

Institutionen för datavetenskap  
Department of Computer and Information Science

**Master's Thesis**

# **State Estimation and Control of a Quadrotor, using Monocular SLAM**

**Jonatan Olofsson**

Reg Nr: LiTH-ISY-EX--YY/XXXX--SE  
Linköping YYYY



**Linköping University**  
**INSTITUTE OF TECHNOLOGY**

Department of Computer and Information Science  
Linköpings universitet  
SE-581 83 Linköping, Sweden



**Master's Thesis**

**State Estimation and Control of a  
Quadrotor, using Monocular SLAM**


**Jonatan Olofsson**

Reg Nr: LiTH-isy-ex--yy/xxxx--se  
Linköping yyyy

Supervisor: **Rudol, Piotr**  
IDA, Linköpings universitet  
**Schön, Thomas**  
ISY, Linköpings universitet  
**Wzorek, Mariusz**  
IDA, Linköpings universitet

Examiner: **Doherty, Patrick**  
IDA, Linköpings universitet



	<b>Avdelning, Institution</b> Division, Department  Department of Computer and Information Science Department of Computer and Information Science Linköpings universitet SE-581 83 Linköping, Sweden		<b>Datum</b> Date  YYYY-09-19
	<b>Språk</b> Language  <input type="checkbox"/> Svenska/Swedish <input checked="" type="checkbox"/> Engelska/English  <input type="checkbox"/> _____	<b>Rapporttyp</b> Report category  <input type="checkbox"/> Licentiatavhandling <input checked="" type="checkbox"/> Examensarbete <input type="checkbox"/> C-uppsats <input type="checkbox"/> D-uppsats <input type="checkbox"/> Övrig rapport <input type="checkbox"/> _____	<b>ISBN</b> _____ <b>ISRN</b> LiTH-ISY-EX--YY/XXXX--SE <b>Serietitel och serienummer ISSN</b> Title of series, numbering _____
<b>URL för elektronisk version</b> <a href="http://www.ida.liu.se/divisions/aics/">http://www.ida.liu.se/divisions/aics/</a> <a href="http://www.ida.liu.se/divisions/aics/">http://www.ida.liu.se/divisions/aics/</a>			
<b>Titel</b> Title                      State Estimation and Control of a Quadrotor, using Monocular SLAM   <b>Författare</b> Jonatan Olofsson Author			
<b>Sammanfattning</b> Abstract  <p>This thesis treats the problem of modelling, estimating and autonomously controlling an Unmanned Aerial Vehicle. The LinkQuad quadrotor developed at Linköping University is used as a development platform, although the results are general for quadrotors and easily extended to other UAV platforms.</p> <p>In the thesis, a control system is developed to fuse state-estimation algorithms with advanced non-linear control.</p> <p>The problem of state-estimation and control of Small UAV's (SUAV's) is often approached using off-board cameras and sensors to get a precise pose estimation. This however restricts the utility of the controlled UAV's to a very limited space where sophisticated and expensive gear is available and properly configured. By using only on-board sensors, not only is the utility of the UAV greatly increased, but costs are also cut.</p> <p>To assist the positioning, the quadrotor is equipped with a camera utilized by a library for monocular video-based SLAM, Simultaneous Localization And Mapping. In the thesis, an algorithm is suggested for transforming the localization in videoframe-coordinates to world coordinates which can be used in the state-estimation.</p>			
<b>Nyckelord</b> Keywords      automated landing, slam, ptam, control, uav			



# Abstract

This thesis treats the problem of modelling, estimating and autonomously controlling an Unmanned Aerial Vehicle. The LinkQuad quadrotor developed at Linköping University is used as a development platform, although the results are general for quadrotors and easily extended to other UAV platforms.

In the thesis, a control system is developed to fuse state-estimation algorithms with advanced non-linear control.

The problem of state-estimation and control of Small UAV's (SUAV's) is often approached using off-board cameras and sensors to get a precise pose estimation. This however restricts the utility of the controlled UAV's to a very limited space where sophisticated and expensive gear is available and properly configured. By using only on-board sensors, not only is the utility of the UAV greatly increased, but costs are also cut.

To assist the positioning, the quadrotor is equipped with a camera utilized by a library for monocular video-based SLAM, Simultaneous Localization And Mapping. In the thesis, an algorithm is suggested for transforming the localization in videoframe-coordinates to world coordinates which can be used in the state-estimation.

# Sammanfattning

Svenskt abstract kan man placera här.





# Acknowledgments



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Unmanned Aerial Vehicles . . . . .	1
1.2	The Platform . . . . .	2
1.3	Previous work . . . . .	3
1.3.1	Autonomous Landing and Control . . . . .	3
1.3.2	Visual SLAM . . . . .	4
1.4	Objectives and Limitations . . . . .	4
1.5	Contributions . . . . .	5
1.6	Thesis Outline . . . . .	5
<b>2</b>	<b>State Estimation</b>	<b>7</b>
2.1	The Filtering Problem . . . . .	7
2.2	The Unscented Kalman Filter . . . . .	10
2.3	Motion Model . . . . .	12
2.3.1	Coordinate Frames . . . . .	12
2.3.2	Kinematics . . . . .	15
2.3.3	Dynamics . . . . .	15
2.4	Sensor Models . . . . .	20
2.4.1	Accelerometers . . . . .	20
2.4.2	Gyroscopes . . . . .	20
2.4.3	Magnetometers . . . . .	20
2.4.4	Pressure Sensor . . . . .	21
2.4.5	Camera . . . . .	21
<b>3</b>	<b>Non-linear Control</b>	<b>25</b>
3.1	The Linear Quadratic Controller . . . . .	25
3.2	State-Dependent Riccati Equations and LQ Gain Scheduling . . . . .	26
3.3	Control Model . . . . .	27
<b>4</b>	<b>Monocular SLAM</b>	<b>29</b>
4.1	Filtering Based Solutions . . . . .	29
4.2	Keyframe-based SLAM . . . . .	30
4.3	The PTAM Library . . . . .	30
4.3.1	Operation . . . . .	31
4.3.2	Modifications to the Library . . . . .	31

4.3.3	Camera Lens . . . . .	32
<b>5</b>	<b>Finite State-Machines</b>	<b>33</b>
5.1	Implemented Modes . . . . .	33
5.1.1	Hovering . . . . .	33
5.1.2	PTAM initialization . . . . .	34
5.1.3	Free Flight . . . . .	34
5.1.4	Landing . . . . .	34
<b>6</b>	<b>Results</b>	<b>37</b>
6.1	Experiment Setup . . . . .	37
6.2	Modelling . . . . .	38
6.2.1	Accelerometers . . . . .	38
6.2.2	Gyroscopes . . . . .	40
6.2.3	Pressure Sensor . . . . .	41
6.2.4	Camera . . . . .	43
6.3	Filtering . . . . .	45
6.4	Control . . . . .	56
<b>7</b>	<b>Discussion</b>	<b>57</b>
7.1	Flapping . . . . .	57
<b>8</b>	<b>Concluding Remarks</b>	<b>59</b>
8.1	Conclusions . . . . .	59
8.2	Further work . . . . .	59
<b>A</b>	<b>CRAP</b>	<b>65</b>
A.1	. . . . .	65

# Chapter 1

## Introduction

The goal for this thesis was to develop and implement a high-level control system for the quadrotor developed by the AIICS<sup>1</sup> team at Linköping University, the LinkQuad. The system was to demonstrate autonomous landing, and while this goal could not be reached within the time-frame of one Master's thesis, the control system developed in this thesis shows good promise.

Although the LinkQuad quadrotor was used as development platform for this thesis, the results are general and portable to other quadrotors and, by extension, other UAV's. The LinkQuad is considered a SUAV, a Small Unmanned Aerial Vehicle - A UAV small enough to be carried by man [? ].

The size of the system poses limitations on the payload and processing power available in-flight. This means that the implementations has to be done in an efficient manner on the small computers that are available on the LinkQuad.

These limitations, however, does not necessarily limit us from utilizing advanced estimation and control techniques, and it turns out that the LinkQuad design is in fact very suitable for the camera-based pose estimation due to its dual onboard computers. The video-based estimation can effectively be detached from the core control and state estimation and modularized as an independent virtual sensor, which is exploited in the thesis.

### 1.1 Unmanned Aerial Vehicles

Unmanned Aerial Vehicles (UAV's) have been imagined and constructed for millennia, starting in ancient Greece and China[37]. The modern concept of UAV's was introduced in the first world war, which illuminates the dominant role that the military has played in the field over the last century. A commonly cited alliteration is that UAV's are intended to replace human presence in missions that are "Dull, Dirty and Dangerous".

While the military continues to lead the development in the field [29], recent years have seen a great increase in domestic and civilian applications [41]. These

---

<sup>1</sup><http://www.ida.liu.se/divisions/aiics/>

applications range from pesticide dispersing and crop monitoring to traffic control, border watch and rescue scenarios [8].

The type of UAV that is used in the implementation of this thesis falls under the category of Small Unmanned Aerial Vehicles (SUAV's) [37]. SUAV's are designed to be man-portable in weight and size and is thus limited in payload and available processing power. This limitation, in combination with the unavailability of indoor positioning (e.g. GPS), has led to extensive use of off-board positioning and control in recent research. Systems developed for instance by Vicon<sup>2</sup> and Qualisys<sup>3</sup> yield positioning with remarkable precision, but they also limit the application to a confined environment with an expensive setup.

This thesis seeks a different approach, with an efficient self-contained on-board implementation. GPS and external cameras are replaced by inertial sensors and an on-board camera which uses visual SLAM to position the LinkQuad relative to its surroundings.

## 1.2 The Platform

The LinkQuad is a modular quadrotor developed at Linköping University. The core configuration is equipped with standard MEMS sensors (accelerometers, gyroscopes and a magnetometer), but for our purposes, a monocular camera has also been mounted, which feeds data into a microcomputer specifically devoted to the processing of the camera feed. This devoted microcomputer is what allows the primary on-board microcomputer to focus on state estimation and control, without being overloaded by video processing.

The primary microcomputer is running a framework named C++ Robot Automation Platform (CRAP), which was developed by the thesis' author for this purpose. CRAP is a light-weight automation platform with a purpose similar to that of ROS<sup>4</sup>. It is, in contrast to ROS, primarily designed to run on the kind relatively low-end Linux systems that fits the payload and power demands of a SUAV. The framework is further described in Appendix A.

Using the framework, the functionality of the implementation is distributed in separate modules:

- **Observer** Sensor fusing state estimation. Chapter 2.
- **Control** Affine Quadratic Control. Chapter 3.
- **Logic** State-machine for scheduling controller parameters and reference trajectory. Chapter 5.

The platform is also equipped with two microcontrollers, responsible for sensor sampling, motor control, flight logging etc, as depicted in Figure 1.1.

---

<sup>2</sup><http://www.vicon.com/>

<sup>3</sup><http://www.qualisys.com/>

<sup>4</sup><http://www.ros.org/>

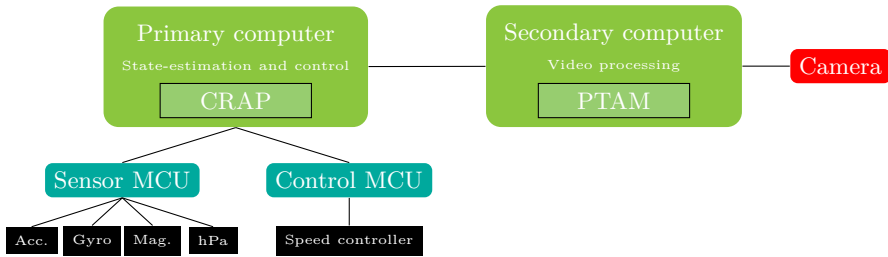


Figure 1.1. LinkQuad schematic.

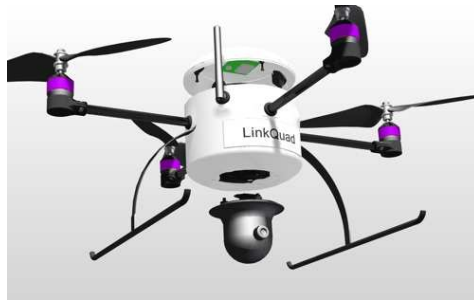


Figure 1.2. The LinkQuad development platform.

## 1.3 Previous work

The problem of positioning an unmanned quadrotor using visual feedback is a problem which has received extensive attention during recent years, and only recently been implemented with convincing results [2, 39]. Few have attempted to use on-board sensors only, but have relied on external setups to track the motions of a quadrotor - reportedly with high precision. Fewer still have succeeded using strictly real-time algorithms running on the limited processing power that standard SUAV's generally are equipped with, although successful implementations do exist, e.g. [? ]. The LinkQuad is in that regard, with its distributed processing power, well suited as a development platform when studying this problem.

### 1.3.1 Autonomous Landing and Control

The problem of landing a quadcopter can to a large extent be boiled down to achieving good pose estimates using available information. This problem is studied in e.g. [26, 4], but perhaps the most interesting results are obtained in [2, 39], where the ideas from [20] of using monocular SLAM are implemented on a UAV platform and excellent results are obtained in terms of achieving real flight, although some problems remain regarding the long-term stability of the controller.

A landing control scheme, inspirational to the approach in this thesis, is suggested by [4], which is summarized in Section 5.1.4.

Both Linear Quadratic control and PID controllers have been used for control in aforementioned projects, and ambiguous results have been attained as to which is better [? ]. Continued effort of quadrotor modelling have however shown great potential[3].

### 1.3.2 Visual SLAM

A first take on a Visual SLAM (VSLAM) algorithm is presented in [19], resting on the foundation of a Rao-Blackwellized particle filter with Kalman filter banks. The algorithm presented in [19] also extends to the case of multiple cameras, which could be interesting in a longer perspective. Similar approaches, using common Kalman Filtering solutions have been implemented by e.g. [7, 9], and are available open-source<sup>5</sup>.

