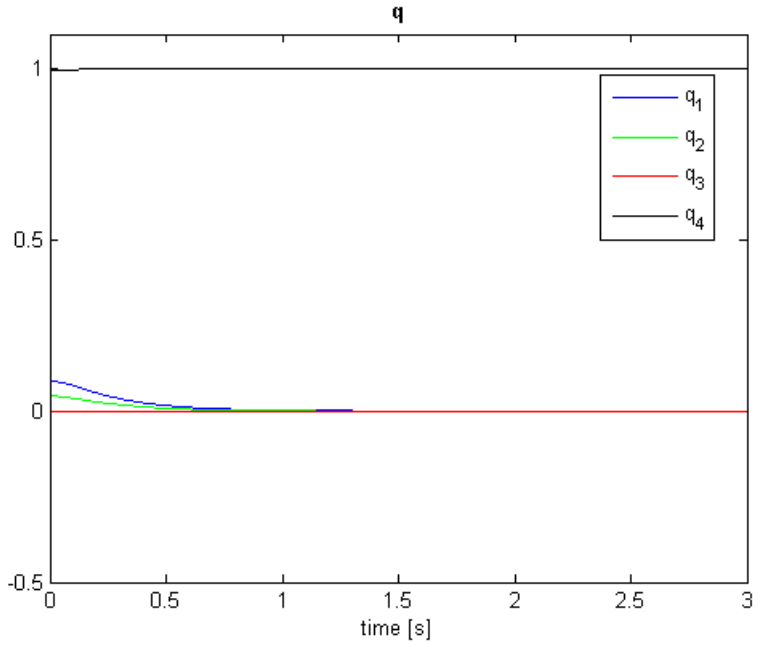

Figure 4.2: The development of the rotation quaternion $\mathbf{q}$ describing the rotation between the current body frame B and the desired frame D. The control objective is $[q_1 \ \ q_2 \ \ q_3 \ \ q_4]^T \rightarrow [0 \ \ 0 \ \ 0 \ \ \pm 1]^T$, meaning that B and D are aligned.

4.1.1 Simulation 1: Small Tilt

In the first simulation the quadrotor starts in hovering position

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

where ϕ =roll, θ =pitch, ψ =yaw and $\boldsymbol{\omega}$ =angular velocity. The desired orientation is

$$\begin{bmatrix} \phi_d \\ \theta_d \\ \psi_d \end{bmatrix} = \begin{bmatrix} 10^\circ \\ 5^\circ \\ 0^\circ \end{bmatrix}$$

Results are shown in figures 4.2, 4.3, 4.4. The main control objective is $[q_1 \ \ q_2 \ \ q_3 \ \ q_4]^T \rightarrow [0 \ \ 0 \ \ 0 \ \ \pm 1]^T$ meaning that the body frame B of the quadrotor aligns with the desired body frame D. This simulation represents a small tilt which is why B very quickly aligns with D.

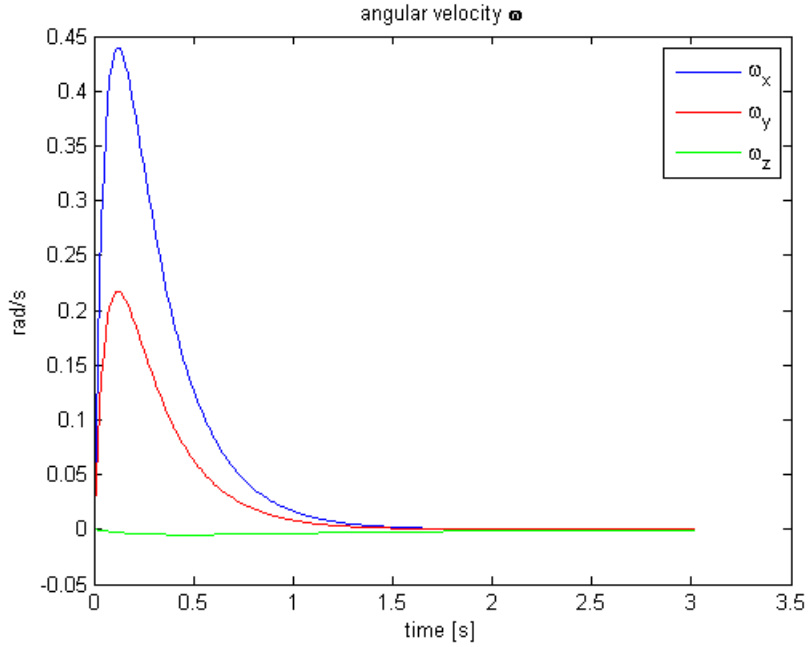

Figure 4.3: The development of the angular velocity $\boldsymbol{\omega}$ of the quadrotor. The control objective is $\boldsymbol{\omega} \to 0$.

4.1.2 Simulation 2: Large Tilt

The second experiment has the aim of letting the quadrotor track a much larger tilt. The starting position corresponds again to hovering state and the desired orientation is described by

$$\begin{bmatrix} \phi_d \\ \theta_d \\ \psi_d \end{bmatrix} = \begin{bmatrix} 50^\circ \\ 25^\circ \\ 30^\circ \end{bmatrix}$$

The results are shown in figure 4.5, 4.6 and 4.7. As seen in figure 4.5 it takes a somewhat longer time to achieve the desired state compared to the case with a small tilt, but one can also see that the controller still prioritizes the alignment of the thrust vector in figure 4.7. Recalling that ϕ represents the error angle between the current body frame B and the auxiliary frame A, and that the z^A-axis is aligned with the z^D-axis, one can see that the quadrotor aligns its thrust vector with the desired thrust vector in less than 1.5 s.

4.2 Position Controller

Simulation results, given by figure 4.8 and 4.9, of the position controller show that the quadrotor reaches the desired position and stays there. The position controller calculates the desired thrust magnitude and direction and the position of the quadrotor evolves according to 3.17. The initial position is $\mathbf{x} = [0 \ 0 \ 1]^T$.

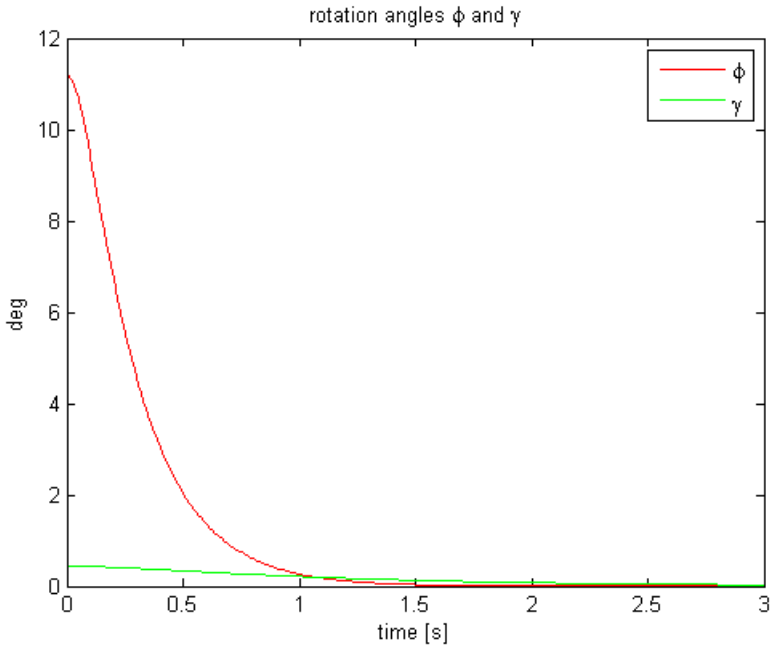

Figure 4.4: The development of the rotation angles ϕ and γ. ϕ describes the error angle between the body frame B and the auxiliary frame A, while γ describes the error angle between the auxiliary frame A and the desired body frame D. The control objective is $\phi, \gamma \to 0°$.

