Institutionen för datavetenskap
Department of Computer and Information Science

Master's Thesis

Autonomous UAV Landing Using Monocular Vision

Jonatan Olofsson

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

LINKÖPINGS UNIVERSITET

Linköping University
INSTITUTE OF TECHNOLOGY

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

Institutionen för datavetenskap
Department of Computer and Information Science

Master's Thesis

Autonomous UAV Landing Using Monocular Vision

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

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

Avdelning, Institution Division, Department Department of Computer and Information Science Department of Computer and Information Science Linköpings universitet SE-581 83 Linköping, Sweden	**Datum** Date YYYY-09-19

Språk Language ☐ Svenska/Swedish ☒ Engelska/English ☐ __________	**Rapporttyp** Report category ☐ Licentiatavhandling ☒ Examensarbete ☐ C-uppsats ☐ D-uppsats ☐ Övrig rapport ☐ __________	**ISBN** — **ISRN** LiTH-ISY-EX--YY/XXXX--SE **Serietitel och serienummer** Title of series, numbering **ISSN** —

URL för elektronisk version

http://www.ida.liu.se/divisions/aiics/

http://www.ida.liu.se/divisions/aiics/

Titel
Title Autonomous UAV Landing Using Monocular Vision

Författare Jonatan Olofsson
Author

Sammanfattning
Abstract

This thesis treats the subject of landing a quadrocopter autonomously.

Traditionally this, and most advanced UAV control, has been 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 SLAM.

Nyckelord
Keywords automated landing, slam, ptam, control, uav

Abstract

This thesis treats the subject of landing a quadrocopter autonomously.

Traditionally this, and most advanced UAV control, has been 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 SLAM.

Sammanfattning

Svenskt abstract kan man placera här.

Acknowledgments

Till mor och far

Contents

Chapter 1

Introduction

The goal for this thesis is to develop and implement an autonomous landing mode for the quadrotor developed by the AIICS team at Linköping University, the LinkQuad.

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.

The limitations, however, do not stop 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. This estimation can be efficiently detached from the core control and modularized as an independent sensor, which will be exploited in the thesis.

1.1 Unmanned Aerial Vehicles

As noted in [37], Unmanned Aerial Vehicles (UAV's) have been imagined and constructed for millennia, starting in ancient Greece and China. 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 continue to lead the development in the field[29], recent years have seen a great increase in domestic and civilian applications[41]. These 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). SUAV's are designed to be man-portable in weight and size and is thus limited in payload and availabe processing power. This limitation, in combination with the unavailability of indoor GPS positioning, has led to extensive use of off-board positioning and control in recent research. Systems developed by for instance Vicon[1] and Qualisys[2] 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 beeing 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[3]. 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.

1.3 Previous work

Here, previous known similar work is referenced and discussed. Previous work can be used for both reference implemetations, but also to recognize limitations and restrictions posed on the systems studied in the works to extend the work further. In addition to the sources presented in this Section, several more are cited in the following chapters.

The problem of positioning an unmanned quadrotor using visual feedback is a problem which was has received extensive attention during recent years, and only recently been implemented with satisfactory results[2, 39]. Few have attempted to

[1]`http://www.vicon.com/`
[2]`http://www.qualisys.com/`
[3]`http://www.ros.org/`

use on-board sensors only, but have relied on external setups to track the motions of the quadrotors - admittedly with high precision. Fewer still have succeeded using strictly real-time algorithms running on the limited processing power that standard UAV's of the size of the LinkQuad generally are equipped with. The LinkQuad is, with its distributed processing power, well suited for a full implementation and solution to the problem.

1.3.1 Autonomous landing

The problem of landing a quadrocopter 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 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.

[4] implements a landing control reference scheme which is inspirational to the approach in this thesis, which is summarized in 5.1.4.

1.3.2 vSLAM

A first take on a 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. An implementation has been made in a ROS project[30], which may be used for reference. Similar approaches have been implemented by e.g. [7, 9].