An alternative approach to VSLAM is suggested by [20] in the PTAM - Parallel Tracking And Mapping - library it presents. This library splits the tracking problem - estimating the camera position - of SLAM from the mapping problem - globally optimizing the positions of registered features with regards to each other. Without the constraint of real-time mapping, this optimization can run more advanced algorithms at a slower pace than required by the tracking. A modified version of the said library has also demonstrationaly been implemented on an iPhone [21]. This is of special interest to this thesis, since it demonstrates a successful implementation in a resource-constrained environment similar to that available on a SUAV.

The algorithm proposed in [20] uses selected keyframes from which offsets are calculated continuously in the tracking thread, while the map optimization problem is addressed separately as fast as possible using information only from these keyframes. This opposed to the traditional VSLAM filtering solution where each frame has to be used for continuous filter updates. Several existing implementations exist with this technique, e.g. as described by [7].

By, for instance, not considering the full uncertainties in either camera pose or feature location, the complexity of the algorithm is reduced and the number of studied points can be increased to achieve better robustness and performance than when a filtering solution is used[35]. Again, this is the method on which [39] bases its implementation.

## 1.4 Objectives and Limitations

The main objective for this thesis was to create a high-level control and state-estimation system and to demonstrate this by performing autonomous landing with the LinkQuad. To achieve this, the main work was put into achieving theoretically solid positioning and control. Having control over the vehicle, landing is

---

<sup>5</sup><http://www.doc.ic.ac.uk/~ajd/Scene/>



then a matter of generating a suitable trajectory and detecting the completion of the landing.

Although time restricted the final demonstration of landing, the necessary tools were implemented, albeit lacking the tuning necessary for real flight.

This thesis does not cover the detection of suitable landing sites, nor any advanced flight control in limited space or with collision detection. The quadrotor modelling is extensive, but is mostly limited to a study of literature on the subject.

## 1.5 Contributions

During the thesis work, several tools for future development have been designed and developed. The *CRAP* framework collects tools that are usable in future projects and theses, both on the LinkQuad and otherwise. The modules developed for the *CRAP* framework include, but is far from limited to;

- General non-linear filtering, using EKF or UKF,
- General non-linear control, implemented as affine quadratic control,
- Extendable scheduling and reasoning through state-machines,
- Real-time plotting,
- Communication API for internal and external communication.

Furthermore, a general physical model of a quadrotor has been assembled. The model extends and clarifies the many sources of physical modelling available, and is presented in a scalable, general manner.

The state-estimation proposed in this thesis uses the full physical model derived in Chapter 2. While the model still needs tuning, it does show promising results, and a physically modelled quadrotor could potentially improve in-flight performance.

In Chapter 2, a general method for retaining the world-to-PTAM transform is proposed. This method could prove useful for extending the utility of the PTAM camera positioning library to more than its intended use in Augmented Reality. The utility for this has already been proven in e.g. [39], and providing the theory and implementation for this, as well as a proposed initialization routine, could be of great use in future work. The PTAM library has also been extended to full autonomy with the proposition of an automated initialization procedure as well as providing full detachment from the graphical interface.

All tools developed during the thesis are released under the GPL license and are available at <https://github.com/jonatanolofsson/>.

## 1.6 Thesis Outline

Following the introductory Chapter 1, four chapters are devoted to presenting the theory and the equations used in the implementation, in order;

- **State Estimation** State estimation theory and physical modelling of a quadrotor,
- **Non-linear Control** Non-linear control theory and its implementation,
- **Monocular SLAM** Video-based SLAM and its applications.

The following two chapters of the thesis, Chapters 6 and 7, present the numerical evaluation of the result and the following discussion respectively. Concluding remarks and suggestions for further work are then presented in Chapter 8.

A detailed description of the *CRAP* framework is appended to the thesis.

# Chapter 2

## State Estimation

A central part of automatic control is to know the state of the device you are controlling. The system studied in this thesis - the LinkQuad - is in constant motion, so determining the up-to-date position is of vital importance. This chapter deals with the estimation of the states relevant for positioning and controlling the LinkQuad. Filter theory and notation is established in Section 2.1.

In this thesis, an Unscented Kalman Filter (UKF) and an Extended Kalman Filter (EKF) both are evaluated. These filters both extends the linear Kalman filter theory to the non-linear case. The UKF circumvents the linearization used in the EKF in an appealing black-box way, albeit it is more sensitive to obscure cases in the physical model, as detailed in the discussion in Chapter 7. The theory of the EKF and UKF is treated in Section 2.1 and 2.2 respectively.

The motion model of the system is derived and discussed in Section 2.3.

As well as beeing modelled, the motions of the system are captured by the on-board sensors. A measurement  $y$  is related to the motion model by the sensor model  $h$ ;

$$y(t) = h(x(t), u(t), t) \quad (2.1)$$

The models for the sensors used in this work are discussed in Section 2.4.

### 2.1 The Filtering Problem

The problem of estimating the state of a system - in this case its position, orientation, velocity etc. - is in general filtering expressed as the problem of finding the state estimate,  $\hat{x}$ , that in a well defined best way (e.g. with Minimum Mean Square Error, MMSE) describes the behaviour of the system.

The evolution of a system plant is traditionally described by a set of differential equations that link the change in the variables to the current state and known inputs,  $u$ . This propagation through time is described by Eq. (2.2) ( $f_c$  denoting the continous case). The system is also assumed to be subject to an additive white Gaussian noise  $v(t)$  with known covariance  $Q$ . This introduces an uncertainty associated with the system's state, which accounts for imperfections in the model

compared to the physical real-world-system.

$$\dot{x}(t) = f_c(x(t), u(t), t) + v_c(t) \quad (2.2)$$

With numeric or analytical solutions, we can obtain the discrete form of (2.2), where only the sampling times are considered. The control signal,  $u(t)$ , is for instance assumed to be constant in the time interval, and we obtain the next predicted state directly, yielding the prediction of  $\hat{x}$  at the time  $t$  given the information at time  $t-1$ <sup>1</sup>. This motivates the notation used in this thesis -  $\hat{x}_{t|t-1}$ .

$$x_{t|t-1} = f(x_{t-1|t-1}, u_t, t) + v(t) \quad (2.3)$$

In the ideal case, a simulation of a prediction  $\hat{x}$  would with the prediction model in (2.3) fully describe the evolution of the system. To be able to provide a good estimate in the realistic case, however, we must also feed back measurements given from sensors measuring properties of the system.

These measurements,  $y_t$ , are fed back and fused with the prediction using the *innovation*,  $\nu$ .

$$\nu_t = y_t - \hat{y}_t \quad (2.4)$$

That is, the difference between the measured value and what would be expected in the ideal (predicted) case. To account for disturbances affecting the sensors, the measurements are also associated with an additive white Gaussian noise  $w(t)$ , with known covariance,  $R$ .

$$\hat{y}_t = h(\hat{x}_t, u_t, t) + w(t) \quad (2.5)$$

The innovation is then fused with the prediction to yield a new estimation of  $x$  given the information available as of the time  $t$  [12].

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t \nu_t \quad (2.6)$$

The choice of  $K_t$  is a balancing between of trusting the model, or trusting the measurement. In the Kalman filter framework, this balancing is made by tracking and weighing the uncertainties introduced by the prediction and the measurement noise.

**Algorithm 1 (Kalman Filter)** *For a linear system, given predictions and measurements with known covariances  $Q$  and  $R$ , the optimal solution to the filtering problem is given by the forward recursion<sup>2</sup>*

**Prediction update**

$$\hat{x}_{t|t-1} = A\hat{x}_{t-1|t-1} + Bu_t \quad (2.7a)$$

$$P_{t|t-1} = AP_{t-1|t-1}A^T + Q \quad (2.7b)$$

---

<sup>1</sup>Note that we have not yet performed any measurements that provide information about the state at time  $t$ .

<sup>2</sup>As first suggested by Rudolf E. Kálmán in [18].

**Measurement update**

$$K = P_{t|t-1} H^T (H P_{t|t-1} H^T + R)^{-1} \quad (2.8a)$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K (y - H \hat{x}_{t|t-1}) \quad (2.8b)$$

$$P_{t|t} = P_{t|t-1} + K H P_{t|t-1} \quad (2.8c)$$

Because of the assumptions on the noise and of the linear property of the innovation feedback, the Gaussian property of the noise is preserved in the filtering process. The system states can thus ideally be considered drawn from a normal distribution.

$$x \sim \mathcal{N}(\hat{x}, P_{xx}) \quad (2.9)$$

Conditioned on the state and measurements before time  $k$  (kept in the set  $\mathcal{Y}^k$ ), the covariance of the sample distribution is defined as

$$P_{xx}(t|k) = E \left[ \{x(t) - \hat{x}_{t|k}\} \{x(t) - \hat{x}_{t|k}\}^T | \mathcal{Y}^k \right]. \quad (2.10)$$

As new measurements are taken, the covariance of the state evolves with the state estimate as in Eq. (2.11) [17].

$$P_{xx}(t|t) = P_{xx}(t|t-1) - K_t P_{\nu\nu}(t|t-1) K_t^T \quad (2.11)$$

$$P_{\nu\nu}(t|t-1) = P_{yy}(t|t-1) + R(t). \quad (2.12)$$

With known covariances,  $K$  can be chosen optimally for the linear case as derived in e.g. [12];

$$K_t = P_{xy}(t|t-1) P_{\nu\nu}^{-1}(t|t-1). \quad (2.13)$$

Note that (2.13) is another, albeit equivalent, way of calculating (2.8a), which will be exploited in Section 2.2.

Although the Kalman filter is optimal in the linear case, no guarantees are given for the non-linear case. Several methods exist to give a sub-optimal estimate of the non-linear cases, two of which will be studied here.

One problem with the non-linear case is how to propagate the uncertainty, as described by the covariance, through the prediction and measurement models. With the assumed Gaussian prior, it is desirable to retain the Gaussian property in the posterior estimate, even though this clearly is in violation with the non-linear propagation, which generally does not preserve this property.

As the linear case is simple however;

$$P_{xx}(t|t-1) = A P(t|t) A^T + Q_t; \quad (2.14)$$

the novel approach is to linearize the system to yield  $A$  in every timestep. This method is called the Extended Kalman Filter, and is considered the de facto standard non-linear filter[16].

**Algorithm 2 (Extended Kalman Filter)** *The Kalman filter in Algorithm 2 is in the Extended Kalman filter extended to the non-linear case by straight-forward linearization where necessary.*

**Prediction update**

$$\hat{x}_{t|t-1} = f(\hat{x}_{t-t|t-1}, u_t) \quad (2.15a)$$

$$P_{t|t-1} = AP_{t-1|t-1}A^T + Q \quad (2.15b)$$

**Measurement update**

$$K = P_{t|t-1}H^T (HP_{t|t-1}H^T + R)^{-1} \quad (2.16a)$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K(y - h(\hat{x}_{t|t-1})) \quad (2.16b)$$

$$P_{t|t} = P_{t|t-1} + KHP_{t|t-1}, \quad (2.16c)$$

where

$$A = \left. \frac{\partial f(x, u)}{\partial x} \right|_{x=\hat{x}_{t|t}}, \quad H = \left. \frac{dh(x)}{dx} \right|_{x=\hat{x}_{t|t-1}}. \quad (2.17)$$

This linearization, some argue[17], fails to capture the finer details of highly non-linear systems and may furthermore be tedious to calculate, analytically or otherwise. An alternative approach, known as the Unscented Kalman Filter, is therefore discussed in Section 2.2.

## 2.2 The Unscented Kalman Filter

The basic version of the Unscented Kalman Filter was proposed in [17] based on the following intuition [17]

*With a fixed number of parameters it should be easier to approximate a Gaussian distribution than it is to approximate an arbitrary nonlinear function.*

The approach is thus to propagate the uncertainty of the system through the non-linear system and fit the results as a Gaussian distribution. The propagation is made by simulating the system in the prediction model for carefully chosen offsets from the current state called *sigma points*, each associated with a weight of importance. The selection scheme for these points can vary (and yield other types of filters), but a common choice is the *Scaled Unscented Transform* (SUT) [38]. The SUT uses a minimal set of sigma points needed to describe the first two moments of the propagated distribution - two for each dimension ( $n$ ) of the state vector and one for the mean.

$$\begin{aligned} \mathcal{X}_0 &= \hat{x} \\ \mathcal{X}_1 &= \hat{x} + \left( \sqrt{(n+\lambda)P_{xx}} \right)_i & i = 1, \dots, n \\ \mathcal{X}_i &= \hat{x} - \left( \sqrt{(n+\lambda)P_{xx}} \right)_i & i = n+1, \dots, 2n \end{aligned} \quad (2.18)$$

Variable	Value	Description
$\alpha$	$0 \leq \alpha \leq 1$ (e.g. 0.01)	Scales the size of the sigma point distribution. A small $\alpha$ can be used to avoid large non-local non-linearities.
$\beta$	2	As discussed in [15], $\beta$ affects the weighting of the center point, which will directly influence the magnitude of errors introduced by the fourth and higher order moments. In the strictly Gaussian case, $\beta = 2$ can be shown to be optimal.
$\kappa$	0	$\kappa$ is the number of times that the center-point is included in the set of sigma points, which will add weight to the centerpoint and scale the distribution of sigma points.

**Table 2.1.** Description of the parameters used in the SUT.

$$W_0^m = \frac{\lambda}{n+\lambda} \quad W_0^c = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta)$$

$$W_i^m = W_i^c = \frac{1}{2(n+\lambda)} \quad i = 1, \dots, 2n \quad (2.19)$$

$$\lambda = \alpha^2(n + \kappa) - n \quad (2.20)$$

The three parameters introduced here,  $\alpha$ ,  $\beta$  and  $\kappa$  are summarized in table 2.2. The term  $\left(\sqrt{(n+\lambda)P_{xx}}\right)_i$  is used to denote the  $i$ 'th column of the matrix square root  $\sqrt{(n+\lambda)P_{xx}}$ .