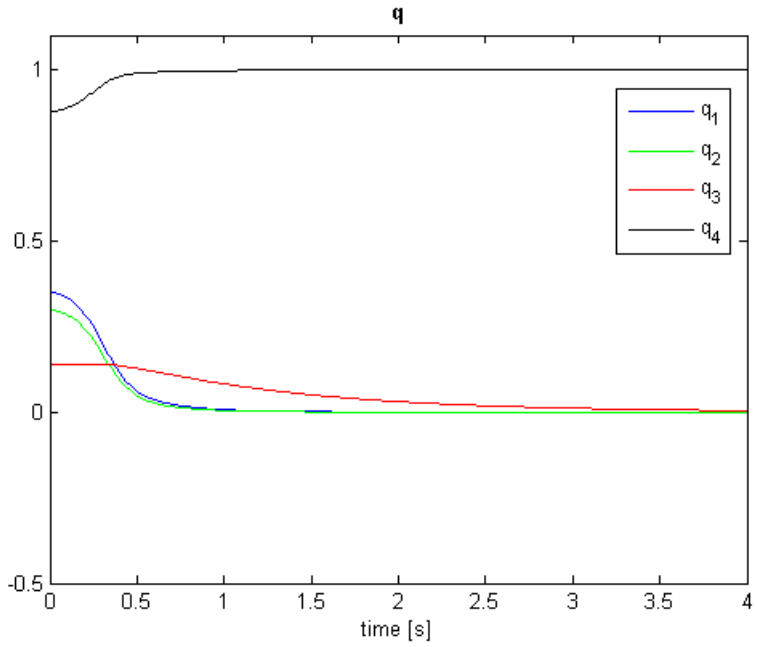

Figure 4.5: The development of the rotation quaternion $\mathbf{q}$ describing the rotation between the current body frame B and the desired frame D. The control objective is $[q_1 \;\; q_2 \;\; q_3 \;\; q_4]^T \rightarrow [0 \;\; 0 \;\; 0 \;\; \pm 1]^T$. One can see the prioritization of aligning the thrust vector by the x and y components quickly converging to zero. $q_x = q_y = 0$ namely means that the difference between the body frame and the desired body frame does not depend on rotations about the x- or y-axis.

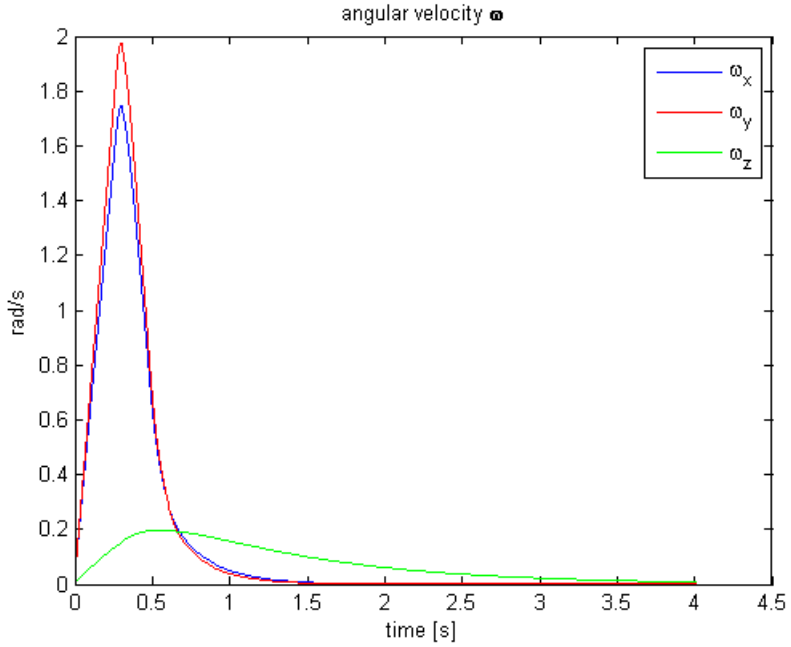

Figure 4.6: The development of the angular velocity $\boldsymbol{\omega}$ of the quadrotor. The control objective is $\boldsymbol{\omega} \to 0$.

The trajectory is described by

$$\begin{aligned}\mathbf{x}_d &= [2 \;\; 1 \;\; 1]^T \\ \dot{\mathbf{x}}_d &= [0 \;\; 0 \;\; 0]^T \\ \ddot{\mathbf{x}}_d &= [0 \;\; 0 \;\; 0]^T\end{aligned}$$

the mass m of the quadrotor was 1.126 kg. The control parameters used were

$$\begin{aligned}A_x &= -\mathbf{I}_{3\times 3} \\ A_v &= -3\,\mathbf{I}_{3\times 3}\end{aligned}$$

where $\mathbf{I}$ represents the identity matrix.

4.3 Collective Circumnavigation

The Collective Circumnavigation algorithm was simulated with 8 quadrotors starting at random position. Each quadrotor has its own id, and each quadrotor is able to determine which is the id of the closest counter clockwise neighbour and so also determine β_{ij}. By knowing the counter clockwise angle β_{ij} of each quadrotor, the group of quadrotors converges to an equilateral polygon while circulating around the target. The results is shown in figure 4.10. The parameters used were

$$\begin{aligned}d &= 3\,\text{m} \\ \alpha &= 0.3\,\text{rad/s} \\ \delta &= 3\,\text{m}\end{aligned}$$

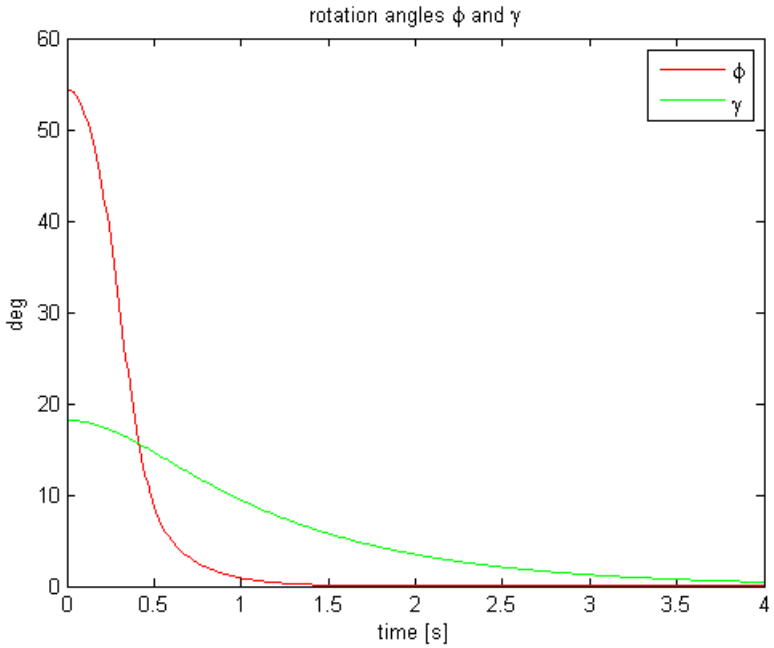

Figure 4.7: The development of the rotation angles ϕ and γ. ϕ describes the error angle between the body frame B and the auxiliary frame A, while γ describes the error angle between the auxiliary frame A and the desired body frame D. The control objective is $\phi, \gamma \to 0°$.

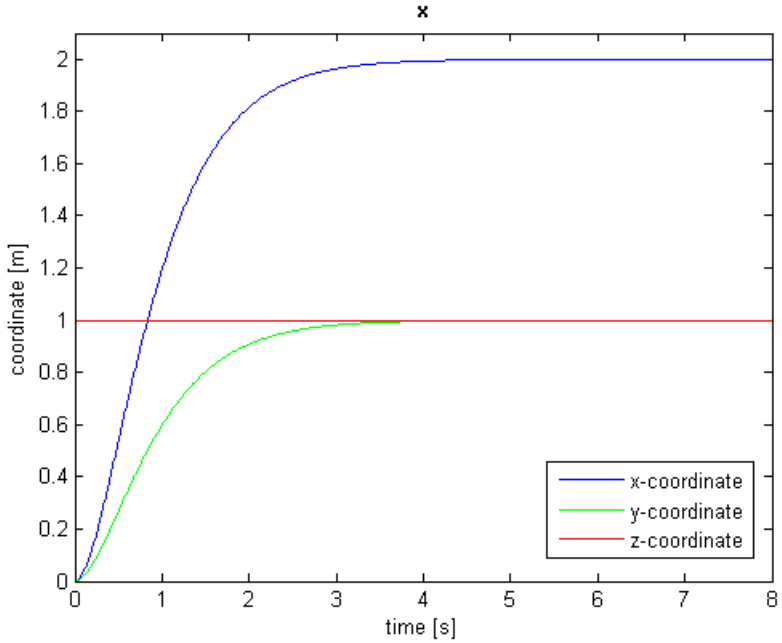

Figure 4.8: The (x, y, z)-position of the quadrotor, simulating the position controller. The desired final position is $[2\ \ 1\ \ 1]^T$, the initial is $\mathbf{x} = [0\ \ 0\ \ 1]^T$.