[20] uses an alternative approach in the PTAM library it presents and splits the tracking problem of SLAM from the mapping problem. This means that the mapping can run more advanced algorithms at a slower pace than required by the tracking. In [21], a modified version of said library was implemented on an iPhone. This is of special interest to this thesis, since it demonstrates a successful implementation in a resource-constrained environment.

The algorithm proposed in [20] uses selected keyframes from which offsets are calculated continously 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.

By for instance not considering 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 the thesis is to perform autonomous landing with the LinkQuad. To achieve this, the main work has been put into achieving good, theoretically solid, positioning and control. Having control over the vehicle, landing is then a matter of generating a suitable trajectory and detecting the completion of the landing.

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 litterature on the subject.

The time restrictions - 800 hours - did in the end force restrictions on the amount of verification and testing that could be done. As very little had been done on the platform previously, much time had to be put into e.g. the generic structure and communication between the different software and hardware modules.

1.5 Contributions

During the thesis work, several tools for future development have been designed and developed. The *CRAP* framework is no doubt 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;

Observer General non-linear filtering, using EKF or UKF.

Control General non-linear control, implemented as affine quadratic control.

Logic State-machine for scheduling and reasoning.

Plotting Real-time plotting is available.

Communication Communication API for internal and external communication.

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

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.

All tools developed during the thesis are released under the GPL license.

1.6 Thesis Outline

Following the introductory Chapter 1, four chapters are devoted to presenting the theory and the equations used in the implementation. The following two chapters of the thesis, Chapters 6 and 7, present the numerical evaluation of the result and the following discussion respectively, after which concluding remarks and suggestions for further work are presented in Chapter **??**.

A detailed description of the *CRAP* framework and the implementation code is attached as appendices.

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 if of vital importance to the performance of the control. 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, the Unscented Kalman Filter (UKF) and the 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 is 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.

The motions of the system are also captured by the on-board sensors. A measurement y is related to the motion model by the sensor model h;

$$y(t) = h(x(t), u(t), t) \tag{2.1}$$

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

2.1 The Filtering Problem

The problem of estimating the state of a system - in this case it's position, orientation, velocity etc. - is in the Kalman filter framework 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. 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, which accounts for imperfections in the model compared to the physical real-world-system.

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

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 obtain the next predicted state directly, yielding the prediction of $\hat{x}$ at the time t *given* the information at time $t-1$. [1] This motivates the notation used in this thesis - $\hat{x}_{t|t-1}$.

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

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*, ν.

$$\nu_t = y_t - \hat{y}_t \tag{2.4}$$

That is, the difference between the measured value and what would be expected in the ideal (simulated) 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.

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

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

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

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 [18]*

Prediction update

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

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

[1]Note that we have not yet performed any measurements that provide information about the state at time t.

Measurement update

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

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

$$P_{t|t} = P_{t|t-1} + KHP_{t|t-1} \tag{2.8c}$$

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.

$$x \sim \mathcal{N}\left(\hat{x}, P_{xx}\right) \tag{2.9}$$

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

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

As new measurements are taken, the covariance of the state evolves [17] with the state estimate as

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

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

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

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

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;

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

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 nescessary.*

Prediction update

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

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

Measurement update

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

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

$$P_{t|t} = P_{t|t-1} + KHP_{t|t-1}, \tag{2.16c}$$

where

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

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.

$$\begin{aligned}
\mathcal{X}_\mathbf{0} &= \hat{x} \\
\mathcal{X}_\mathbf{i} &= \hat{x} + \left(\sqrt{(n+\lambda)P_{xx}}\right)_i & i &= 1, \cdots, n \\
\mathcal{X}_\mathbf{i} &= \hat{x} - \left(\sqrt{(n+\lambda)P_{xx}}\right)_i & i &= n+1, \cdots, 2n
\end{aligned} \tag{2.18}$$