When the sigma points  $\mathcal{X}_i$  have been calculated, they are propagated through the non-linear prediction function and the resulting mean and covariance can be calculated.

$$\mathcal{X}_i^+ = f(\mathcal{X}_i, u, t) \quad i = 0, \dots, 2n \quad (2.21)$$

$$\hat{x} = \sum_{i=0}^{2n} W_i^m \mathcal{X}_i^+ \quad (2.22)$$

$$P_{xx} = \sum_{i=0}^{2n} W_i^c \{ \mathcal{X}_i^+ - \hat{y} \} \{ \mathcal{X}_i^+ - \hat{y} \}^T \quad (2.23)$$

For the measurement update, similar results are obtained, and the equations (2.26)-(2.27) can be connected to equations (2.12)-(2.13).

$$\mathcal{Y}_i = h(\mathcal{X}_i, u, t) \quad i = 0, \dots, 2n \quad (2.24)$$

$$\hat{y} = \sum_{i=0}^{2n} W_i^m \mathcal{Y}_i \quad (2.25)$$

$$P_{yy} = \sum_{i=0}^{2n} W_i^c \{ \mathcal{Y}_i - \hat{y} \} \{ \mathcal{Y}_i - \hat{y} \}^T \quad (2.26)$$

$$P_{xy} = \sum_{i=0}^{2n} W_i^c \{ \mathcal{X}_i - \hat{x} \} \{ \mathcal{Y}_i - \hat{y} \}^T \quad (2.27)$$

As can be seen from the equations in this section, the UKF handles the propagation of the probability densities through the model without the need for explicit calculation of the Jacobians or Hessians for the system. The filtering is based solely on function evaluations of small offsets from the expected mean state<sup>3</sup>, be it for the measurement functions, discussed in Section 2.4, or the time update prediction function - the motion model.

## 2.3 Motion Model

As the system studied in the filtering problem progresses through time, the state estimate can be significantly improved if a prediction is made on what measurements can be expected, and evaluating the plausibility of each measurement after how well they correspond to the prediction. With assumed Gaussian white noise distributions, this evaluation can be done in the probabilistic Kalman framework as presented in Sections 2.1-2.2, where the probability estimate of the sensors' measurements are based on the motion model's prediction. In this section, a motion model is derived and evaluated. The model is presented in its continuous form, since the discretization is made numerically on-line.

### 2.3.1 Coordinate Frames

In the model of the quadrotor, there are several frames of reference.

**North-East-Down (NED)** The NED-frame is fixed at the center of gravity of the quadrotor. The NED system's  $\hat{z}$ -axis is aligned with the gravitational axis and the  $\hat{x}$ -axis along the northern axis. The  $\hat{y}$ -axis is chosen to point east to form a right-hand system.

**North-East-Down Earth Fixed (NEDEF)** This frame is fixed at a given origin in the earth (such as the take-off point) and is considered an inertial frame, but is in all other aspects equivalent to the NED frame. All states are expressed in this frame of reference unless explicitly stated otherwise. This frame is often referred to as the "world" coordinate system, or " $w$ " for short in equations in disambiguation from the NED system.

**Body-fixed (BF)** The body-fixed coordinate system is fixed in the quadrotor with  $\hat{x}$ -axis in the forward direction and the  $z$ -axis in the downward direction - as depicted in Figure ??.

**Propeller fixed** Each of the propellers are associated with their own frame of reference,  $P_i$ , which tracks the virtual tilting of the thrust vector due to flapping, discussed in Section 2.3.3.

---

<sup>3</sup>It is not, however, merely a central difference linearization of the functions, as [17] notes.



**Camera frame** This is the frame which describes the location of the camera.

**PTAM frame** This is the frame of reference used by the PTAM video SLAM library. This frame is initially unknown, and its recovery is discussed in Section 2.4.5.

**IMU frame** This is the body-fixed frame in which the IMU measurements are said to be done. The origin is thus fixed close to the inertial sensors.

The coordinate frames are visualized in Figures 2.1-2.2.

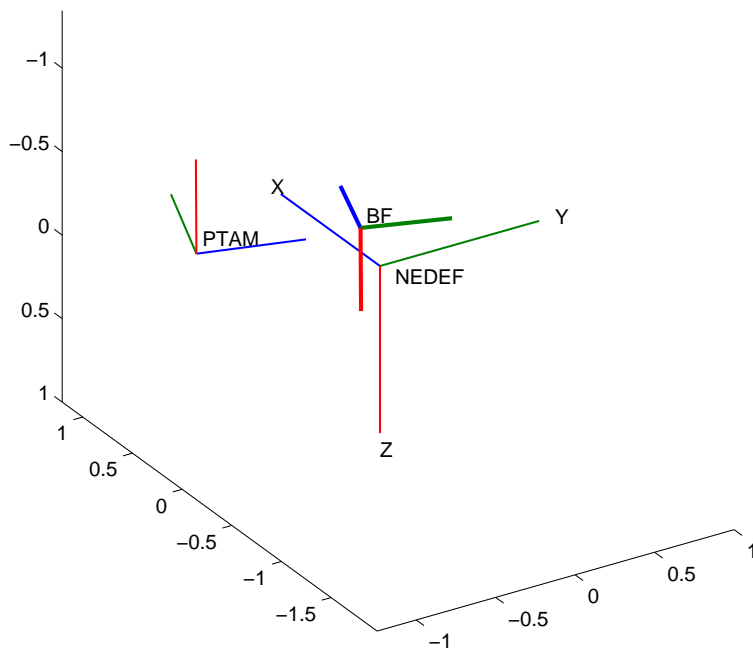


Figure 2.1.

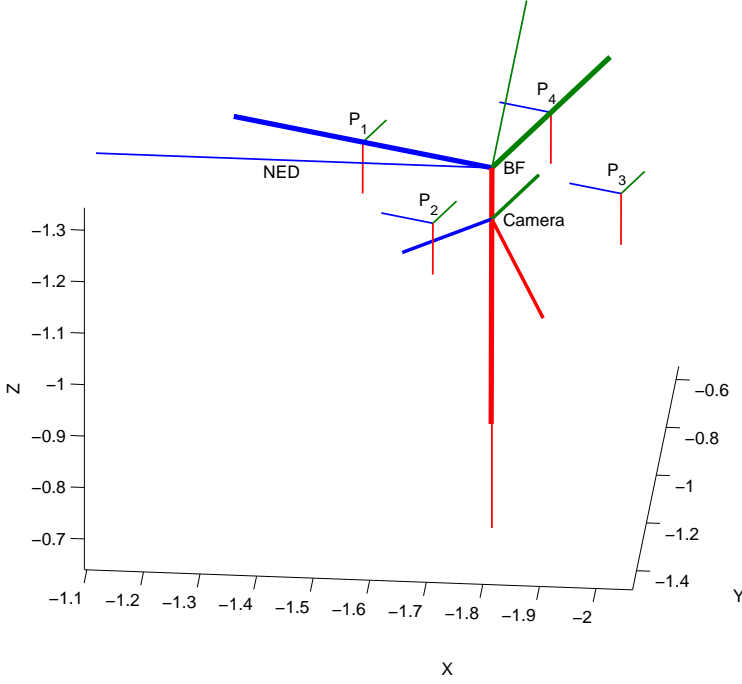


Figure 2.2.

The conversion between the reference frames are characterized by a transformation including translation and a three-dimensional rotation. Both the origin of the body-centered reference frames - the quadrotor's position - and the rotation of the body-fixed system are stored as system states.

The centers of each of the propeller fixed coordinate systems are parametrized on the height  $h$  and distance  $d$  from the center of gravity as follows

$$D_0 = (d, 0, h)^{BF} \quad (2.28)$$

$$D_1 = (0, -d, h)^{BF} \quad (2.29)$$

$$D_2 = (-d, 0, h)^{BF} \quad (2.30)$$

$$D_3 = (0, d, h)^{BF} \quad (2.31)$$

In the following sections, vectors and points in e.g. the NED coordinate systems are denoted  $x^{NED}$ . Rotation described by unit quaternions are denoted  $R(q)$  for the quaternion  $q$ , corresponding to the matrix rotation [22]<sup>4</sup> given by

$$\begin{pmatrix} q_1^2 + q_i^2 - q_j^2 - q_k^2 & 2q_iq_j - 2q_1q_k & 2q_iq_k + 2q_1q_j \\ 2q_iq_j + 2q_1q_k & q_1^2 - q_i^2 + q_j^2 - q_k^2 & 2q_jq_k - 2q_1q_i \\ 2q_iq_k - 2q_1q_j & 2q_jq_k + 2q_1q_i & q_1^2 - q_i^2 - q_j^2 + q_k^2 \end{pmatrix}. \quad (2.32)$$

<sup>4</sup>[22] uses a negated convention of signs compared to what is used here.

Rotation quaternions describing the rotation *to frame a from frame b* is commonly denoted  $q^{ab}$ , whereas the  $b$  may be dropped if the rotation is global, i.e. relative the NEDEF system.  $a$  and  $b$  are, where unambiguous, replaced by the first character of the name of the reference frame. Full transformations between coordinate systems - including rotation, translation and scaling - are similarly denoted  $\mathcal{J}^{ab}$ .

### 2.3.2 Kinematics

The motions of the quadrotor are described by the following relations [31]:

$$\dot{\xi} = V \quad (2.33a)$$

$$\begin{pmatrix} \dot{q}_0^{wb} \\ \dot{q}_i^{wb} \\ \dot{q}_j^{wb} \\ \dot{q}_k^{wb} \end{pmatrix} = -\frac{1}{2} \begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & -\omega_z & \omega_y \\ \omega_y & \omega_z & 0 & -\omega_x \\ \omega_z & -\omega_y & \omega_x & 0 \end{pmatrix} \begin{pmatrix} q_0^{wb} \\ q_i^{wb} \\ q_j^{wb} \\ q_k^{wb} \end{pmatrix} \quad (2.33b)$$

In practice, a normalization step also has to be added to account for the unit length constraint on rotation quaternions.

### 2.3.3 Dynamics

The motions of the quadrotor can be fully mathematically explained by the forces and moments acting on the vehicle. Using the rigid-body assumption, Eulers' extension of Newton's laws of motion for the quadrotor's center of gravity,  $\mathcal{G}$ , yields

$$\dot{V} = a_{\mathcal{G}}^w = R(q^{wb}) \frac{1}{m} \sum F \quad (2.34a)$$

$$\dot{\omega} = R(q^{wb}) I_{\mathcal{G}}^{-1} \sum M_{\mathcal{G}} \quad (2.34b)$$

The main forces acting upon the quadrotor are the effects of three different components

$\sum_{i=0}^3 F_{ri}$  Rotor thrust,

$F_g$  Gravity,

$F_{wind}$  Wind.

Of these, the gravity is trivially described with the gravitational acceleration and the total mass of the quadrotor as

$$F_g = mg \cdot e_3^{NED} \quad (2.35)$$

The following sections will describe the rotor thrust and wind forces respectively. Additionally, other minor forces and moments are discussed in 2.3.3.

## Rotor thrust

Each of the four propellers on the quadrotor induce a torque and a thrust vector on the system, proportional to the square of the propeller velocity. The rotational velocity of the propeller is directly influenced by the controller. It may thus be modelled as a first order system - using the time constant  $\tau_{rotor}$  with the reference velocity as input, as in Eq. (2.36). For testing purposes, or where the control signal is not available, Eq. (2.37) may be used instead.

$$\dot{\omega}_{ri} = \frac{1}{\tau_{rotor}} (\omega_{ri} - r_i) \quad (2.36)$$

$$\dot{\omega}_{ri} = 0 \quad (2.37)$$

Due to the differences in relative airspeed around the rotor blade tip as the blades move either along or against the wind-relative velocity, the lifting force on the blade will vary around a rotation lap. This unbalance in lifting force will cause the blades to lean and the direction of the thrust vector to vary with regards to the motions of the quadrotor.

This phenomenon is called *flapping*, and is discussed in e.g. [31]. The flapping of the rotors and the centrifugal force acting upon the rotating blades will result in that the tilted blade trajectories will form a cone with the plane to which the rotor axis is normal. These motions of the propellers add dynamics to the description of the quadrotors motion which must be considered in a deeper analysis.

It is desirable, for the purpose of this thesis and considering computational load, to find a closed-form solution to the flapping equations. This implies several approximations and restrictions which will be discussed in 7.1. The resulting flapping angles and their impact on the thrust vectors can be described as in equations (2.38a)-(2.39) [31, 32, 23].