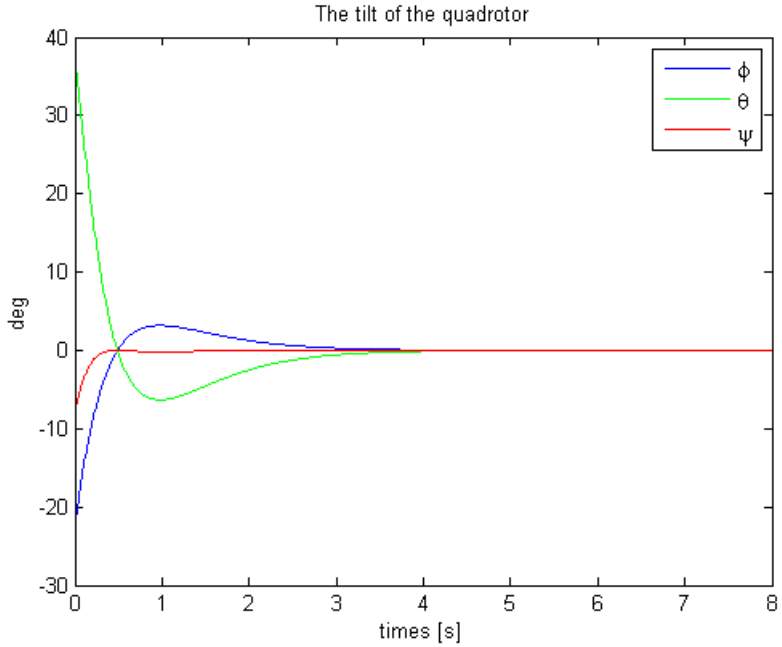

Figure 4.9: Simulating the position controller. The desired tilt, i.e. the roll, pitch and yaw angles of the quadrotor are shown in the figure.

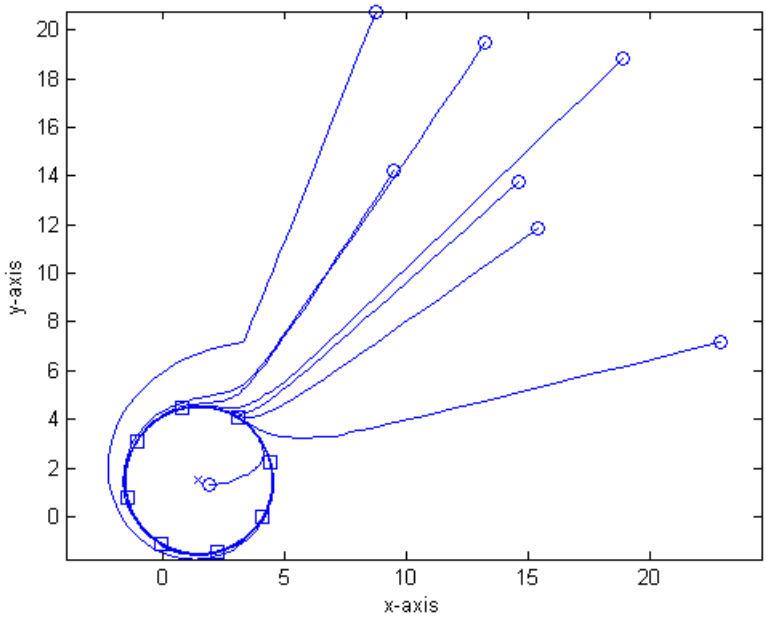

Figure 4.10: Collective Circumnavigation for a group of 8 quadrotors with random starting points marked with a "$\circ$". The target's position is marked with an "$\times$", and the quadrotors final position are marked with a "$\square$". One can see that the quadrotors distribute themselves uniformly on the circle with center at the target and radius $d = 3$ m.

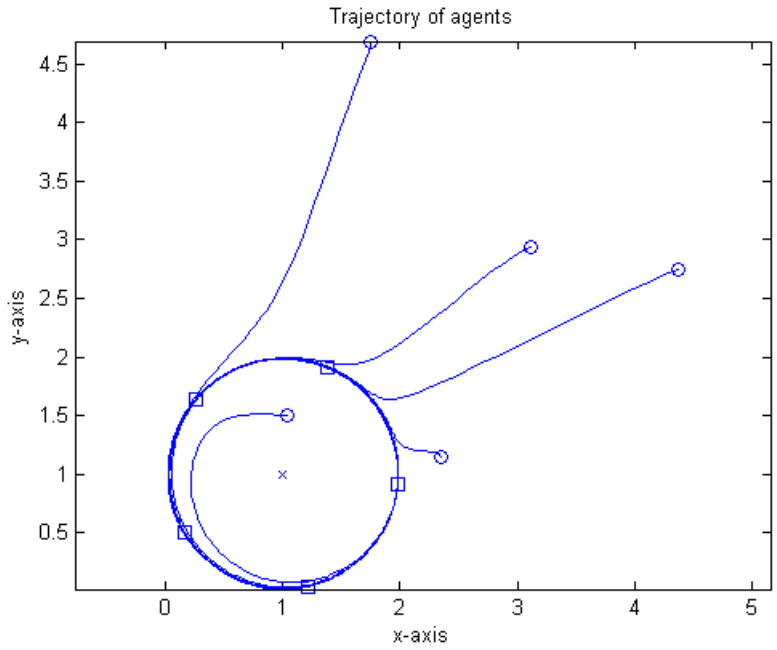

Figure 4.11: Collective Circumnavigation for a group of 5 quadrotors with random starting points marked with a "$\circ$". The target's position is marked with an "$\times$", and the quadrotors final position are marked with a "$\square$". The position of each quadrotor is calculated by the position controller.

where d is the radius of circle around target, α is the angular speed of the quadrotors and δ is the distance $\parallel D_i(t) - d \parallel$ where the second stage of the control law takes over for quadrotor i.

4.4 Merging the Controllers

The final aim of the simulations has been to see if it is possible to merge the three controllers. Both the Circumnavigation controller and the attitude controller have conceptually remained as they were designed by their inventors (only parameter values where changed). For the position controller however, the main difference between the controller implemented in this project and how it is designed in [10] is that in [10] they have a full control of the quadrotor, meaning they can send the control inputs $[U_1 \ U_2 \ U_3 \ U_4]^T$ directly to the motors while in this project the objective is to send the desired direction of the thrust vector to the attitude controller. Therefore it was sufficient to know the desired thrust vector and to calculate the desired quaternion which then could be sent as a reference to the attitude controller.

The next simulation result presents the case when the Circumnavigation algorithm calculates the desired trajectory given by $\mathbf{x}_d(t)$, $\dot{\mathbf{x}}_d(t)$ and $\ddot{\mathbf{x}}_d(t)$ and the position controller simulates the true position of each quadrotor through equation (3.17). The results are shown in figures 4.11, 4.12 and 4.13, and the

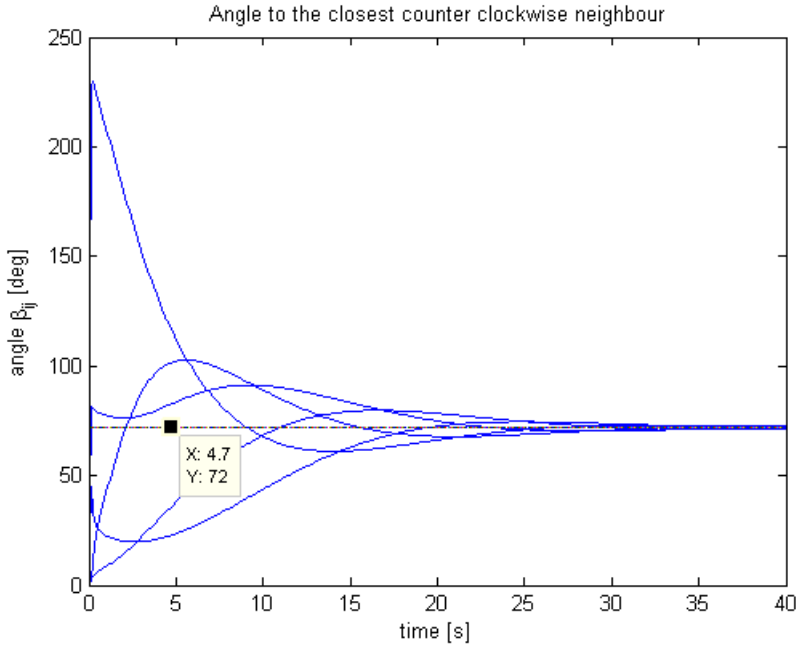

Figure 4.12: This figure shows how the counter clockwise angle β_{ij} evolves. The Circumnavigation algorithm ensures that each β_{ij} converges to $\frac{360^\circ}{n}$, where n is the number of quadrotors. In this case $\beta_{ij} \to \frac{360^\circ}{5} = 72^\circ$.