Variable	Value	Description
α	$0 \leq \alpha \leq 1$ (e.g. 0.01)	Scales the size of the sigma point distribution. A small α can be used to avoid large non-local non-linearities.
β	2	As discussed in [15], β 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.
κ	0	κ is the number of times that the centerpoint 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.

$$\begin{aligned} W_0^m &= \tfrac{\lambda}{n+\lambda} \quad W_0^c = \tfrac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \\ W_i^m &= W_i^c = \tfrac{1}{2(n+\lambda)} \qquad i = 1, \cdots, 2n \end{aligned} \tag{2.19}$$

$$\lambda = \alpha^2(n + \kappa) - n \tag{2.20}$$

The three parameters introduced here, α, β and κ 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 $\mathcal{X}_\mathbf{i}$ have been calculated, they are propagated through the non-linear prediction function and the resulting mean and covariance can be calculated.

$$\mathcal{X}_\mathbf{i}^+ = f(\mathcal{X}_\mathbf{i}, u, t) \qquad i = 0, \cdots, 2n \tag{2.21}$$

$$\hat{x} = \sum_{i=0}^{2n} W_i^m \mathcal{X}_\mathbf{i}^+ \tag{2.22}$$

$$P_{xx} = \sum_{i=0}^{2n} W_i^c \left\{\mathcal{X}_\mathbf{i}^+ - \hat{y}\right\} \left\{\mathcal{X}_\mathbf{i}^+ - \hat{y}\right\}^T \tag{2.23}$$

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

$$\mathcal{Y}_\mathbf{i} = h(\mathcal{X}_\mathbf{i}, u, t) \qquad i = 0, \cdots, 2n \tag{2.24}$$

$$\hat{y} = \sum_{i=0}^{2n} W_i^m \mathcal{Y}_\mathbf{i} \tag{2.25}$$

$$P_{yy} = \sum_{i=0}^{2n} W_i^c \left\{\mathcal{Y}_\mathbf{i} - \hat{y}\right\} \left\{\mathcal{Y}_\mathbf{i} - \hat{y}\right\}^T \tag{2.26}$$

$$P_{xy} = \sum_{i=0}^{2n} W_i^c \{\mathcal{X}_\mathrm{i} - \hat{x}\} \{\mathcal{Y}_\mathrm{i} - \hat{y}\}^T \tag{2.27}$$

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[2], 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 probablity 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.

[2]It is not however, as [17] notes, merely a central difference linearization of the functions.

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 interial sensors.

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

$$D_0 = (d, 0, h)^{BF} \tag{2.28}$$
$$D_1 = (0, -d, h)^{BF} \tag{2.29}$$
$$D_2 = (-d, 0, h)^{BF} \tag{2.30}$$
$$D_3 = (0, d, h)^{BF} \tag{2.31}$$

Notation: 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][3] given by

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

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, if 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]:

$$\dot{\xi} = V \tag{2.33a}$$

$$\begin{pmatrix} \dot{q}_1^{wb} \\ \dot{q}_i^{wb} \\ \dot{q}_j^{wb} \\ \dot{q}_k^{wb} \end{pmatrix} = -\frac{1}{2} \begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & -\omega_z & \omega_y \\ \omega_y & \omega_z & 0 & -\omega_x \\ \omega_z & -\omega_y & \omega_x & 0 \end{pmatrix} \begin{pmatrix} q_1^{wb} \\ q_i^{wb} \\ q_j^{wb} \\ q_k^{wb} \end{pmatrix} \tag{2.33b}$$

[3] [22] uses a negated convention of signs compared to what is used here.

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 assumtion, Eulers' extension of Newton's laws of motion for the quadrotor's center of gravity, $\mathcal{G}$, yields

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

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

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 as

$$F_g = mg \cdot e_3^{NED} \tag{2.35}$$

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. 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 as 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 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 (2.36a)-(2.37) [31, 32, 23].