The momenta induced by the propeller rotation and thrust are described in equations (2.38b)-(2.38c). All equations in this section are given in the body-fixed coordinate system.

$$F_{ri} = C_T \rho A_r R^2 \omega_{ri}^2 \begin{pmatrix} -\sin(a_{1si}) \\ -\cos(a_{1si}) \sin(b_{1si}) \\ -\cos(a_{1si}) \cos(b_{1si}) \end{pmatrix} \quad (2.38a)$$

$$M_{Qi} = -C_Q \rho A R^3 \omega_{ri} |\omega_{ri}| e_3^{\text{NED}} \quad (2.38b)$$

$$M_{ri} = F_{ri} \times D_{ri} \quad (2.38c)$$

The equations for the flapping angles  $(a_{1si}, b_{1si})$  are derived in [31, 32, 23], but are in (2.39) extended to include the velocity relative to the wind.  $V_{ri(n)}$  denotes the  $n$ 'th element of the vector  $V_{ri}$ .

$$V_{\text{rel}} = V - V_{\text{wind}} \quad (2.39a)$$

$$V_{ri} = V_{\text{rel}} + \Omega \times D_{ri} \quad (2.39b)$$

$$\mu_{ri} = \frac{\|V_{ri(1,2)}\|}{\omega_i R} \quad (2.39c)$$

$$\psi_{ri} = \arctan\left(\frac{V_{ri(2)}}{V_{ri(1)}}\right) \quad (2.39d)$$

$$\begin{pmatrix} a_{1s} i \\ b_{1s} i \end{pmatrix} = \begin{pmatrix} \cos(\psi_{ri}) & -\sin(\psi_{ri}) \\ \sin(\psi_{ri}) & \cos(\psi_{ri}) \end{pmatrix} \begin{pmatrix} \frac{1}{1 - \frac{\mu_{ri}^2}{2}} \mu_{ri} (4\theta_{twist} - 2\lambda_i) \\ \frac{1}{1 + \frac{\mu_{ri}^2}{2}} \frac{4}{3} \left( \frac{C_T}{\sigma} \frac{2}{3} \frac{\mu_{ri} \gamma}{a} + \mu_{ri} \right) \end{pmatrix} + \begin{pmatrix} \frac{-\frac{16}{\gamma} \left( \frac{\omega_\theta}{\omega_{ri}} \right) + \left( \frac{\omega_\psi}{\omega_{ri}} \right)}{1 - \frac{\mu_{ri}^2}{2}} \\ \frac{-\frac{16}{\gamma} \left( \frac{\omega_\psi}{\omega_{ri}} \right) + \left( \frac{\omega_\theta}{\omega_{ri}} \right)}{1 + \frac{\mu_{ri}^2}{2}} \end{pmatrix} \quad (2.39e)$$

$$\lambda_i = \mu \alpha_{si} + \frac{v_{1i}}{\omega_i R} \quad (2.39f)$$

$$v_{1i} = \sqrt{-\frac{V_{rel}^2}{2} + \sqrt{\left(\frac{V_{rel}^2}{2}\right)^2 + \left(\frac{mg}{2\rho A_r}\right)^2}} \quad (2.39g)$$

$$C_T = \frac{\sigma a}{4} \left\{ \left( \frac{2}{3} + \mu_{ri}^2 \right) \theta_0 - \left( \frac{1}{2} + \frac{\mu^2}{2} \right) \theta_{twist} + \lambda \right\} \quad (2.39h)$$

$$\alpha_{si} = \frac{\pi}{2} - \arccos\left(-\frac{V_{rel} \cdot e_z}{\|V_{rel}\|}\right) \quad (2.39i)$$

$$C_Q = \sigma a \left[ \frac{1}{8a} (1 + \mu_{ri}^2) \bar{C}_d + \lambda \left( \frac{1}{6} \theta_0 - \frac{1}{8} \theta_{twist} + \frac{1}{4} \lambda \right) \right] \quad (2.39j)$$

## Wind

For describing the wind's impact on the quadrotor motion, a simple wind model is applied where the wind is modelled with a static velocity that imposes forces and moments on the quadrotor. The wind velocity vector is estimated by the observer and may thus still vary in its estimation through the measurement update. The wind velocities in the filter are given in the NEDEF reference frame.

The wind drag force is calculated using equation (2.40), whereas the moments are given by equations (2.41). In this thesis, the moments acting on the quadrotor body (as opposed to the rotors) are neglected or described by moments imposed by the wind acting on the rotors.

$$F_{wind} = F_{wind,body} + \sum_{i=0}^3 F_{wind,ri} \quad (2.40a)$$

$$F_{wind,body} = -\frac{1}{2} C_D \rho A V_{rel} \|V_{rel}\| \quad (2.40b)$$

Symbol	Expression	Description	Unit
$a$	$\frac{dC_L}{d\alpha} \approx 2\pi$	Slope of the lift curve.	$\frac{1}{\text{rad}}$
$\alpha_{si}$	-	Propeller angle of attack.	rad
$A_r$	-	Rotor disk area.	$\text{m}^2$
$c$	-	Blade chord - the (mean) length between the trailing and leading edge of the propeller.	m
$C_L$	-	Coefficient of lift.	1
$C_T$	*	Coefficient of thrust. This is primarily the scaling factor for how the thrust is related to the square of $\omega_i$ , as per 2.38a.	1
$C_{T0}$	-	Linearization point for thrust coefficient.	1
$C_Q$	*	Torque coefficient. This constant primarily is the scaling factor relating the square of $\omega_i$ to the torque from each rotor.	1
$\gamma$	$\frac{\rho a c R^4}{I_b}$	$\gamma$ is the Lock Number [23], described as the ratio between the aerodynamic forces and the inertial forces of the blade.	1
$I_b$	-	Rotational inertia of the blade	$\text{kgm}^2$
$\lambda_i$	*	$\lambda_i$ denotes the air inflow to the propeller.	1
$R$	-	Rotor radius.	m
$\rho$	-	Air density.	$\frac{\text{kg}}{\text{m}^3}$
$\sigma$	$\frac{\text{blade area}}{\text{disk area}}$	Disk solidity.	1
$\theta_0$	-	The angle of the propeller at its base, relative to the horizontal disk plane.	rad
$\theta_{\text{twist}}$	-	The angle with which the propeller is twisted.	rad
$\omega_\phi, \omega_\theta, \omega_\psi$	-	The rotational, body-fixed, velocity of the quadrotor.	$\frac{\text{rad}}{\text{s}}$
$\omega_{ri}$	-	The rotational velocity of propeller $i$ .	$\frac{\text{rad}}{\text{s}}$
$\mu_{ri}$	-	The normalized, air-relative, blade tip velocity.	1

**Table 2.2.** Table of symbols used in the flapping equations

Symbol	Expression	Description
$A$	-	3x3 matrix describing the area of the quadrotor, excluding the rotors.
$C_D$	-	3x3 matrix describing the drag coefficients of the quadrotor.
$C_{Dr}$	-	Propeller's coefficient of drag.

**Table 2.3.** Table of symbols used in the wind equations

$$F_{\text{wind},ri}^{BF} = -\frac{1}{2}\rho C_{D,r}\sigma A_r(V_{ri} \cdot e_{P_{ri}3}^{BF})\|V_{ri} \cdot e_{P_{ri}3}^{BF}\|e_{P_{ri}3}^{BF} \quad (2.40c)$$

$$M_{\text{wind}} = M_{\text{wind,body}} + \sum_{i=0}^3 M_{\text{wind},ri} \quad (2.41a)$$

$$M_{\text{wind,body}} \approx 0 \quad (2.41b)$$

$$M_{\text{wind},ri} = D_{ri}^{BF} \times F_{\text{wind},ri}^{BF} \quad (2.41c)$$

### Additional Forces and Moments

Several additional forces act on the quadrotor to give its dynamics in flight. Some of these are summarized briefly in this section, and are discussed further in [3]. Unless where explicitly noted, annotation is similar to Section 2.3.3.

#### Hub Force

$$C_H = \sigma a \left[ \frac{1}{4a}\mu\bar{C}_d + \frac{1}{4}\lambda\mu \left( \theta_0 + \frac{\theta_{twist}}{2} \right) \right] \quad (2.42)$$

$$F_{\text{hub},i} = -C_H \rho A R^2 \omega_i^2 \hat{x} \quad (2.43)$$

$$M_{\text{hub},i} = D_i \times F_{\text{hub},i} \quad (2.44)$$

#### Rolling Moment

$$C_{\text{RM}} = -\sigma a \mu \left[ \frac{1}{6}\theta_0 + \frac{1}{8}\theta_{twist} - \frac{1}{8}\lambda_i \right] \quad (2.45)$$

$$M_{\text{RM},i} = C_{\text{RM}} \rho A R^3 \omega_i^2 \quad (2.46)$$

**Ground Effect** As the vehicle gets close to ground, the wind foils of the propellers provide a cushion of air under the vehicle, giving extra lift.

$$T_{\text{IGE}} = \frac{1}{1 - \frac{R^2}{16z^2}} T \quad (2.47)$$

**Gyro Effects and Counter-Torque**  $I_{\text{rotor}}$  is the propeller inertia.

$$\begin{pmatrix} \dot{\omega}_{\theta}\dot{\omega}_{\psi}(I_{yy} - I_{zz}) + I_{\text{rotor}}\dot{\omega}_{\theta} \sum_{i=0}^4 \omega_{ri} \\ \dot{\omega}_{\theta}\dot{\omega}_{\psi}(I_{zz} - I_{xx}) + I_{\text{rotor}}\dot{\omega}_{\theta} \sum_{i=0}^4 \omega_{ri} \\ \dot{\omega}_{\theta}\dot{\omega}_{\phi}(I_{xx} - I_{yy}) + I_{\text{rotor}} \sum_{i=0}^4 \dot{\omega}_{ri} \end{pmatrix} \quad (2.48)$$

## 2.4 Sensor Models

This Section relates the estimated state of the quadrotor to the expected sensor measurements,  $\hat{y}$ . The first four sensors - accelerometers, gyroscopes, magnetometers and the pressure sensor - are described quite briefly, while the estimation of the camera's frame of reference requires some more detail.

It is common to include a GPS, providing world-fixed measurements, to prevent drift in the filtering process. Here, the GPS is replaced with a camera, which will be discussed in Section 2.4.5.

In most equations below, a zero-mean Gaussian term with known covariance is added to account for measurement noise. The Gaussian assumption may in some cases be severely inappropriate, but the Kalman filter framework requires its use.

### 2.4.1 Accelerometers

The accelerometers, as the name suggests, provides measurements of the accelerations of the sensor. In general, this does not directly correspond to the accelerations of the mathematical centre of gravity used as centre of the measured vehicle, which motivates correction for angular acceleration and rotation.

$$\hat{y}_{\text{acc}} = a_{\mathcal{G}} + \dot{\omega} \times r_{\text{acc}/\mathcal{G}} + \omega \times (\omega \times r_{\text{acc}/\mathcal{G}}) + e_{\text{acc}} \quad (2.49)$$

### 2.4.2 Gyroscopes

Gyroscopes, or rate gyroscopes specifically, measure the angular velocity of the sensor. Unlike acceleration, the angular rate is invariant of the relative position of the gyro to the center of gravity. However, gyroscope measurements are associated with a bias which may change over time. This bias term is introduced as a state variable which is modelled constant, Eq. (2.51), through the time update, leaving its adjustment to the filter measurement update.

$$y_{\text{gyro}} = \omega + b + e_{\text{gyro}} \quad (2.50)$$

$$\dot{b} = 0 + e_{\text{bias}} \quad (2.51)$$

### 2.4.3 Magnetometers

Capable of sensing magnetic fields, the magnetometers can be used to sense the direction of the earth's magnetic field and, from knowing the field at the current location, estimate the orientation of a vehicle.

$$y = R(q)m^e + e_m \quad (2.52)$$



Parameter	Description	Value	Unit
$L$	Temperature lapse rate.	0.0065	$\frac{\text{K}}{\text{m}}$
$M$	Molar mass of dry air.	0.0289644	$\frac{\text{kg}}{\text{mol}}$
$p_0$	Atmospheric pressure at sea level.	101325	Pa
$R$	Universal gas constant.	8.31447	$\frac{\text{J}}{\text{mol}\cdot\text{K}}$
$T_0$	Standard temperature at sea level.	288.15	K

**Table 2.4.** Table of Symbols used in the pressure equation, (2.54)

The Earth’s magnetic field can be approximated using the World Magnetic Model[6], which for Linköping, Sweden, yields

$$m^e = \begin{pmatrix} 15.7 & 1.11 & 48.4 \end{pmatrix}_{NEDEF}^T \mu T. \quad (2.53)$$

Magnetometer measurements are however very sensitive to disturbances, and in indoor flight, measurements are often useless due to electrical wiring, lighting etc. Thus, the magnetometers were not used for the state estimation in this thesis.

#### 2.4.4 Pressure Sensor

The air pressure,  $p$ , can be related to altitude using the following equation[?] ]

$$p = p_0 \left( 1 - \frac{L \cdot h}{T_0} \right)^{\frac{g \cdot M}{R \cdot L}} + e_p \quad (2.54)$$

#### 2.4.5 Camera

To estimate the position of the camera using the captured images, the PTAM-library is used. Because the main application of the PTAM library is reprojection of augmented reality into the image, consistency between a metric world-fixed coordinate frame (such as the NEDEF-system used on the LinkQuad), and the internally used coordinate system is not of importance - and thus not implemented in the PTAM positioning.

The measurements from the camera consists of the transform from the PTAM “world”-coordinates to the camera lens, in terms of

- translation,  $X^{\text{PTAM}}$ ,
- and orientation,  $q^{\text{PTAM},c}$ .

However, since the quite arbitrary[20] coordinate system of PTAM is neither of the same scale nor aligned with the quadrotor coordinate system, we need to estimate the affine transformation between the two in order to get useful results.

The transformation is characterized by

- a translation  $T$  to the origin,  $\mathcal{O}_{PTAM}$ ,
- a rotation  $R$  by the quaternion  $q^{Pw}$ ,
- and a scaling  $S$  by a factor  $s$ .

These are collected to a single transformation in Equation 2.55, forming the full transformation from the global NEDEF system to the PTAM coordinate frame.

$$x^{PTAM} = \underbrace{S(s)R(q^{Pw})T(-\mathcal{O}_{PTAM})}_{\triangleq \mathcal{J}^{Pw}, \text{transformation from camera to PTAM}} x^{NEDEF} \quad (2.55)$$

E.g. [13] studies this problem in the offline case, whereas the method used in this thesis extends the idea to the on-line case where no ground truth is available. This is performed by using the first measurement to construct an initial guess which then is filtered through time using the observer.

While PTAM exhibit very stable positioning, it has a tendency to move its origin due to association errors. To provide stable position measurements, we need to detect these movements and adjust the camera transformation accordingly. Initialization and tracking is dealt with in Section 2.4.5 and 2.4.5 respectively, whereas the teleportation problem is discussed in Section 2.4.5.

## Initialization

When the first camera measurement arrives, there is a need to construct a first guess of the transform. Since the PTAM initialization places the origin at what it considers the ground level, the most informed guess we can do without any information about the environment is to assume that this is a horizontal plane at zero height.