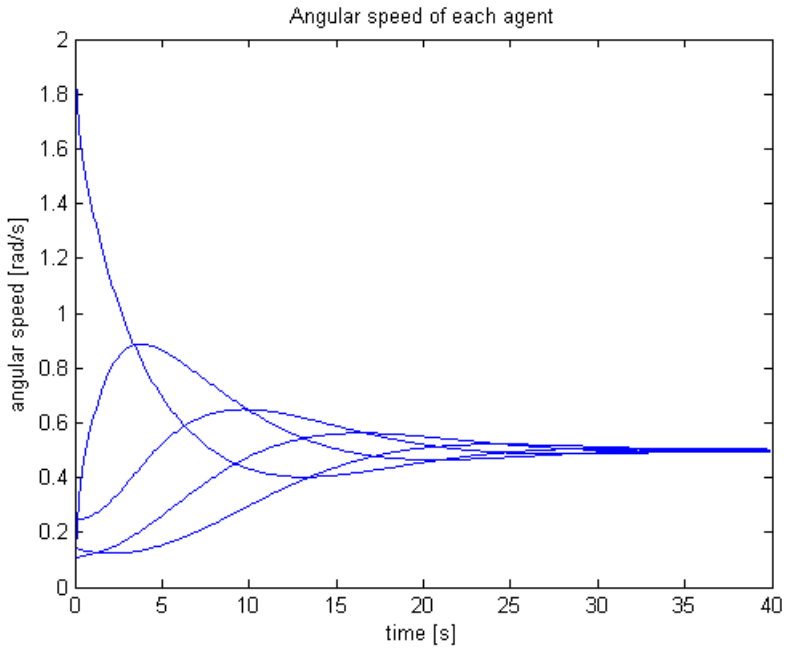

Figure 4.13: This figure shows how the angular speed of each agent evolves. By the Circumnavigation control law it increases if the counter clockwise neighbour is far ahead on the circle, and decreases if the counter clockwise neighbour is close.

parameters used were the same as in previous simulations except for

$$
\begin{aligned}
d &= 1\,\mathrm{m} \\
\alpha &= 0.5\,\mathrm{rad/s} \\
\delta &= 1\,\mathrm{m}
\end{aligned}
$$

where d is the radius of the circle around the target, α is the fixed part of the angular velocity and δ is the distance outside the circle around target where the second stage of the control law takes over.

Chapter 5

Test bed

This chapter is thought to give an overview of the equipment used at the SML. At first an overall presentation of the control process and the hardware components involved will be presented. This is followed by a more detailed description of each part.

There are three major components at the SML for controlling the quadrotors. The camera system, a controller PC and flight electronics on board of the quadrotors. The camera system is responsible for determining the 3D position of the quadrotors. This information is sent to the controller PC which runs the path planner and position controller based on the data from the camera system and sends the desired attitude wirelessly to the quadrotors. The quadrotors have a control board with an attitude controller installed. It tracks the desired attitude and forwards reference signals to the speed controllers of the motors, the system is shown in figure 5.1.

The experiments are conducted on a floor area of about 4×4 meters. Every motion within this area can be captured by cameras connected to a motion capture system.

5.1 The Quadrotors

The quadrotors used for the experiments were JDrones ArduCopter quadrocopters presented in figure 5.2.

5.1.1 Hardware

The propellers are made of plastic and attached to four electrical, brushless AC motors. The motors are quite powerful and rotate with a maximum angular speed of 11000 RPM. They are power by one LiPo 3-cells battery with nominal voltage 11.1 V and capacity of 2200 mAh. A flight with a fully charged battery lasts up to 10 minutes. Each motor is controlled by a speed controller which has the task of adjusting the AC current to the motors so that they spin with desired velocity (proportional to the input signal) [12] [29].

The quadrotor is also equipped with flight electronics. There are four boards on the quadrotors, two for receiving and forwarding wireless information and two for the control (attitude control) of the quadrotor.

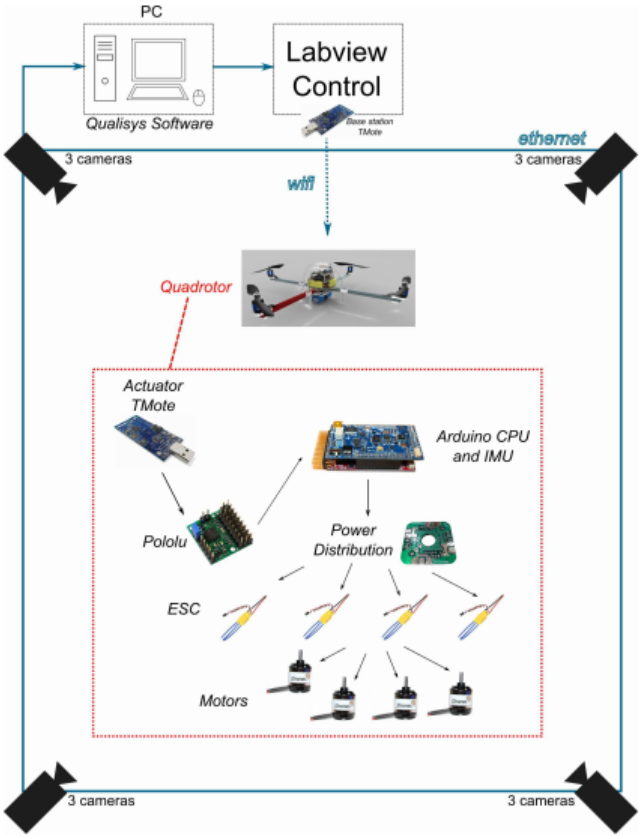

Figure 5.1: The set up for the test bed at the SML. Cameras read the position of the quadrotor and send the information to a Controller PC. The Controller PC runs the path planner and the position controller and send the reference signals for the attitude controller wirelessly through *Tmotes*, several boards on board of the quadrotor are then used for the attitude controller which sends reference signals to the speed controllers (ESC) of the motors. Source: *Quadrocopter Getting Started Manual* [30]

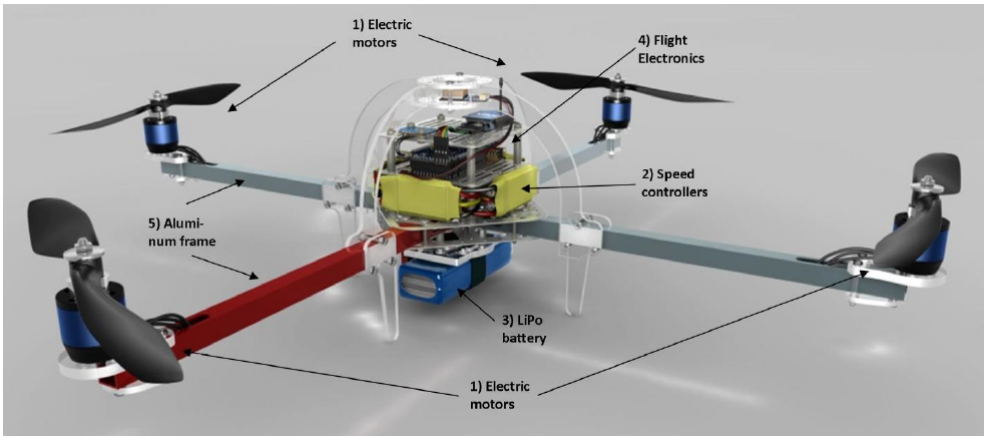

Figure 5.2: A JDrones ArduCopter quadrocopter. Source: *Smart mobility manual* [29]

The sensor board. One board is equipped with a number of sensors used to measure the state of the quadrotor. The board is the blue IMU-board in figure 5.1. Sensor readings are used by the control board.
The control board The control board is responsible for tracking the desired attitude. It receives reference values and sensor readings and sends control signals to the motor speed controllers.
The two remaining boards will be explained in section 5.1.3.