The momentums induced by the propeller rotation and thrust are described in equations (2.36b)-(2.36c). All equations in this section are given in the body-fixed coordinate system.

$$F_{ri} = C_T \rho A_r R^2 \omega_{ri}^2 \begin{pmatrix} -\sin(a_{1_s i}) \\ -\cos(a_{1_s i})\sin(b_{1_s i}) \\ -\cos(a_{1_s i})\cos(b_{1_s i}) \end{pmatrix} \tag{2.36a}$$

$$M_{Qi} = -C_Q \rho A R^3 \omega_{ri} |\omega_{ri}| e_3^{\text{NED}} \tag{2.36b}$$

$$M_{ri} = F_{ri} \times D_{ri} \tag{2.36c}$$

The equations for the flapping angles $(a_{1_s i}, b_{1_s i})$ are derived in [31, 32, 23], but are in (2.37) extended to include the velocity relative to the wind. $V_{ri(n)}$ denotes the n'th element of the vector V_{ri}.

$$V_{\text{rel}} = V - V_{\text{wind}} \tag{2.37a}$$

$$V_{ri} = V_{\text{rel}} + \Omega \times D_{ri} \tag{2.37b}$$

$$\mu_{ri} = \frac{||V_{ri(1,2)}||}{\omega_i R} \tag{2.37c}$$

$$\psi_{ri} = \arctan\left(\frac{V_{ri(2)}}{V_{ri(1)}}\right) \tag{2.37d}$$

$$\begin{pmatrix} a_{1_s} i \\ b_{1_s} i \end{pmatrix} = \begin{pmatrix} \cos(\psi_{ri}) & -\sin(\psi_{ri}) \\ \sin(\psi_{ri}) & \cos(\psi_{ri}) \end{pmatrix} \begin{pmatrix} \frac{1}{1-\frac{\mu_{ri}^2}{2}} \mu_{ri}\left(4\theta_{twist} - 2\lambda_i\right) \\ \frac{1}{1+\frac{\mu_{ri}^2}{2}} \frac{4}{3}\left(\frac{C_T}{\sigma}\frac{2}{3}\frac{\mu_{ri}\gamma}{a} + \mu_{ri}\right) \end{pmatrix} + \begin{pmatrix} \frac{-\frac{16}{\gamma}\left(\frac{\omega_\theta}{\omega_{ri}}\right) + \left(\frac{\omega_\psi}{\omega_{ri}}\right)}{1-\frac{\mu_{ri}^2}{2}} \\ \frac{-\frac{16}{\gamma}\left(\frac{\omega_\psi}{\omega_{ri}}\right) + \left(\frac{\omega_\theta}{\omega_{ri}}\right)}{1+\frac{\mu_{ri}^2}{2}} \end{pmatrix} \tag{2.37e}$$

$$\lambda_i = \mu \alpha_{si} + \frac{v_{1i}}{\omega_i R} \tag{2.37f}$$

$$v_{1i} = \sqrt{-\frac{V_{rel}^2}{2} + \sqrt{\left(\frac{V_{rel}^2}{2}\right)^2 + \left(\frac{mg}{2\rho A_r}\right)^2}} \tag{2.37g}$$

$$C_T = \frac{\sigma a}{4}\left\{\left(\frac{2}{3} + \mu_{ri}^2\right)\theta_0 - \left(\frac{1}{2} + \frac{\mu^2}{2}\right)\theta_{\text{twist}} + \lambda\right\} \tag{2.37h}$$

$$\alpha_{si} = \frac{\pi}{2} - \arccos\left(-\frac{V_{rel} \cdot e_z}{||V_{rel}||}\right) \tag{2.37i}$$

$$C_Q = \sigma a\left[\frac{1}{8a}\left(1 + \mu_{ri}^2\right)\bar{C}_d + \lambda\left(\frac{1}{6}\theta_0 - \frac{1}{8}\theta_{\text{twist}} + \frac{1}{4}\lambda\right)\right] \tag{2.37j}$$