First, however, the orientation of the PTAM coordinate system is calculated from the estimated quadrotor orientation and the measurement in the PTAM coordinate frame;

$$q^{Pw} = q^{PTAM,c} q^{cb} q^{bw}. \quad (2.56)$$

To determine the distance to this plane according to the current estimation, Equation (2.57) is solved for  $\lambda$  in accordance with 2.58.

$$\begin{cases} \mathcal{O}_{PTAM} &= \xi + R(q^{wb})r_{\text{camera}/\mathcal{G}} + \lambda R(q^{wP}) \frac{X^{PTAM}}{|X^{PTAM}|} \\ \mathcal{O}_{PTAM} \cdot \hat{z} &= 0 \end{cases} \quad (2.57)$$

$$\lambda = -\frac{(\xi + R(q^{wb})r_{\text{camera}/\mathcal{G}}) \cdot \hat{z}}{\left(R(q^{wP}) \frac{X^{PTAM}}{|X^{PTAM}|}\right) \cdot \hat{z}} \quad (2.58)$$

By comparing the approximated distance with the distance measured in the PTAM coordinate system, we obtain a starting guess for the scaling factor,  $s$ ;

$$s = \frac{|X^{PTAM}|}{|\lambda|}. \quad (2.59)$$

Together, these parameters form an initial estimate of the transformation connecting the PTAM reference frame. This estimate could be inserted into the global observer filter - introducing eight new states containing  $q^{Pw}$ ,  $s$  and  $\mathcal{O}_{\text{cam}}$  - although little benefit of this has been observed in simulated testing. Because the PTAM coordinate system is defined fixed in the global NEDEF coordinate system, the transform parameters are unchanged in the observer's time update.

### Continuous Refinement

The measurement update is made separate from the update of the inertial sensors, using the measurement equations in (2.60), expanding the equation derived in Eqs. (2.55) and (2.56).

$$\hat{X}^{\text{PTAM}} = \mathcal{J}^{Pw}(\xi + R(q^{wb})r_{\text{camera}/\mathcal{G}}) + e_{\text{PTAM},X} \quad (2.60a)$$

$$\hat{q}^{\text{PTAM},c} = q^{Pw}q^{wb}q^{bc} + e_{\text{PTAM},q} \quad (2.60b)$$

### Teleportation

The PTAM tracking may sometimes exhibit a “teleporting” behaviour. That is, although tracking is overall stable, the origin may sometimes be misassociated and placed at a new position as the tracking gets lost. To detect this, the measurements may be monitored for sudden changes in position. If a teleportation is detected, a reinitialization would be needed, either performing a new initial estimation, or utilizing the previous state to recognize the new pose of the origin. The teleporting behaviour is detected using simple thresholding, as in Eq. (2.61), although no action is currently implemented to recover. In Eq. (2.61), the estimated motion is scaled by the factor  $s$  from the tranformation from PTAM coordinates, and thresholded by a configurable parameter  $\epsilon$ .

$$|X_t^{\text{PTAM}} - X_{t-1}^{\text{PTAM}}| \cdot s > \epsilon \quad (2.61)$$



# Chapter 3

## Non-linear Control

To control a quadrotor's movements, a non-linear controller is applied to the physical system, using a model of the system to calculate the best (in a sense well defined in this chapter) signals of control to each of the engines driving the propellers.

The controller approach chosen in this thesis is based on the Linear Quadratic (LQ) controller, the theory of which is presented in Section 3.1. An extension to the technique of *gain-scheduling* is discussed in Section 3.2.

The physical model of the system was derived in Section 2.3. In Section 3.3, this is further developed and adapted for compatibility as a model for the controller.

### 3.1 The Linear Quadratic Controller

In this section, the theory of Linear Quadratic control is presented, considering the continuous linear plant in (3.1), with control signal  $u$  and reference  $r$ .

$$\dot{x} = Ax + Bu \tag{3.1a}$$

$$z = Mx \tag{3.1b}$$

$$e = z - r \tag{3.1c}$$

The basic LQ controller, described in e.g. [11], uses a linear state-space system model and weights on the states ( $Q$ ) and control signals ( $R$ ) respectively to calculate the control signals that would - given a starting state, a motion model and a constant reference - minimize the integral (3.2).

$$\mathcal{J} = \int_0^{\infty} e^T(t)Qe(t) + u^T(t)Ru(t)dt. \tag{3.2}$$

Thus, by varying the elements of the cost matrices  $Q$  and  $R$  respectively, the solution to the optimization will yield control signals that will steer the system in a fashion that the amplitude of the control signals and the errors are balanced.

By e.g. increasing the costs of the control signals, the system LQ controller will issue smaller control signals, protecting the engines but slowing the system down.

In the linear case, (3.2) can be solved analytically, resulting in a linear feedback,

$$u_t = -L\hat{x}_t \quad (3.3)$$

$$L = R^{-1}B^TS, \quad (3.4)$$

where  $S$  is the Positively Semi-Definite (PSD) solution to the Continuous Algebraic Riccati Equation (CARE)[11], stated in (3.5).

$$A^TS + SA + M^TQM - SBR^{-1}B^TS = 0 \quad (3.5)$$

To improve the reference following abilities of the controller, the reference may be brought into the control signal by a scaling matrix  $L_r$ , which is chosen so that the static gain of the system is equal to identity [11]. In the case of equal number of control signals as controlled states, the following result is obtained;

$$u_t = -L\hat{x}_t + L_r r_t \quad (3.6)$$

$$L_r = [M(BL - A)^{-1}B]^{-1}. \quad (3.7)$$

## 3.2 State-Dependent Riccati Equations and LQ Gain Scheduling

Even though any system could be described at any point by its linearization, the linear nature of the LQ control poses a limitation in that a general system such as the one studied in this thesis - a quadrotor - will sooner or later leave the vicinity of the linearization point and no longer adhere to the physical circumstances valid there.

This will lead to sub-optimal control and possibly even to system failure. A common approach in non-linear control is to switch between pre-calculated control gains which has been calculated for selected linearization points.

The approach used in this thesis is closely related to gain scheduling, but instead of using pre-calculated gains, the linearization is done in-flight.

The problem of solving of the Riccati equation on-line is treated under the subject of *State-Dependent Riccati Equations*, SDRE's. The basic formulation of the problem is covered in e.g. [33], and an extensive survey is presented in [5]. Implementation details are detailed and evaluated in e.g [10, 1, 34].

The LQ theory is extended to the non-linear case using the Taylor expansion of a general motion model is exploited to form the general result of Equation (3.9). The linearization of the motion model is presented in Equation (3.8).

$$\dot{x} = f(x, u) \approx \underbrace{f(x_0, u_0) + \frac{\partial f}{\partial x} \bigg|_{\substack{x=x_0 \\ u=u_0}} \underbrace{(x-x_0)}_{\Delta x}}_A + \underbrace{\frac{\partial f}{\partial u} \bigg|_{\substack{x=x_0 \\ u=u_0}} \underbrace{(u-u_0)}_{\Delta u}}_B \quad (3.8)$$

In the standard formulation of LQ, the linearization is made around a stationary point  $(x_0, u_0)$ , where  $f(x_0, u_0) = 0$ . In a more general formulation, it is possible to lift this constraint using a homogenous state [33];

$$\dot{X} = \begin{bmatrix} \dot{x} \\ 0 \end{bmatrix} = \begin{bmatrix} A & f(x_0, u_0) - Ax_0 \\ 0 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} x \\ 1 \end{bmatrix}}_X + \begin{bmatrix} B \\ 0 \end{bmatrix} \Delta u. \quad (3.9)$$

Equation 3.9 is a linear system for which the ordinary LQ problem can be solved, using eq. (3.10)-(3.7).

The linearized output signal,  $\Delta u$ , is then added to  $u_0$  to form the controller output, as in Equation (3.10).

$$u = u_0 + \Delta u = u_0 - L\bar{X} + L_r r \quad (3.10)$$

However, the linearizing extension of the affine controller in (3.9) introduces a non-controllable constant state, with an associated eigenvalue inherently located in the origin. This poses a problem to the traditional solvers of the Riccati equation, which expects negative eigenvalues to solve the problem numerically, even though the weights of (3.2) theoretically could be chosen to attain a well defined bounded integral.

The problem is circumvented by adding slow dynamics to the theoretically constant state, effectively nudging the eigenvalue to the left of the imaginary axis to retain stability. This guarantees that (3.2) tends to zero.

Equation (3.9) is thus really implemented as in (3.11).

$$\dot{X} = \begin{bmatrix} \dot{x} \\ 0 \end{bmatrix} = \begin{bmatrix} A & f(x_0, u_0) - Ax_0 \\ 0 & -\mathbf{10}^{-9} \end{bmatrix} \underbrace{\begin{bmatrix} x \\ 1 \end{bmatrix}}_X + \begin{bmatrix} B \\ 0 \end{bmatrix} \Delta u. \quad (3.11)$$

### 3.3 Control Model

In Chapter 2, a physical model of the system was derived. To incorporate the information from the physical model into the governing control law, the model needs to be fitted into eq. (3.9) by providing the Jacobi matrices with regards to  $x$  and  $u$ , and removing states which will not be used in the controller.

The Jacobians of the system are aquired numerically by using central difference

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \quad (3.12)$$

While all states are of importance to the dynamics of the quadrotor, it should be noted that only a subset of the states in  $x$  is used for control. We thus need to form new matrices containing the relevant states.

### Notation

$\bar{x}$  denotes the rows in  $x$  that are used for control.

$\bar{x}^\dagger$  is used to denote the rows in  $x$  that are *not* used for control.

$\bar{A} = [\bar{A}^\square \bar{A}^\dagger]$  defines the separation of the square part ( $\square$ ) of  $\bar{A}$  with columns associated with the controller states, from the other columns ( $^\dagger$ ).

The new matrices need to be formed only with the rows that are to be used for control. Extracting those rows from Equation (3.8) yields Equation (3.13).

$$\begin{aligned} \dot{\bar{x}} = \bar{f}(x, u) &\approx \bar{f}(x_0, u_0) + \bar{A}(x - x_0) + \bar{B}\Delta u \\ &= \bar{f}(x_0, u_0) - \bar{A}^\square \bar{x}_0 - \bar{A}^\dagger \bar{x}_0^\dagger + \bar{A}^\square \bar{x} + \bar{A}^\dagger \bar{x}^\dagger + \bar{B}\Delta u \\ &= \bar{f}(x_0, u_0) - \bar{A}^\square \bar{x}_0 + \bar{A}^\square \bar{x} + \bar{B}\Delta u \end{aligned} \quad (3.13)$$

In the last equality of eq. (3.13), it is assumed that the states not described in the controller are invariant of control and time, giving  $\bar{x}_0^\dagger = \bar{x}^\dagger$ . The results of Equation (3.13) can be directly fitted into Equation (3.9) to form the new matrices for the controller. Note also that the derivation in (3.13) is completely analogous in the discrete-time case.



# Chapter 4

## Monocular SLAM

In the interest of extracting positioning information from a video stream of a general, unknown, environment, the common approach is to use SLAM, *Simultaneous Localisation And Mapping*. In the mathematical SLAM framework, features are identified and tracked throughout time in the video stream, and a filtering solution is then traditionally applied to estimate the probability densities of the tracked landmarks' positions, as well as that of the camera position.

To determine the depth distance to a feature, stereo vision is often applied with which the correlation between two synchronized images is used together with the known distance between the cameras to calculate the image depth. As the distance to the tracked objects increase however, the stereo effect is lost and the algorithms' performance drops. There are several other issues to the stereo vision approach - increased cost, synchronization issues, computational load to name a few - which has led to the development of techniques to utilize the information on movement in only a single camera to calculate the depth of the features in the image. The application of SLAM to this approach is referred to as *Monocular SLAM*, and two approaches are presented in this chapter.

Both approaches rely on feature detection in video frames. An extensive survey of available feature-detecting algorithms is given in [14].

### 4.1 Filtering Based Solutions

One novel approach for camera tracking, used by for instance [7], is to use the EKF-SLAM framework and couple the feature and camera tracking in a time- and measurement update for each consecutive frame ([9] does this with FastSLAM 2.0). The *Scenelib* library described in [7] uses a combination of the particle filter and the EKF for feature initialization and feature tracking respectively.

Common to the filter approaches is that as new features are detected, they are initialized and appended as states to the filter. As frames arrives, the change of position of each feature is measured and the filter is updated accordingly. Thus, one must take care to avoid misassociation of the features, as this leads to false measurements that will mislead the filter.

It is trivial to include motion models into the filtering approaches, since the general algorithm utilize a classical state-based filter time-update.

## 4.2 Keyframe-based SLAM

A fundamentally different approach is presented in [20], where the focus is put on collecting video frames of greater importance - keyframes - in which a large amount of features are detected. The camera's position is recorded at the time the keyframe is grabbed - see Figure 4.1 - and the new features detected are added to the active set. As new video frames arrive, features are reprojected and sought for in the new frame, giving a position offset from recorded keyframes which by extension gives the updated position of the camera.



Figure 4.1.

## 4.3 The PTAM Library

PTAM - Parallel Tracking And Mapping is a software library for camera tracking developed for Augmented Reality (AR) applications in small workspaces [20]. It has only recently been applied for quadrotor state estimation [39], although as the library is intended for AR applications, the connection with the world's coordinate system is loose, as duly discussed in Chapter 2.4.5. Several libraries exist extending the functionality of the PTAM library [? ].

As recently published, new algorithms - presented in [? ] - surpass the library's performance at the expense of computational cost and the demand for a high-profile graphics card. The performance is nonetheless proven<sup>1</sup>, and improves the usability over preceding libraries, such as *Scenelib*, by including a camera calibration tool, estimating parameters describing the lens properties.



Figure 4.2.

---

<sup>1</sup>Examples of projects using PTAM are listed at <http://ewokrampage.wordpress.com/projects-using-ptam/>

### 4.3.1 Operation

In the tracking procedure, the PTAM library randomly projects previously recorded features into the captured video frame, visualized in Figure 4.3. It could be argued that the feature projection in the PTAM library could make more use of external sensors and position estimates in this process. This, however, remains as future work.



Figure 4.3.

As for mapping; Instead of continuously updating the keyframe's position estimates - as in the filtering solution - the PTAM library presented in [20] separates this task from the tracking entirely to a parallel, less time-critical, processing thread<sup>2</sup>. As this update does not have to be performed in real-time each video-frame, this parallelization allows for the use of a more advanced adjustment technique [20], namely Bundle Adjustment[25]. The bundle adjustment problem performs a non-linear least-squares optimization over the available keyframe positions, utilizing the sparse structure of the optimization problem that is solved to make the solution computationally tractable [24].