5.1.2 Software

As mentioned, the attitude control takes place on the control board. The firmware is installed through the programme *Mission Planner*[1] through which one also can do different calibrations, e.g. accelerometer calibration.

The attitude controller consists of two PID controllers. A representative scheme can be seen in figure 5.3. Important observations are what type of input signals the attitude controller accepts. One can see that for roll and pitch, the reference signal can vary from $-45°$ to $+45°$, but for yaw it is $-45°/s$ to $45°/s$. This means that the position controller should send the desired attitude described by $[\phi \ \theta \ \dot{\psi}]^T$.

5.1.3 Wireless Communication

Pololu Micro Serial Servo Controller Board. The control board receives the control signals (reference signals for the attitude controller) from the Polulu board. The Polulu board accepts logic level serial signals and uses Pulse Width Modulation (PWM) to convert the input signals to the reference signals sent to the control board. This is needed since the firmware on the control board is designed so that the control board accepts PWM signals for each channel (corresponding to roll, pitch yaw, etc).
TMote Sky. For wireless communication the SML uses TMote Sky devices

[1] Available at `https://code.google.com/p/ardupilot-mega/downloads/list`

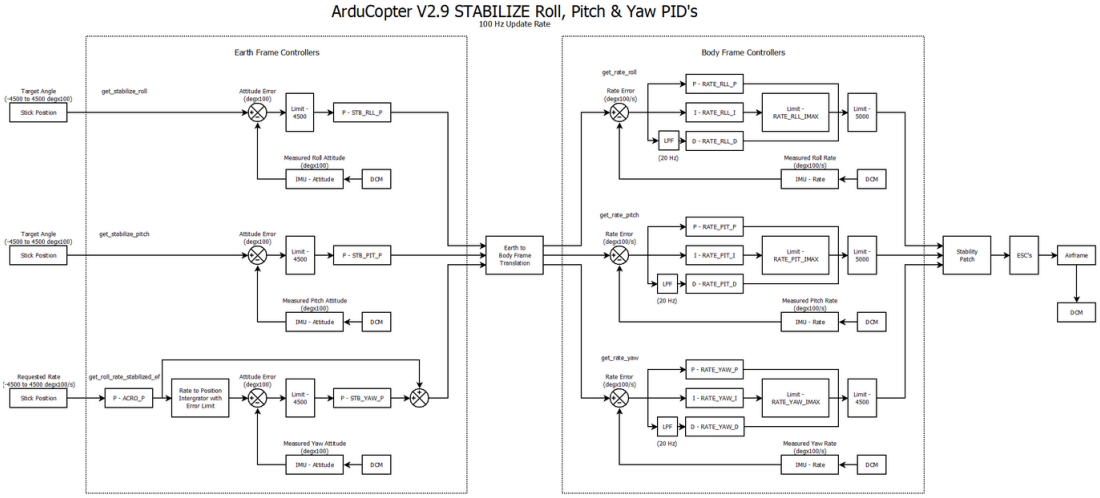

Figure 5.3: A scheme for the attitude controller installed on the control board through the Mission Planner. Source: `http://diydrones.com/profiles/blogs/arducopter-2-9-pid-loops-for-stabilize-acro-and-alt-hold`

which are wireless sensor nodes (or motes). The Tmote Sky devices use the IEEE 802.15.4 protocol and consist of a pair of motes, one for sending data and one for receiving. The sending mote is connected to the controller PC through a USB port, and the receiving mote is connected to a Serial Adapter board on the quadrotor. The motes have the capacity of sending 8 signals simultaneously, where each signal can represent $2^7 = 128$ values.

Serial Adapter Board. The Serial Adapter board is costume made and has the task of receiving serial signals from the receiver mote and forwarding them to the Polulu board while converting them to the logic-level serial signals that the Polulu board accepts [29] [30].

5.2 Controller PC

The controller PC runs the position controller and the path planner. The software used for implementing the controllers is Labview from National Instruments. From the Motion Capture system the computer receives information about the position of each quadrotor. Using this information the path planner calculates the next desired position and the position controller calculates the reference values for the attitude controller. These are continuous signals which are quantized to the 128 possible values and sent as serial signals from the sending mote on the PC to the receiving mote on the quadrotor. The Labview program runs with the frequency $f = 10$ Hz.

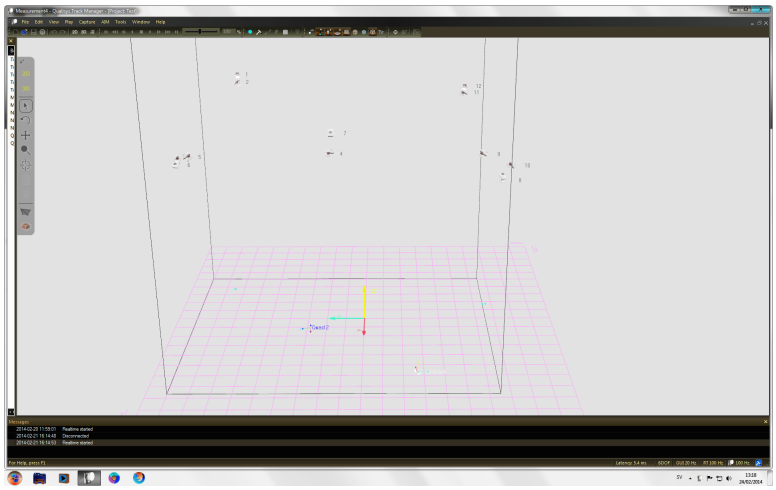

Figure 5.4: A screen shot from the QTM system showing the workspace, the world coordinate system and the body-fixed coordinate system.

5.3 The Motion Capture System

The Motion Capture system used at the SML is developed by Qualisys[2] and consist of twelve infra red (IR) cameras that together cover the floor area where the experiments are conducted. To track a body, a set of *markers* with the shape of small balls are attached the the body. The cameras then send out IR flashes and the markers reflect the light. The reflected light is then captured by the cameras which calculate the relative position of the markers. The information from the cameras is sent via ethernet to a computer running the software Qualisys Tracking Manager (QTM). Merging the information from the cameras, the QTM calculates the 3D position of the markers relative to a world coordinate system defined in the QTM, see figure 5.4 and 5.5.

Through the QTM one can configure a set of markers so that the QTM recognizes them as one specific body. In this way one can keep track of several bodies simultaneously [29][30].

[2] `www.qualisys.com`

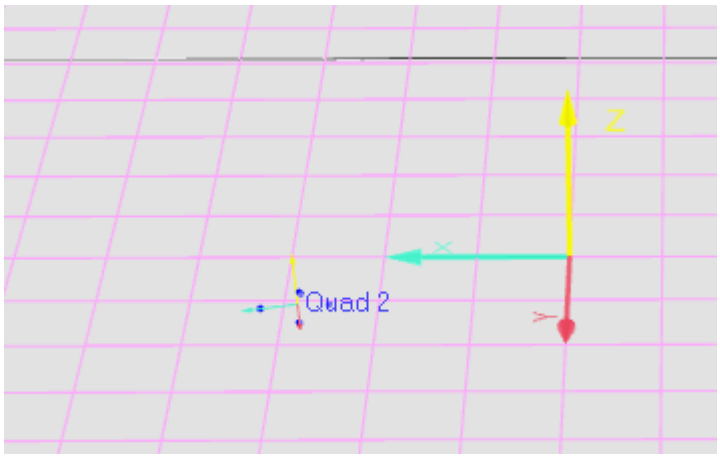

Figure 5.5: Zooming in on the quadrotor from figure 5.4. The small balls are the markers that reflect IR light. It is through configuration of the markers that the QTM system recognizes each body.

Chapter 6

Implementation

In this section the result from two test flights will be presented. The final aim of the experiments has been to prove the feasibility of Circumnavigation rather than focusing on the performance. There are some parts of the control law that were left out and some parts that can be improved in order to optimize the performance. These will be discussed more in detail in chapter 7.