Symbol	Expression	Description	Unit
a	$\frac{\mathrm{d}C_L}{\mathrm{d}\alpha} \approx 2\pi$	Slope of the lift curve.	$\frac{1}{\mathrm{rad}}$
α_{si}	-	Propeller angle of attack.	rad
A_r	-	Rotor disk area.	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 ω_i, as per 2.36a.	1
C_{T0}	-	Linearization point for thrust coefficient.	1
C_Q	*	Torque coefficient. This constant primarily is the scaling factor relating the square of ω_i to the torque from each rotor.	1
γ	$\frac{\rho a c R^4}{I_b}$	γ is the Lock Number [23], described as the ratio between the aerodynamic forces and the inertal forces of the blade.	1
I_b	-	Rotational inertia of the blade	kgm^2
λ_i	*	λ_i denotes the air inflow to the propeller.	1
R	-	Rotor radius.	m
ρ	-	Air density.	$\frac{\mathrm{kg}}{\mathrm{m}^3}$
σ	$\frac{\text{blade area}}{\text{disk area}}$	Disk solidity.	1
θ_0	-	The angle of the propeller at its base, relative to the horizontal disk plane.	rad
θ_{twist}	-	The angle with which the propeller is twisted.	rad
$\omega_\phi, \omega_\theta, \omega_\psi$	-	The rotational, body-fixed, velocity of the quadrotor.	$\frac{\mathrm{rad}}{\mathrm{s}}$
ω_{ri}	-	The rotational velocity of propeller i.	$\frac{\mathrm{rad}}{\mathrm{s}}$
μ_{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

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 trough the measurement update. The wind velocities in the filter is given in the NEDEF reference frame.

The wind drag force is calculated using equation (2.38), whereas the moments are given by equations (2.39). 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.

$$F_{\text{wind}} = F_{wind,body} + \sum_{i=0}^{3} F_{wind,ri} \tag{2.38a}$$

$$F_{\text{wind,body}} = -\frac{1}{2} C_D \rho A V_{rel} ||V_{rel}|| \tag{2.38b}$$

$$F^{BF}_{\text{wind},ri} = -\frac{1}{2} \rho C_{D,r} \sigma A_r (V_{ri} \cdot e^{BF}_{P_{ri}3}) ||V_{ri} \cdot e^{BF}_{P_{ri}3}|| e^{BF}_{P_{ri}3} \tag{2.38c}$$

$$M_{\text{wind}} = M_{\text{wind,body}} + \sum_{i=0}^{3} M_{\text{wind},ri} \tag{2.39a}$$

$$M_{\text{wind,body}} \approx 0 \tag{2.39b}$$

$$M_{\text{wind},ri} = D^{BF}_{ri} \times F^{BF}_{\text{wind},ri} \tag{2.39c}$$

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

$$C_H = \sigma a \left[\frac{1}{4a}\mu \bar{C_d} + \frac{1}{4}\lambda\mu\left(\theta_0 + \frac{\theta_{twist}}{2}\right)\right] \tag{2.40}$$

$$F_{\text{hub},i} = -C_H \rho A R^2 \omega_i^2 \hat{x} \tag{2.41}$$

$$M_{\text{hub},i} = D_i \times F_{hub,i} \tag{2.42}$$

Rolling Moment

$$C_{\text{RM}} = -\sigma a \mu \left[\frac{1}{6}\theta_0 + \frac{1}{8}\theta_{twist} - \frac{1}{8}\lambda_i\right] \tag{2.43}$$

$$M_{\text{RM},i} = C_{\text{RM}} \rho A R^3 \omega_i^2 \tag{2.44}$$

Ground Effect

$$T_{\text{IGE}} = \frac{1}{1 - \frac{R^2}{16z^2}} T \tag{2.45}$$

Gyro Effects and Counter-Torque I_{rotor} is the propeller inertia.