### 4.3.2 Modifications to the Library

The PTAM library is originally interfaced with a included graphical interface. However, as the source-code is provided for non-commercial use, the library was modified to interface over serial port with the CRAP framework and also extended with automatic initialization and full non-graphical use. This to enable full automatization on the development platform.

In the standard PTAM initialization procedure, an image is captured at a starting position. The camera is then translated sideways until the user deems the stereo initialization can be performed with good results. A second frame is then captured, and features from the original image are tracked between the two to perform a stereo initialization to extract the ground plane and a placement for the origin.

---

<sup>2</sup>Hence its name; *Parallell Tracking And Mapping*



Figure 4.4.

In the graphical interface, the tracked features are visualized during the process with lines in the camera image, as in Figure 4.4. After the initial frame has been captured, the implementation of the automated initialization uses the length of these lines as a measure of when to finalize the initialization procedure. When the length of the line associated with the first<sup>3</sup> feature exceeds a given threshold, the second frame is captured and the stereo initialization algorithm is started. Should the initialization procedure fail - e.g. by losing track of most features - the procedure will be restarted until a successful initialization has been performed. The procedure is initially triggered by a command from the serial port.

### 4.3.3 Practical Use

The PTAM library is designed to use a wide-angle lens, with performance dropping should such not be available [20]. In its current state however, PTAM does not support fish-eye lenses, and although the LinkQuad-mounted camera features a changeable lens, all available wide-angle lenses are fish-eye. It is possible to use the PTAM library with a standard lens, although tracking is easier lost.

The tracking and initialization quality is also - as all video tracking - dependent on the amount of trackable features in the scene. While the library can recover from lost positioning, an insufficient amount of features can cause the initialization process to fail, or the tracking to be irreparably lost. The amount of features tracked is also dependent on the processing power available, and because the complexity of the mapping problem grows fast with the number of keyframes, the library may have difficulties extending the map to unexplored areas.

---

<sup>3</sup>First in the list of features from the first image still found in the latest frame.

# Chapter 5

## Finite State-Machines

In projects related to the LinkQuad quadrotor, generic state-machine frameworks have been developed and researched[27, 42]. On the LinkQuad, for the purpose of this thesis, a stripped down state-machine engine was implemented in the *CRAP* framework presented in Appendix A.

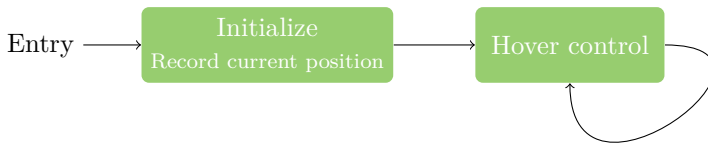
The state machine is responsible for transitioning between the states - action modes - predefined in an action sequence, ultimately providing reference signals for the controller.

### 5.1 Implemented Modes

In this Section, four basic modes are presented which were implemented in the timeframe of this thesis.

#### 5.1.1 Hovering

In the hover mode, the position of the quadrotor is noted at the time of the activation, and three independent PID-controllers are initialized to generate reference signals to the main controller, working to keep the quadrotor at the place where it was first initialized.



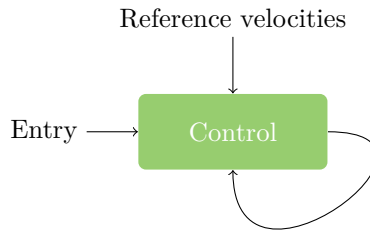
**Figure 5.1.** Hovering scheme.

### 5.1.2 PTAM initialization

Entering this mode starts an automated initialization process to set up the initial PTAM coordinate system. Commands are sent to the PTAM module to initialize tracking, after which the quadrotor should be moved sideways for the stereo initialization performed by the PTAM library.

### 5.1.3 Free Flight

In the free flight mode, the control reference is forwarded from the joystick reference provided over the serial interface from the ground station.



**Figure 5.2.** Freeflight scheme

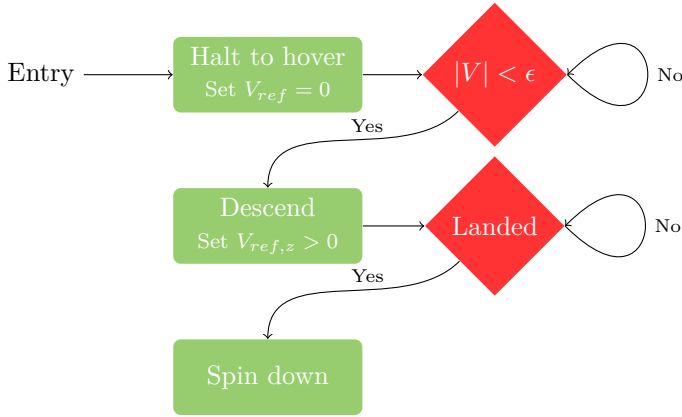
### 5.1.4 Landing

There have been several studies of autonomous quadrotor landing, in e.g.[26, 4]. [4] implements a landing scheme closely related to that which is proposed in this thesis. The algorithm used can be summarized in the following steps:

- Detection,
- Refinement,
- Descent,
- Landig detection.

In the detection phase, the environment is searched for a suitable landing place. Landing is then performed on an elevated surface which is detected using video processing. After the landing area has been located, the position of the landing site - relative to the quadrotor - is filtered to increase the certainty of the positioning.

While the filtered position converges, the quadrotor is moved to a position above the landing site as preparation for the descent phase, where the quadrotor lowers until landing has been detected, using the camera feedback and other sensors to stabilize the descent.



**Figure 5.3.** Landing scheme.

### Landing Detection

Since we recognize that our estimated position - with origin at our starting point - is not necessarily consistent with the Height Over Ground (HOG) at the landing site, it is necessary to choose a more robust method to detect the completion of the landing procedure than simply halting at zero height. The approach is to use the measurements to determine when movement has stopped; that is, when the quadrotor has reached ground. Detection theory, as discussed in e.g. [36, 28], provides several tools for detecting the non-linear event that the quadrotor can descend no further.

In the physical model presented in Chapter 2, two terms are of specific interest for the detection. The first - and the obvious - is the velocity in the gravity-aligned  $z$ -axis. When sensor measurements pull this term towards zero, this is a first indication that the quadrotor has stopped. When the sensor measurements indicate a halt, the observer - oblivious to the forces imposed by the ground contact - will explain the lack of movement by a drastic increase in the estimated wind acting in the  $z$ -direction. This observer state - the second of interest - is easily monitored and could e.g. be filtered using the Cumulative Sum (CUSUM) algorithm to increase detection confidence, or simply thresholded to detect landing.





# Chapter 6

## Results

This chapter contains results evaluating the theory of the previous chapters. The model is verified against data collected on the LinkQuad, as well as ground truth from the Vicon motion tracking system available. Ground truth was used only for initializing the model, and all test were run off-line on recorded sensor data and video feed.

### 6.1 Experiment Setup

The data used in this Chapter was recorded on the LinkQuad quadrotor in the Witas Vicon Lab at Linköping University, Sweden. Ground truth data was recorded using the Vicon tracking system at a rate of  $10\text{ Hz}$ , while sensors were sampled at  $500\text{ Hz}$  and logged on-board the LinkQuad. For the dataset used in this Chapter, where not noted otherwise, Camera data was collected at a rate of  $30\text{ Hz}$  during 20-second bursts after which the data had to be written to memory. The camera was tilted approximately 30 degrees downwards from the horizontal body-fixed plane of the quadrotor, giving an overview of the cluttered floor in Figure 6.1. The camera settings were tuned to minimize the disturbance from lightsources and the infrared light used by the Vicon system.



Figure 6.1.

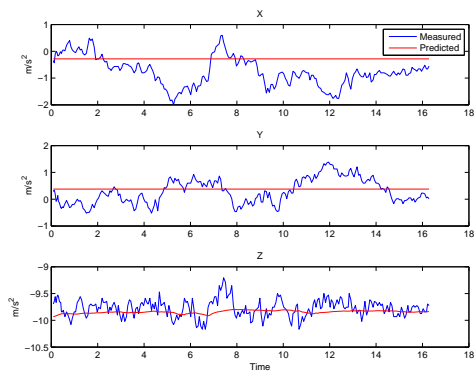
## 6.2 Modelling

The verification of a complex model is best done in small parts. It is however, with the model given in Chapter 2, difficult to evaluate each equation by itself due to the couplings of the model. Instead, the verification is performed by evaluating a full test-flight with recorded data, using one dataset for calibration and a second for validation.

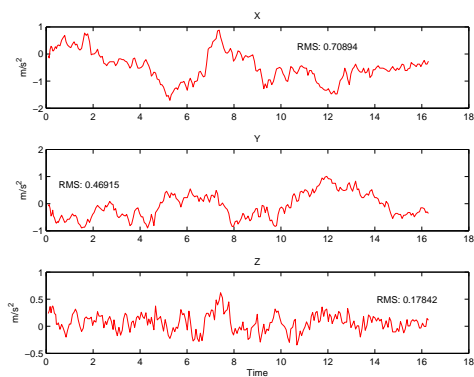
For each sensor the predicted and the measured values are compared, and the residuals - the difference between the two - are studied and fitted to a normal probability density function, *PDF*.

### 6.2.1 Accelerometers

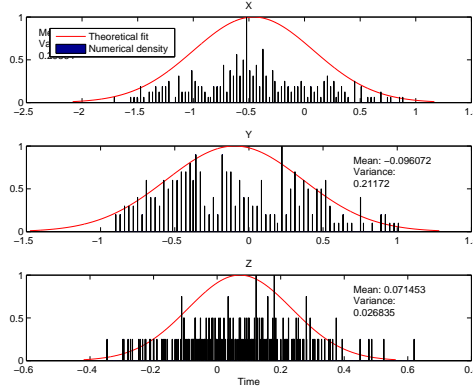
As most of the modelling of Chapter 2 concerns the forces acting upon the quadrotor, the accelerometers provide an interesting measure of the quality of the model. It should be noted that no parameter tuning was performed on the motion model, yielding the results to be merely directional. As depicted in Figure 6.2, the model does leave clearly trended residuals, not least in the X- and Y-directions where the model does very little. From the Figures 6.3 and 6.4, it can be seen that the model does have beneficial effects for the estimation, yielding residuals with zero-close means.



**Figure 6.2.** Measured and predicted accelerations in the NEDEF system.



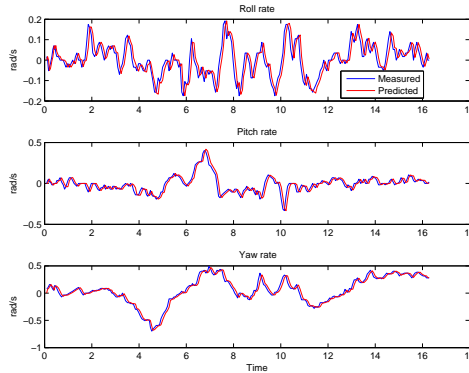
**Figure 6.3.** Residuals between measured and predicted accelerations



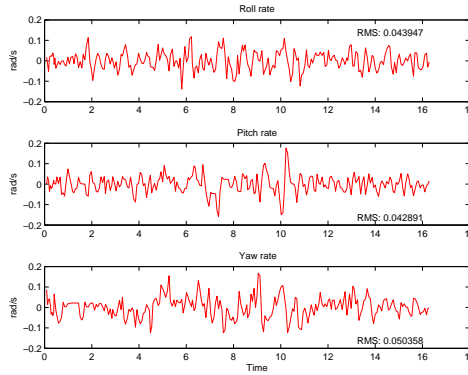
**Figure 6.4.** Accelerometer residuals fitted to a normal PDF. Both theoretical and numerical values have been normalized to a maximum height of one.

## 6.2.2 Gyroscopes

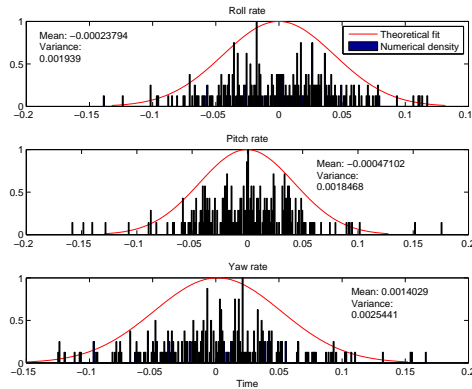
It is clear, from Figure 6.5, that the model well describes the angular velocity of the quadrotor. The residuals, described by Figures 6.6 and 6.7 displays a behaviour which is seemingly well described by a random, normally distributed, variable, which is expected from the standard Kalman filter framework.



**Figure 6.5.** Measured and predicted angular velocities, in the body-fixed coordinate frame.



**Figure 6.6.** Residuals between measured and predicted angular velocities



**Figure 6.7.** Gyroscope residuals fitted to a normal PDF. Both theoretical and numerical values have been normalized to a maximum height of one.

### 6.2.3 Pressure Sensor

The pressure sensor is, as clearly seen in Figure 6.8, associated with a great amount of noise. While the residuals, Figures 6.9 and 6.10, does not exhibit any obvious trends, noise does spill into the positioning with the current tuning, currently adding little contribution to the state estimation. This is however likely to be improved by firther filter tuning.

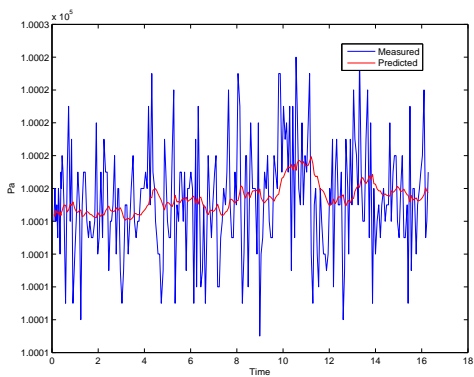


Figure 6.8. Measured and predicted pressure.