6.1 Path planner: Collective Circumnavigation

The Collective Circumnavigation algorithm was implemented in the Controller PC of the SML using Labview. The first test flight was performed with one quadrotor and a start point far away from the target. The result is presented i figure 6.1 and in `http://youtu.be/bnjhdICYvSU`. The second test flight was performed with two quadrotors. Their start point was relatively close to the target but also to each other. The aim of this test was to show that when $D \approx d$ and β_{12} is very different from β_{21} (from equation (3.15)), then one quadrotor decreases its angular velocity while the other one increases it until $\beta_{12} \approx \beta_{21}$. The results are presented in figure 6.2 and 6.3, and in `http://youtu.be/w4WllxJh-Bg`.

The parameters used for the first test flight were:

- radius of circle around target, $d = 1$ m
- angular velocity, $\alpha = 0.5$ rad/s.

At the beginning the quadrotor clearly prioritizes to approach the target. When $D \approx d$ the quadrotor slows down and starts rotating around the target.

The parameters used for the second flight test were:

- radius of circle around target $d = 1.5$ m
- angular velocity $\frac{(\alpha+\beta_{ij})}{\pi/2} = \frac{(0.2+\beta_{ij})}{\pi/2}$ rad/s.

The angular velocity was in this case modified to $\frac{(\alpha+\beta_{ij})}{\pi/2}$ in order to have a smooth transition between the cases when β_{12} is very different from β_{21}, and when $\beta_{12} \approx \beta_{21}$, otherwise a quadrotor could have a very high angular velocity if its counter clockwise angle β is large which is not desirable since one wants the quadrotor to fly near hovering state.

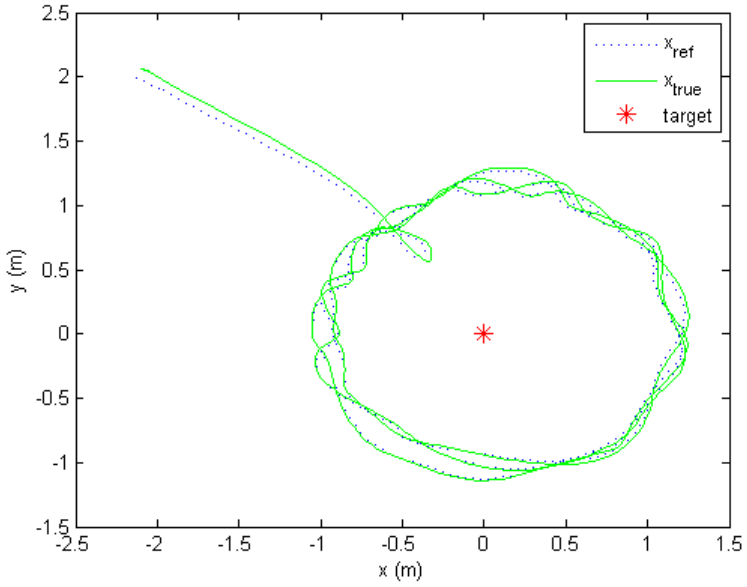

Figure 6.1: In the first test flight the quadrotor starts far away from the target, approaches it and starts circulating around it.

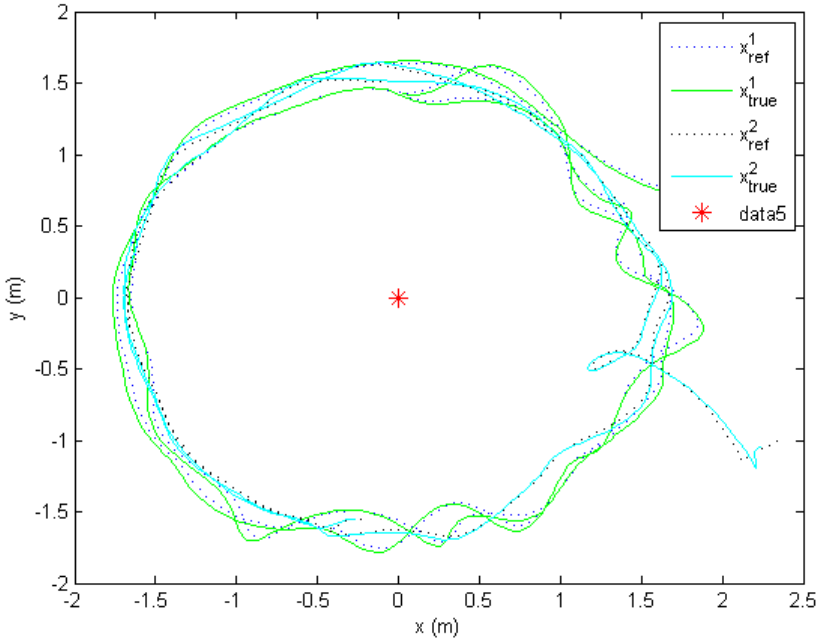

Figure 6.2: In the second test flight the quadrotors start close to each other, approaches the target and start circulating around it.

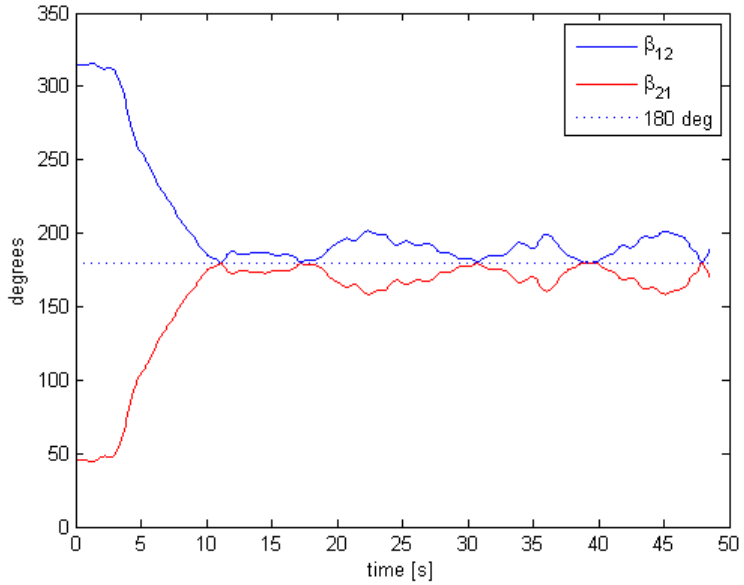

Figure 6.3: The quadrotors regulate their angular speed so that $\beta_{12} \approx \beta_{21}$.

6.2 Position Controller

The position controller was also implemented in Labview on the Controller PC. The performance was good with results from the first test flight shown in figure 6.4 and 6.5. The RMS error is given by

$$e_{RMS} = \sqrt{\frac{1}{T}\sum_{t=t_i}^{t_f}(x_d(t) - x_{true}(t))^2}$$

where T is the total duration of the experiment, t_i is the initial time and t_f is the final time. Measuring for the case when the quadrotor is flying on the circle (no abrupt changes in path) the RMS error was 0.023 m for both $e_x = x_d - x_{true}$ and $e_y = y_d - y_{true}$. In other words, the mean deviation from the reference point was 2.3 cm in x-position and y-position. Including the phase where the quadrotor approaches the target (which contains abrupt changes in path) the RMS errors are $e_{xRMS} = 0.029$ m and $e_{yRMS} = 0.034$ m.

6.2.1 Limitations

There were some differences in the implementation of the position controller comparing to the controller used for simulations. To start with, in theory the output of the position controller is a quaternion which is sent as the reference quaternion to the attitude controller. The attitude controller used for the experiments was however the pre-installed attitude controller loaded through the firmware to the control board. This attitude controller tracks roll and pitch angles and the rate of change of the yaw angle. Therefore a function to convert

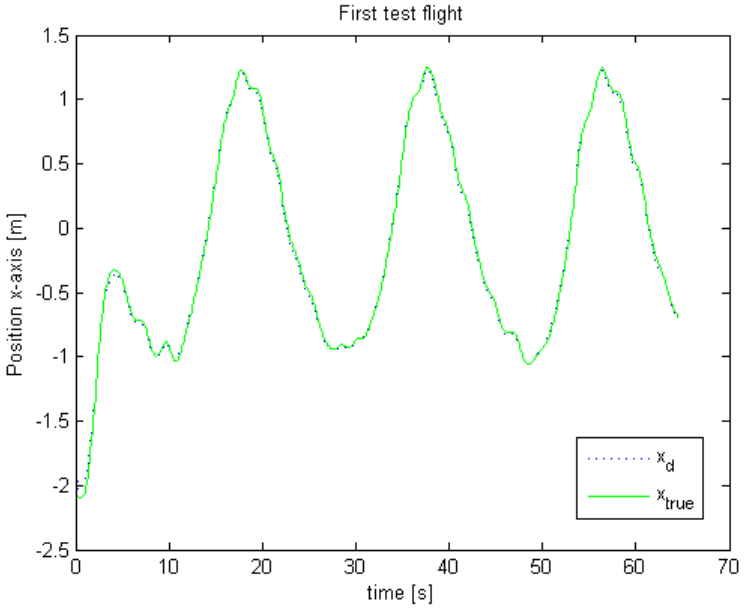

Figure 6.4: Reference tracking in x-position for the first test flight, x_d is the reference and x_{true} is the position of the quadrotor.