$$\begin{pmatrix} \dot{\omega}_\theta \dot{\omega}_\psi (I_{yy} - I_{zz}) + I_{\text{rotor}} \dot{\omega}_\theta \sum_{i=0}^4 \omega_{ri} \\ \dot{\omega}_\theta \dot{\omega}_\psi (I_{zz} - I_{xx}) + I_{\text{rotor}} \dot{\omega}_\theta \sum_{i=0}^4 \omega_{ri} \\ \dot{\omega}_\theta \dot{\omega}_\phi (I_{xx} - I_{yy}) + I_{\text{rotor}} \sum_{i=0}^4 \dot{\omega}_{ri} \end{pmatrix} \tag{2.46}$$

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 as these are quite common, 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 Accelerometer

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.

$$\hat{y}_{\text{acc}} = a_{\mathcal{G}} + \dot{\omega} \times r_{\text{acc}/\mathcal{G}} + \omega \times \left(\omega \times r_{\text{acc}/\mathcal{G}}\right) + e_{\text{acc}} \tag{2.47}$$

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

Table 2.4. Table of Symbols used in the pressure equation, (2.52)

.

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.

$$y_{\text{gyro}} = \omega + b + e_{\text{gyro}} \tag{2.48}$$

$$\dot{b} = 0 + e_{\text{bias}} \tag{2.49}$$

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 the vehicle.

$$y = R(q)m^e + e_m \tag{2.50}$$

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

$$m^e = \begin{pmatrix} 15.7 & 1.11 & 48.4 \end{pmatrix}^T_{NEDEF} \mu T. \tag{2.51}$$

2.4.4 Pressure Sensor

The air pressure, p, can be related to height using the following equation[40]

$$p = p_0 \left(1 - \frac{L \cdot h}{T_0}\right)^{\frac{g \cdot M}{R \cdot L}} + e_p \tag{2.52}$$

2.4.5 Camera

To estimate the position of the camera using the captured images, the PTAM-library[4] 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^{PTAM},
- and orientation, $q^{PTAM,c}$.

However, since the quite arbtirary[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, $\varnothing_{\text{PTAM}}$,
- 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 2.53, forming the full transformation from the global NEDEF system to the PTAM coordinate frame.

$$x^{\text{PTAM}} = \underbrace{S(s)R(q^{Pw})T(-\varnothing_{\text{PTAM}})}_{\triangleq \mathcal{J}^{Pw}\text{,transformation from camera to PTAM}} x^{\text{NEDEF}} \qquad (2.53)$$

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 2.4.6 respectively, whereas the teleportation problem is discussed in Section 2.4.7.

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

[4]`http://www.robots.ox.ac.uk/~gk/PTAM/`

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;

$$q^{Pw} = q^{PTAM,c} q^{cb} q^{bw}. \tag{2.54}$$

To determine the distance to this plane according to the current estimation, Equation (2.55) is solved for λ in accordance with 2.56.

$$\begin{cases} \varnothing_{\text{PTAM}} & = \xi + R(q^{wb}) r_{\text{camera}/\mathcal{G}} + \lambda R(q^{wP}) \frac{X^{PTAM}}{|X^{PTAM}|} \\ \varnothing_{\text{PTAM}} \cdot \hat{z} & = 0 \end{cases} \tag{2.55}$$

$$\lambda = -\frac{\left(\xi + R(q^{wb}) r_{\text{camera}/\mathcal{G}}\right) \cdot \hat{z}}{\left(R(q^{wP}) \frac{X^{PTAM}}{|X^{PTAM}|}\right) \cdot \hat{z}} \tag{2.56}$$

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

$$s = \frac{|X^{PTAM}|}{|\lambda|}. \tag{2.57}$$

Together, these parameters form an initial estimate of the transformation connecting the PTAM reference frame. This estimate is inserted into the global observer filter, introducing eight new states containing q^{Pw}, s and $\varnothing_{\text{cam}}$.