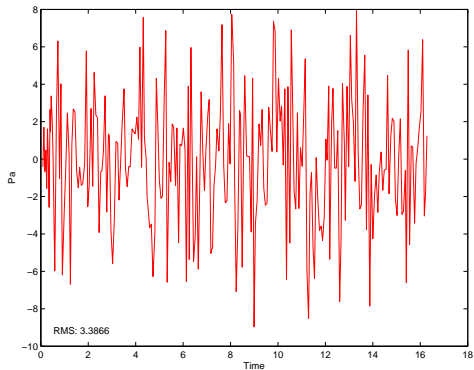
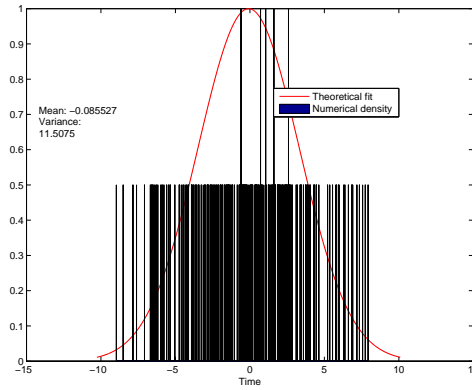


Figure 6.9. Residuals between measured and predicted pressure

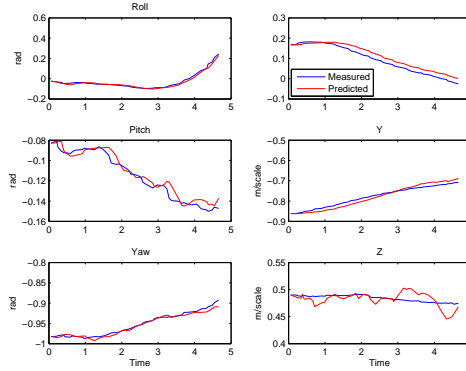


**Figure 6.10.** Pressure residuals fitted to a normal PDF. Both theoretical and numerical values have been normalized to a maximum height of one.

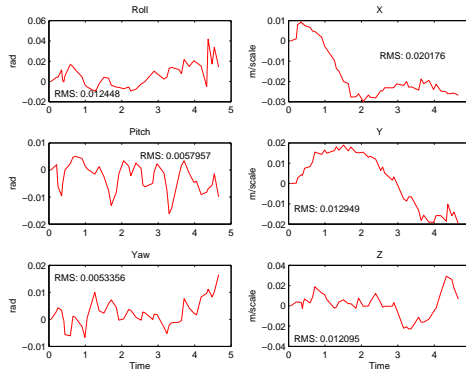
### 6.2.4 Camera

The camera tracking, displayed in Figures 6.11-6.13, exhibit very good stability and performance, and significantly add to the filter performance. The absolute positioning provided by the camera does not only counter the drift in derived observer states, but also, not least through its accurate measurements of orientation angles exhibited in Figure 6.11. It should be noted that the results, especially in Figures 6.12-6.13 should be scaled approximately by a factor of three to correspond to metric quantities.

When the PTAM library is initialized, it tries to determine the ground plane. Thus, we are able to verify, in Figure ??, the initialization process and transformation by confirming that the Z-axes are approximately parallel. The slight tilting observed in Figure ?? is caused by a misplacement of the ground plane in the initialization process. Knowing the pose at the initialization allows us to compensate for this in the transformation, making the positioning less sensitive for PTAM initialization errors. Exact positioning on the moment of initialization is still of importance, however.

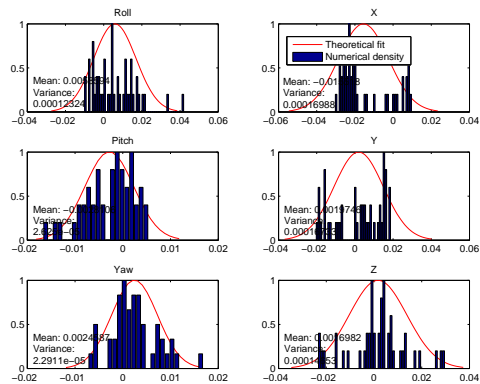


**Figure 6.11.** Measured and predicted angles and positions, in the PTAM coordinate frame.



**Figure 6.12.** Residuals between measured and predicted angles and positions

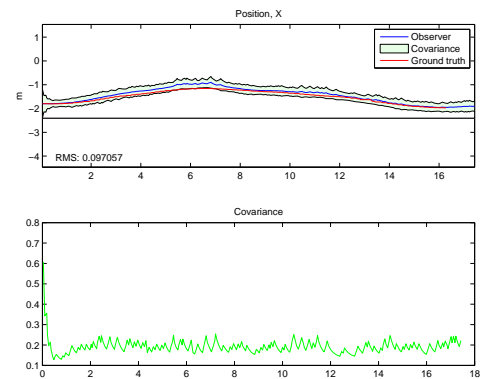




**Figure 6.13.** Residuals fitted to a normal PDF. Both theoretical and numerical values have been normalized to a maximum height of one.

### 6.3 Filtering

**Positioning** While the altitude positioning, plotted in Figure 6.16, exhibit disturbances correlated with pressure sensor noise, the positioning exhibit very good performance. The position state is observed in the state-estimation more or less directly by the camera and the stability of the camera positioning is thus of course reflected here.



**Figure 6.14.**

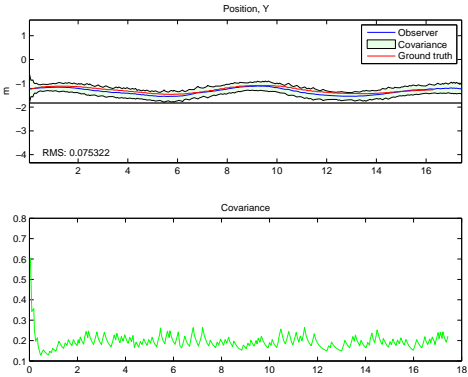


Figure 6.15.

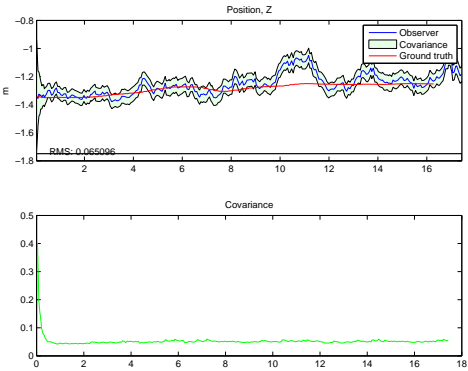


Figure 6.16.

**Velocities** The velocities, being closely coupled with the camera observed position, also exhibit good performance in Figures 6.17-6.19. There are shortcomings to the estimation's horizontal precision, although this could probably be significantly improved with further filter tuning.

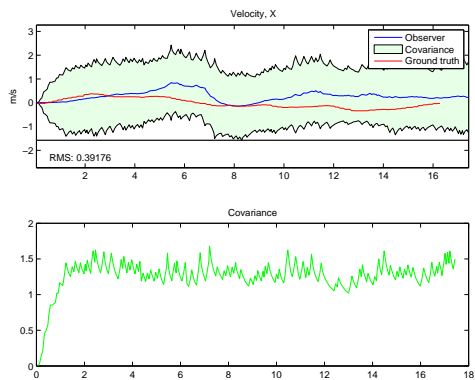


Figure 6.17.

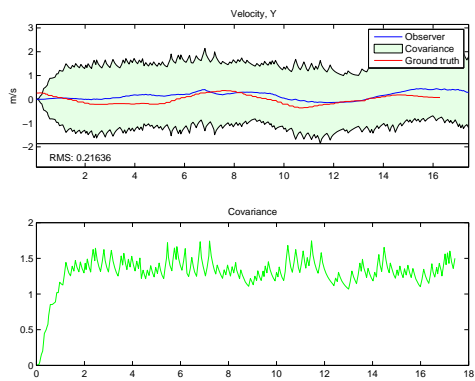


Figure 6.18.

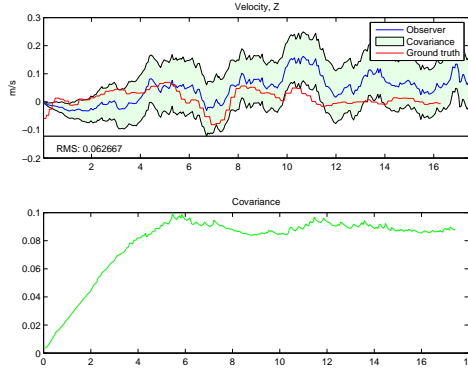


Figure 6.19.

**Orientation, Rotational Velocity and Gyroscope Offset** Along with the position, the orientation is estimated from the camera, yielding notable precision, as seen in Figures 6.20-6.23.

The bias of the gyroscopes is removed during the initialization process. Since the time-frame of the tests were far less than the time expected to detect a change in the bias, these should thus be estimated to zero. As depicted in Figures 6.27-6.29, they are.

As noted in Section 6.2.2, the filtering of the rotational velocities of the quadrotor body, exhibited with its associated covariance in Figures 6.24-6.26, correlates very well to the measured results.

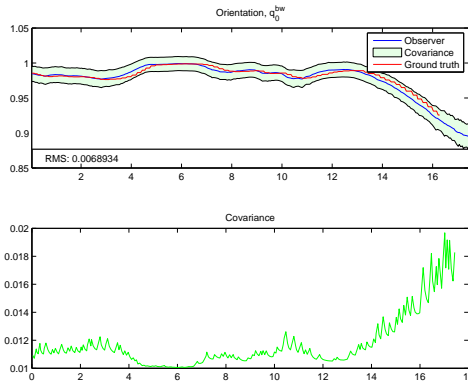


Figure 6.20.

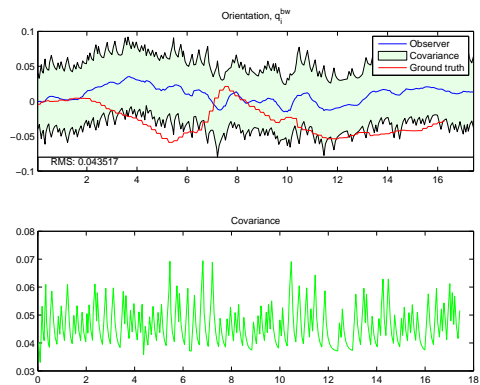


Figure 6.21.

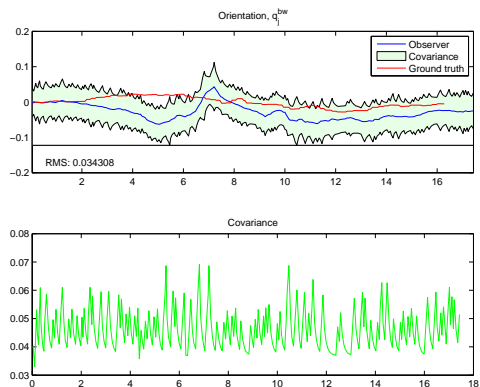


Figure 6.22.

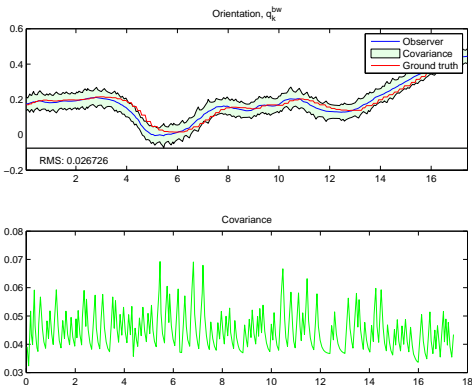


Figure 6.23.

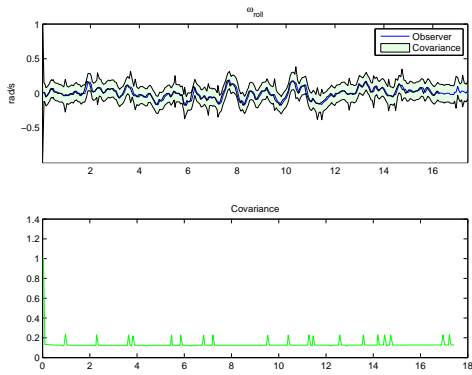


Figure 6.24.

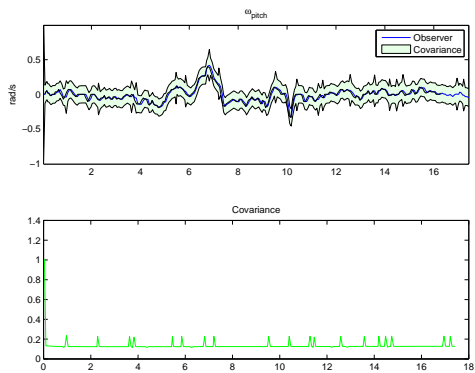


Figure 6.25.

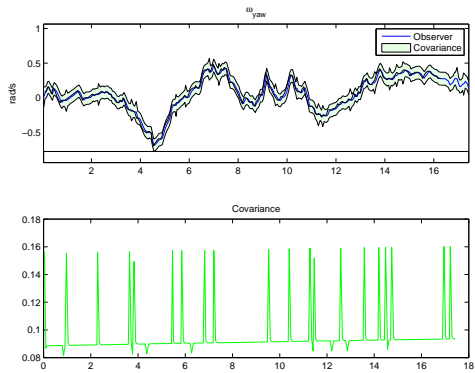


Figure 6.26.

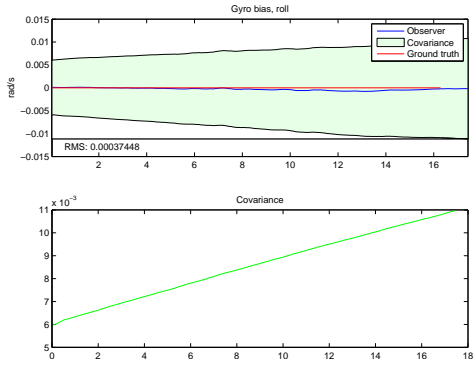


Figure 6.27.