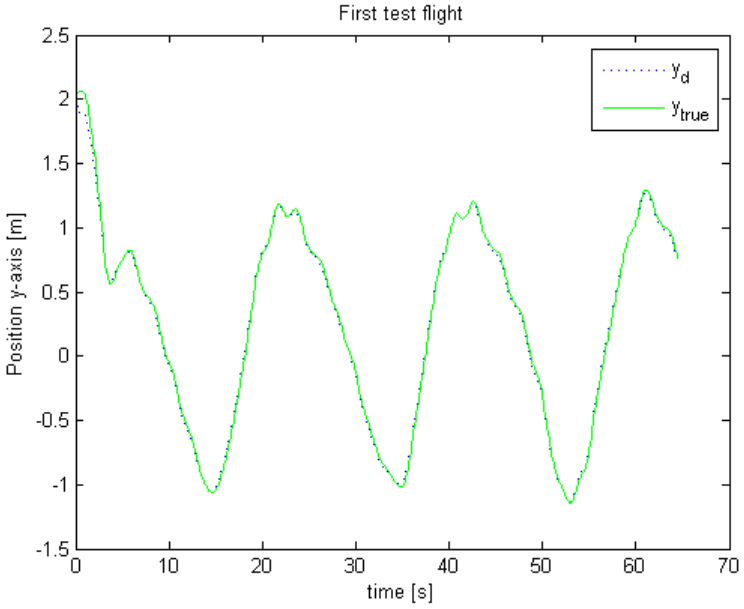

Figure 6.5: Reference tracking in y-position for the first test flight, y_d is the reference and y_{true} is the position of the quadrotor.

quaternions to Euler angles had to be implemented. When converting quaternions back to Euler angles one can use different rotation sequences (the rotation matrices look different according to which rotation sequence one chooses to use, but they all give the same result). The most common is zyx which means "first a rotation about the z-axis, then a rotation about the new y-axis and last a rotation about the newer x-axis", but there are twelve different rotation sequences for Euler angles. Using the zyx-sequence one can find the rotation angles (ψ, θ, ϕ) by

$$\begin{aligned}\tan\psi &= \tfrac{m_{12}}{m_{11}}\\ \sin\theta &= -m_{13}\\ \tan\phi &= \tfrac{m_{23}}{m_{33}}\end{aligned}$$

where

$$\begin{aligned}m_{11} &= 2q_w^2 + 2q_x^2 - 1\\ m_{12} &= 2q_xq_y + 2q_wq_z\\ m_{13} &= 2q_xq_z - 2q_wq_y\\ m_{23} &= 2q_yq_z + 2q_wq_x\\ m_{33} &= 2q_w^2 + 2q_z^2 - 1\end{aligned}$$

for a quaternion

$$\mathbf{q} = \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix}$$

see [20] for more details.

The obtained roll and pitch angles are sent to the attitude controller. However, since the attitude controller only accepts the rate of change of the yaw angle, the obtained yaw angle from the position controller was not used, instead the yaw controller (as part of the attitude controller) was set to track 0°/s which means that the yaw angle remained fixed.

Another aspect to think of is the quantization. The motes can send 8 signals which can take integer values between 0 and 127. This means for roll and pitch that 0 represents $-45°$, 64 represents 0° and 127 represents 45°. Similarly, for yaw 0, 64 and 127 represent $-45°$/s, 0°/s and 45°/s respectively. Another limitation has been that the attitude controller works best for small roll and pitch reference angles. This problem became clear when for example letting the quadrotor fly from point $(x, y) = (0, 0)$ to $(x, y) = (1, 0)$. The position controller then calculated a fairly large pitch angle that was sent to the attitude controller, but when approaching the set point the quadrotor instead of stabilizing started oscillating around set point with a pitch reference angle never converging to zero. This was not much further investigated but instead the reference angles were saturated to some certain small ϕ_{min}, ϕ_{max} and θ_{min}, θ_{max}. In the case of Circumnavigation however, this has been no problem since the path planner calculates the next desired position every iteration, so the distance between $\mathbf{x}_d(k-1)$ and $\mathbf{x}_d(k)$ is small since the control loops executes every 100 ms. This leads to the reference roll and pitch angles always being small.

The parameters used to for the position controller were

- $A_x = -\mathbf{I}_{3\times3}$
- $A_v = -3 \cdot \mathbf{I}_{3\times3}$

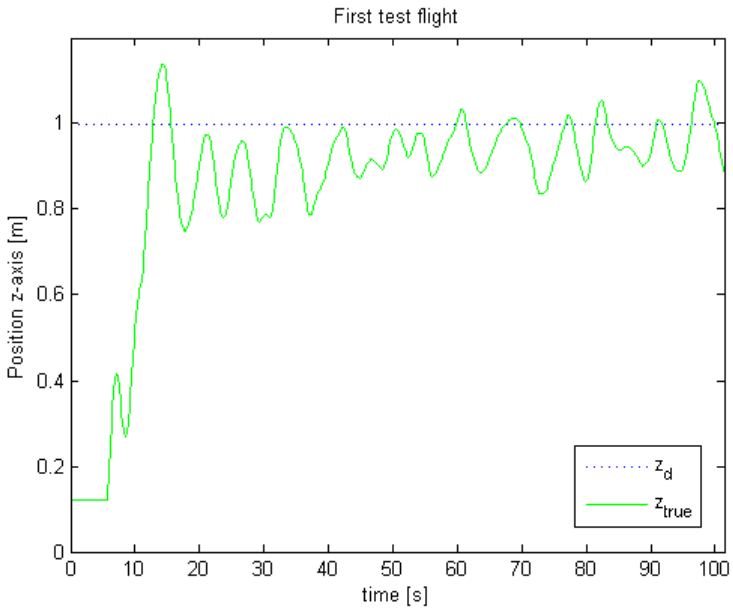

Figure 6.6: Reference tracking in z-position for the first test flight, z_d is the reference and z_{true} is the position of the quadrotor. This figure includes the take-off phase to show the step response of the height controller.

- $m_1 = 1.13$ kg for one quadrotor and $m_2 = 1.21$ kg for the second quadrotor.

where $\mathbf{I}$ is the identity matrix.

Height Control

In theory the height control is just the magnitude of the thrust vector sent as the input signal U_1 to the rotors. In reality one has to go through the control board of the quadrotor. The reference signal that is sent from the Controller PC to the quadrotor is referred to as throttle. It can be thought of as equivalent to the gas pedal in a car. Increasing the throttle increases the speed of the rotors and so also the thrust, but the relationship between the throttle and the thrust is unknown. In addition the relationship is clearly battery dependent; the quadrotor hovers with a lower throttle when using a fully charged battery than when hovering with a battery that has been used for a while. Since it is unknown which value between 0 and 127 that corresponds to the desired thrust, the height control from the position controller has not been used. Instead a PID controller designed in [12] was used to control the height. In the case of one quadrotor this controller works fine but when flying with two quadrotors the performance was poor and height control was done manually. The result from the first test flight is shown in figure 6.6. As one can see the height control is a much slower process than attitude and position control with larger momentary errors than for reference tracking in (x, y)-position.

Chapter 7

Conclusions and Discussion

This chapter contains an analysis about difficulties in implementation, what could be improved and what work there is left to do in order to implement all the controllers as they are designed in theory, chapter 3.