2.4.6 Continuous Refinement

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. The measurement update is made separate from the IMU update, using the measurement equations in (2.58).

$$\hat{X}^{\text{PTAM}} = \mathcal{J}^{Pw}(\xi + R(q^{wb}) r_{\text{camera}/\mathcal{G}}) + e_{\text{PTAM,X}} \tag{2.58a}$$

$$\hat{q}^{\text{PTAM,c}} = q^{Pw} q^{wb} q^{bc} + e_{\text{PTAM,q}} \tag{2.58b}$$

2.4.7 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 are monitored for sudden changes in position and if a teleportation is detected, a reinitialization is performed as described in Section 2.4.5, with the exception of the map scale, which remains unchanged.

The teleportation behaviour is detected using simple thresholding

$$X_t^{\text{PTAM}} - X_{t-1}^{\text{PTAM}} > \epsilon \tag{2.59}$$

Chapter 3

Controller

To control the quadrotor's movements, a 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 continous 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,

$$u_t = -L\hat{x}_t \tag{3.3}$$

$$L = R^{-1}B^TS, \tag{3.4}$$

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

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

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;

$$u_t = -L\hat{x}_t + L_r r_t \tag{3.6}$$

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

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 survery 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).

$$\dot{x} = f(x,u) \approx f(x_0,u_0) + \underbrace{\left.\frac{\partial f}{\partial x}\right|_{\substack{x = x_0\\ u = u_0}}}_{A} \underbrace{(x - x_0)}_{\Delta x} + \underbrace{\left.\frac{\partial f}{\partial u}\right|_{\substack{x = x_0\\ u = u_0}}}_{B} \underbrace{(u - u_0)}_{\Delta u} \tag{3.8}$$

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];

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

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, Δu, is then added to u_0 to form the controller output, as in Equation (3.10).

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

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).

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

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

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

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).

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

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 to 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 information in the image. The application of SLAM to this approach is refferred to as *Monocular SLAM*, and two approaches in particular is 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 and appended as states to the filter. As frames arrives, the movement of each feature's movement 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 e.g. [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 at the time the keyframe is grabbed is recorded, and the features first detected in a given keyframe are added to the active set of features. As new video frames arrive, features are reprojected and sought for in the new frame, giving an offset from recorded keyframes which by extension gives the updated position of the camera.

In the PTAM library presented in [20], the adjustment of the keyframe positions - corresponding to the feature states' inherent adjustment in each step of the EKF - is separated from the tracking entirely and moved to a parallell, less time-critical, processing thread[1]. As the update need not be performed real-time each video-frame, this parallellization 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[24] to make the solution computationally tractable.

In the tracking procedure, the PTAM library randomly projects features into the captured video frame. 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.

[1]Hence its name; *Parallell Tracking And Mapping*

Chapter 5

Logic

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 keep it at this position.

Insert tikz here

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, while the control reference is adjusted to move the quadrotor slowly sideways for the stereo initialization performed by the PTAM library.

Insert tikz here

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.

Insert tikz here

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 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. [4] performs the landing 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 descention phase, where the quadrotor lowers until landing has been detected, using the camera feedback and other sensors to stabilize the descent.

Landing Detection

Since we recognize that our estimated position - with origin at our starting point - is not nescessarily consistent with the Height Over Ground (HOG) at the landing site, it is nescessary 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 is 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 state - the second of interest - is easily monitored and could e.g. be filtered using the CUSUM algorithm to increase detection confidence, or simply thresholded to detect landing.

Chapter 6

Results

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

The flapping equations pretty much suck.

Chapter 8

Concluding Remarks

8.1 Conclusions

8.2 Further work

Wind model, drag model, wind momentum on body

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. Eigth 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*[1], while implementing these in a much more efficient and uniform manner. By constraining the intermodular messaging to C++[2] 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

[1] `http://ros.org/`

[2] ROS allows messaging between Python and C++ modules