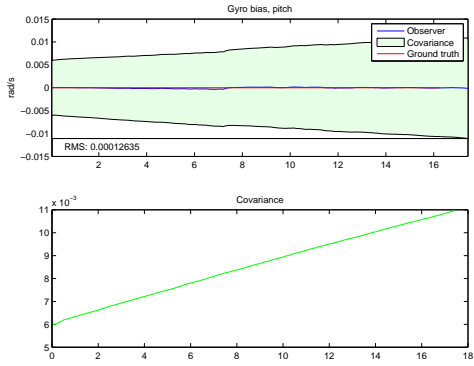


Figure 6.28.

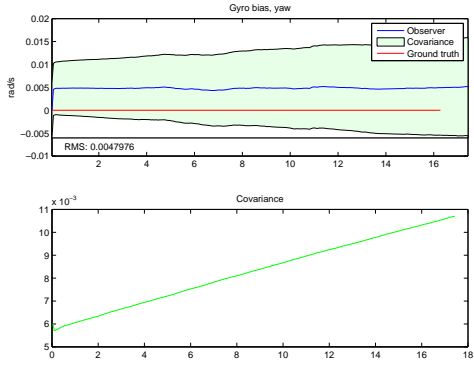


Figure 6.29.



**Wind force** As the tests were performed inside, the filter was tuned to basically keep the wind constant. Thus, it is difficult to come to any conclusions regarding the wind impact on the model. They are however, as seen in Figures 6.30-6.31, correctly estimated to zero.

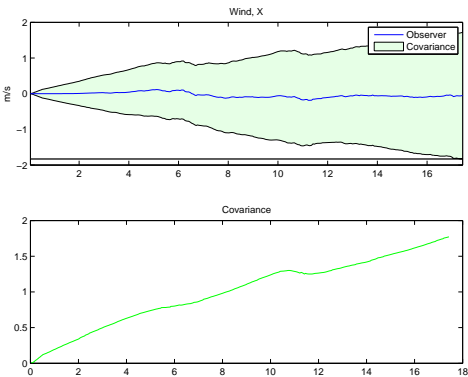


Figure 6.30.

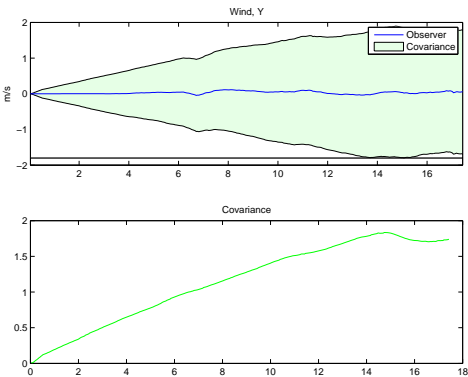


Figure 6.31.

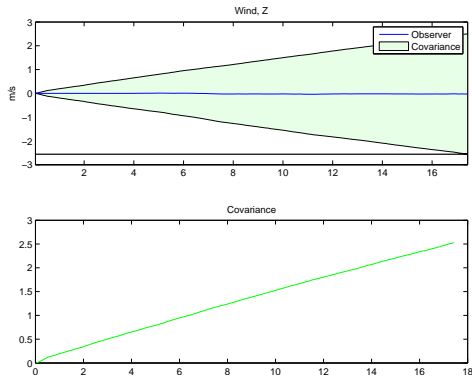


Figure 6.32.

**Propeller Velocity** As the filter evaluation was performed without the use of the controller, the control signal is unavailable. Thus, Eq. 2.37 was used as motion model, effectively leaving the estimation of the propeller velocities to the measurement update. It is evident, in Figures 6.33-6.36 that the estimation is active, however it is impossible to validate. Ideally, the velocities of the propellers should be measured in flight. However, that data is currently unavailable in the development system used for evaluation. The estimated velocities are, notably, in a reasonable range, increasing the plausability for correctness of Eq. 2.38a, which is otherwise hard to verify using the available data.

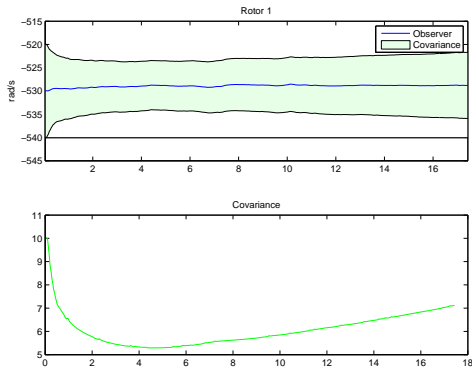


Figure 6.33.

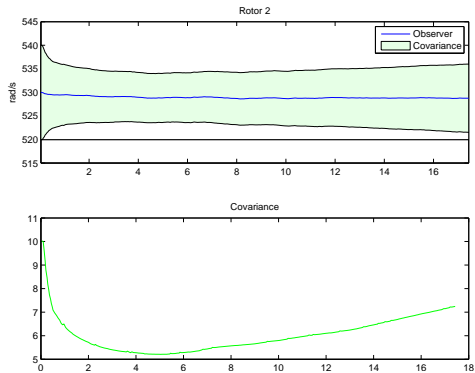


Figure 6.34.

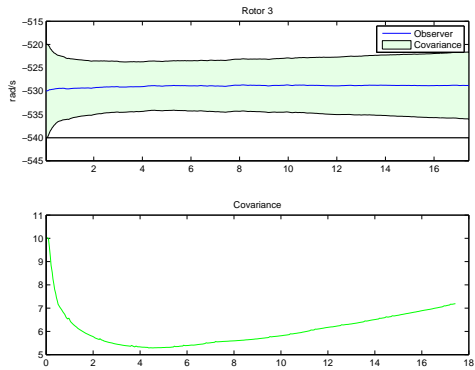


Figure 6.35.

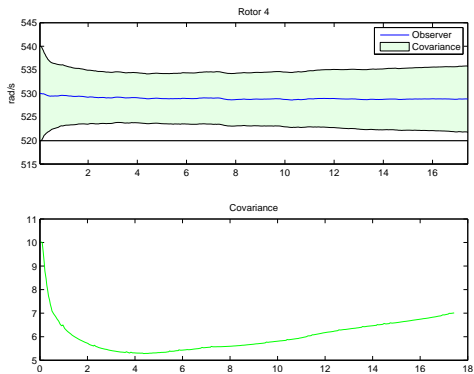


Figure 6.36.

## 6.4 Control

For the control evaluation,

# Chapter 7

## Discussion

In this chapter the results of the thesis are discussed. Specific attention is given to restrictions and their impact on system performance, as well as providing a basis for later discussions on further work.

### 7.1 Flapping



## Chapter 8

# Concluding Remarks

### 8.1 Conclusions

### 8.2 Further work





# Bibliography

- [1] Peter Benner. Accelerating Newton's Method for Discrete-Time Algebraic Riccati Equations. In *In Proc. MTNS 98*, pages 569–572, 1998.
- [2] Michael Blösch, Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Vision based MAV navigation in unknown and unstructured environments. In *ICRA*, pages 21–28, 2010.
- [3] Samir Bouabdallah and Roland Siegwart. Full control of a quadrotor. In *IROS'07*, pages 153–158, 2007.
- [4] Roland Brockers, Patrick Bouffard, Jeremy Ma, Larry Matthies, and Claire Tomlin. Autonomous landing and ingress of micro-air-vehicles in urban environments based on monocular vision. In Thomas George, M. Saif Islam, and Achyut K. Dutta, editors, *Micro- and Nanotechnology Sensors, Systems, and Applications III*, volume 8031, page 803111. SPIE, 2011.
- [5] Tayfun Çimen. State-Dependent Riccati Equation (SDRE) Control: A Survey. In *Proceedings of the 17th IFAC World Congress*, pages 3761–3775, 2008.
- [6] National Geophysical Data Center. The World Magnetic Model. <http://www.ngdc.noaa.gov/geomag/WMM/DoDWMM.shtml>, March 2012.
- [7] Andrew J. Davison. Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *ICCV*, pages 1403–1410. IEEE Computer Society, 2003.
- [8] Patrick Doherty and Piotr Rudol. A UAV Search and Rescue Scenario with Human Body Detection and Geolocalization, 2007.
- [9] Ethan Eade and Tom Drummond. Scalable Monocular SLAM. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, pages 469–476, Washington, DC, USA, 2006. IEEE Computer Society.
- [10] Evrin Bilge Erdem. ANALYSIS AND REAL-TIME IMPLEMENTATION OF STATE-DEPENDENT RICCATI EQUATION CONTROLLED SYSTEMS BY, 2001.

- [11] T. Glad and L. Ljung. *Reglerteori: flervariabla och olinjära metoder*. Studentlitteratur, 2003.
- [12] F. Gustafsson. *Statistical Sensor Fusion*. Utbildningshuset/Studentlitteratur, 2010.
- [13] Masayuki Hayashi et al. A Study of Camera Tracking Evaluation on TrakMark Data-Set. Technical report, University of Tsukuba, 2011.
- [14] M.Y.I. Idris, H. Arof, E.M. Tamil, N.M. Noor, and Z. Razak. Review of Feature Detection Techniques for Simultaneous Localization and Mapping and System on Chip Approach. *Information Technology Journal*, 8:250–262, 2009.
- [15] Simon J. Julier and Idak Industries. The scaled unscented transformation. In *in Proc. IEEE Amer. Control Conf*, pages 4555–4559, 2002.
- [16] Simon J. Julier and Jeffrey K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, pages 401–422, 2004.
- [17] Simon J | Uhlmann Jeffrey K | Durrant-Whyte Hugh F Julier. A new approach for filtering nonlinear systems. In *1995 American Control Conference, 14th, Seattle, WA; UNITED STATES; 21-23 June 1995*, pages 1628–1632, 1995.
- [18] Kalman, Rudolph, and Emil. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.
- [19] Niklas Karlsson, Enrico Di Bernardo, Jim Ostrowski, Luis Goncalves, Paolo Pirjanian, and Mario E. Munich. The vSLAM algorithm for robust localization and mapping. In *In Proc. of Int. Conf. on Robotics and Automation (ICRA)*, pages 24–29, 2005.
- [20] Georg Klein and David Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR’07)*, Nara, Japan, November 2007.
- [21] Georg Klein and David Murray. Parallel Tracking and Mapping on a Camera Phone. In *Proc. Eighth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR’09)*, Orlando, October 2009.
- [22] J.B. Kuipers. *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*. Princeton paperbacks. Princeton University Press, 2002.
- [23] J.G. Leishman. *Principles of helicopter aerodynamics*. Cambridge aerospace series. Cambridge University Press, 2002.
- [24] Manolis I.A. Lourakis, editor. *Bundle adjustment gone public*, 10 2011.
- [25] M.I. A. Lourakis and A.A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009.

- [26] D. Mellinger, M. Shomin, and V. Kumar. Control of Quadrotors for Robust Perching and Landing. In *Proceedings of the International Powered Lift Conference*, Oct 2010.
- [27] Torsten Merz, Piotr Rudol, and Mariusz Wzorek. Control System Framework for Autonomous Robots Based on Extended State Machines. In *ICAS 2006 - International Conference on Autonomic and Autonomous Systems, 2006*, 2006.
- [28] Mattias Nyberg and Erik Frisk. *Model Based Diagnosis of Technical Processes*. LiU-tryck, 2011.
- [29] United States. Dept. of Defense. Office of the Secretary of Defense. *U.S. Army Roadmap for unmanned aircraft systems, 2010-2035*. U. S. Army UAS Center of Excellence, 2010.
- [30] Helen Oleynikova. ROS vSLAM, September 2011. [online] <http://www.ros.org/wiki/vslam>.
- [31] Paul Pounds, Robert Mahony, and Peter Corke. Modelling and Control of a Quad-Rotor Robot.
- [32] R.W. Prouty. *Helicopter performance, stability, and control*. Krieger Pub., 1995.
- [33] A. Rantzer and M. Johansson. Piecewise Linear Quadratic Optimal Control, 1999.
- [34] V. Sima and P. Benner. A SLICOT Implementation of a Modified Newton's Method for Algebraic Riccati Equations. *Mediterranean Conference on Control and Automation*, 0:1–6, 2006.
- [35] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. Real-time monocular SLAM: Why filter? In *ICRA*, pages 2657–2664, 2010.
- [36] David Törnqvist. *Estimation and Detection with Applications to Navigation*. PhD thesis, Linköping University, 2008.
- [37] K. Valavanis. *Advances in unmanned aerial vehicles: state of the art and the road to autonomy*. International series on intelligent systems, control, and automation. Springer, 2007.
- [38] Rudolph van der Merwe, Arnaud Doucet, Nando de Freitas, and Eric A. Wan. The Unscented Particle Filter. In *NIPS'00*, pages 584–590, 2000.
- [39] S Weiss, D Scaramuzza, and R Siegwart. Monocular-SLAM based navigation for autonomous micro helicopters in GPS-denied environments. *Journal of Field Robotics*, 28(6):854–874, 2011.
- [40] Wikipedia. Atmospheric Pressure. *Wikipedia*, April 2012.
- [41] K C Wong and C Bil. UAVs OVER AUSTRALIA - Market And Capabilities. *Flight International*, pages 1–16, 2006.

- [42] Mariusz Wzorek. Selected Aspects of Navigation and Path Planning in Unmanned Aircraft Systems, 2011.

# Appendix A

## CRAP

For the implementation of the theory presented in this report, a framework was developed for connecting the different separable modules and provide a core library of useful functions. The result is called *C++ Robot Automation Platform*, or *CRAP* for short. The main ideas of *CRAP* are borrowed from the *Robot Operating System*, *ROS*<sup>1</sup>, while implementing these in a much more efficient and uniform manner. By constraining the intermodular messaging to C++<sup>2</sup> and compartmentalizing the modules in separate threads as opposed to processes, *CRAP* significantly reduces the overhead associated with the flexibility of such modular software.

### A.1

---

<sup>1</sup><http://ros.org/>

<sup>2</sup>ROS allows messaging between Python and C++ modules