Regarding the merge of Circumnavigation with the position controller the main two constraints are frequency vs. rotational speed and number of UAVs vs. radius of circle around the target. In the experiments conducted the control loop iterates every 0.1 s which is a frequency at which the UAVs still converges to the desired formation. At a lower frequency one has to adjust (decrease) the rotational speed (α) in order to have optimal performance. In the case of many UAVs, the radius of the circle around the target should not be too small so that each counter clockwise angle converges to the same value, otherwise they converge to a region with an upper and lower limit (which also might be acceptable depending on the circumstances). From an implementation aspect there might be a problem conducting experiments with more than two quadrotors since the path planning and position control runs on the same computer and might be more time consuming than the maximum iteration time that still guarantees stability.

Regarding the position controller, what needs to be improved is mainly the height controller. Doing a system identification that reveals the relationship between the throttle and the thrust would be of great significance. Having an accurate height controller the position of the quadrotor in 3D space would be very precise and other projects such as formation or acrobatics could be implemented. Overall, the position controller would be more accurate if the real thrust vector corresponded to the desired one since translational movements are dependent on the x and y components of the thrust vector in the inertial frame. Further changes that would improve the performance are getting rid of the oscillating behaviour when the reference angles roll or pitch are "too large". The oscillating behaviour might disappear when implementing another attitude controller than the two PIDs installed with the firmware. If not one could add a damping term to the position controller that would bound the reference angles when the quadrotor is close to the desired position $\mathbf{x}_d$. Using the system at the SML there has always been an unidentified small offset in position both in this and in previous projects. This has probably to do with the QTM system and could be further investigated if one wants absolute accuracy in position.

The attitude controller was not implemented at all and it would be a very

interesting project to do so. Aspects to consider would be sensor readings, capacity of the micro controller and to have a good model in order to know the relationship between the motor speeds and the control torque and thrust. To merge the position controller with the attitude controller as described in this thesis would also eliminate the use of rotation matrices which are more computationally demanding.

Further research that would be interesting to consider could be to decentralize the control and let all controllers be installed on the control board of the quadrotor. In that case all measurements would either have to rely on on-board sensors, on the other hand the quadrotors would not be dependent on their location, i.e. experiments would not necessarily have to take place indoors at the SML.

Bibliography

[1] P. Bouffard, A. Aswani, C. Tomlin: *Learning-Based Model Predictive Control on a Quadrotor: Onboard Implementation and Experimental Results*, $p.279-284$, 2012 IEEE International Conference on Robotics and Automation RiverCentre, Saint Paul, Minnesota, USA.

[2] K. Alexis, G. Nikolakopoulos, A. Tzes: *Model Predictive Control Scheme for the Autonomous Flight of an Unmanned Quadrotor*, $p.2243-2248$, Industrial Electronics (ISIE), 2011.

[3] B. Szlachetko M. Lower: *Stabilisation and Steering of Quadrocopters Using Fuzzy Logic Regulators*, ICAISC 12 Proceedings of the 11th international conference on Artificial Intelligence and Soft Computing - Volume Part I, $p.691-698$, 2012.

[4] J.F. Guerrero-Castellanos, J.J. Tellez-Guzman, S. Durand, N. Marchand, J.U. Alvarez-Munoz: *Event-triggered nonlinear control for attitude stabilization of a quadrotor*, 2013 International Conference on Unmanned Aircraft Systems (ICUAS) May 28-31, $p.584-591$, Atlanta 2013.

[5] J. Wang, M. S. Geamanu, A. Cela, H. Mounier, S. I. Niculescu: *Event driven model free control of quadrotor* , 2013 IEEE International Conference on Control Applications (CCA) Part of 2013 IEEE Multi-Conference on Systems and Control, $p.722-727$, India 2013

[6] A. Honglei, L. Jie, W. Jian, W. Jianwen M. Hongxu: *Backstepping-Based Inverse Optimal Attitude Control of Quadrotor*, International Journal of Advanced Robotic Systems, Vol. 10, 2013.

[7] P. De Monte, B. Lohmann: *Position Trajectory Tracking of a Quadrotor Helicopter based on L1 Adaptive Control*, 2013 European Control Conference (ECC), $p.3346-3353$, Switzerland 2013.

[8] O. Fritsch, B. Henze, B. Lohmann: *Fast and Saturating Attitude Control for a Quadrotor Helicopter*, 2013 European Control Conference (ECC), July 17-19, $p.3851-p.3857$.

[9] O. Fritsch, B. Henze, B. Lohmann: *Fast and Saturating Thrust Direction control for a Quadrotor Helicopter*, Automatisierungstechnik 3/2013, $p.172-p.181$.

[10] P. De Monte, B. Lohmann: *Trajectory Tracking Control for a Quadrotor Helicopter based on Backstepping using a Decoupling Quaternion*

Parametrization in 21st Mediterranean Conference on Control & Automation (MED), Greece, June 25-28, 2013.

[11] E. Fresk, G. Nikolakopoulos: *Full Quaternion Based Attitude Control for a Quadrotor*, 2013 European Control Conference (ECC), July 17-19, *p*.3864−*p*.3869.

[12] M. Vanin: *Modeling, identification and navigation of autonomous air vehicles*, Master's Thesis, KTH, 2013

[13] H. H. Asada: Lecture notes to *Introduction to Robotics*, MIT, chapter 7, 2005. Available at `http://ocw.mit.edu/courses/mechanical-engineering/2-12-introduction-to-robotics-fall-2005/lecture-notes/chapter7.pdf`

[14] S. Bouabdallah: *Design and control of quadrotors with application to autonomous flying*, PhD Thesis, EPFL, 2006.

[15] T. Bresciani: *Modelling, Identification and Control of a Quadrotor Helicopter*, Master's Thesis, Lund University, 2008.

[16] R. A. Serway, J. W. Jewett: *Physics for Scientists and Engineers with Modern Physics* (9th edition), Brooks/Cole, USA 2008.

[17] S. Bouabdallah, R. Siegwart: *Full Control of a Quadrotor*, The IEEE International Conference on Intelligent Robots (IROS), 2007.

[18] M. Shuster: *A survey of attitude representations*, The Journal of the Astronautical Sciences, volume 41, no 4, 1993.

[19] J. Vince: *Quaternions for Computer Graphics*, Springer (2011).

[20] J. B. Kuipers: *Quaternions and Rotation Sequances*, Princeton University Press, 2002.

[21] H. Bolandi, M. Rezaei, R. Mohsenipour, H. Nemati, S. M. Smailzadeh: *Attitude Control of a Quadrotor with Optimized PID Controller*, Intelligent Control and Automation, 2013, 4, 335-342.

[22] M. H. Tanveer, S. F. Ahmed, D. Hazry, F. A. Warsi, M. K. Joyo: *Stabilized Controller Design for Attitude and Altitude Controlling of Quad-rotor Under Disturbance and Noisy Conditions*, American Journal of Applied Sciences 10 (8): 819-831, 2013.

[23] F. J. Niroumand, A. Fakharian, M. S. Seyedsajadi: *Fuzzy Integral Backstepping Control Approach in Attitude Stabilization of a Quadrotor UAV*, 13th Iranian Conference on Fuzzy Systems (IFSC) 2013.

[24] M. Schreier: *Modeling and Adaptive Control of a Quadrotor*, Proceedings of 2012 IEEE International Conference on Mechatronics and Automation August 5 - 8, Chengdu, China.

[25] M. Mohammadi, A. M. Shahri: *Decentralized Adaptive Stabilization Control for a Quadrotor UAV*, Proceeding of the 2013 RSI/ISM International Conference on Robotics and Mechatronics February 13-15,2013, Tehran, Iran.

[26] H. K. Khalil: *Nonlilnear systems*, chapter 4, 3rd edition, Prentice-Hall Inc., 2002.

[27] Video clip by M. French: *Measuring Mass Moment of Inertia - Brain Waves*, available at`http://youtu.be/m9iHEanmNWc`.

[28] G. H. Martin: *Kinematics and Dynamics of Machines*, Second Edition, McGraw-Hill International Book Company, 1982.

[29] M. Amoozadeh: *Smart mobility lab manual*, KTH Stockholm 2013. Available at `https://code.google.com/p/kth-smart-mobility-lab/downloads/list`

[30] M. Vanin: *Quadrocopter Manual for the Smart Mobility Lab*, KTH Stockholm 2014. Available at `http://www.kth.se/polopoly_fs/1.451019!/Menu/general/column-content/attachment/quad_manual.pdf`