ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Semester Thesis

Estimation of Actuation Configuration for a Multi-Actuated Blimp

Spring Term 2014

Supervised by:
Konstantinos Alexis
Markus Achtelik

Authors:
Matthias Krebs
Simon Laube

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Estimation of Actuation Configuration for a Multi-Actuated Blimp

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Matthias	Krebs
Simon	Laube

Student supervisor(s)

Konstantinos	Alexis
Markus	Achtelik

Supervising lecturer

Roland	Siegwart

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Place and date	Signature
Place and date	Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Contents

Abstract

In this semester thesis, we present a system calibration framework for the actuation configuration of a multi-actuated blimp. To improve the system model used for control, the model parameters estimate of a spherical blimp is optimized using onboard sensors. By that, the actuator allocation used for control can be calculated more precisely than from CAD data or hand measurements without additional devices. The optimization problem is formulated as a least-squares batch optimization problem of the angular acceleration of the blimp. Using data from both simulation and a real blimp, the actuation configuration and its certainty is calculated and compared for different inputs and blimp configurations.

Symbols

Symbols

a	Style for real scalar.
$\mathbf{a}$	Style for real column vectors.
$\mathbf{A}$	Style for real matrices.
$\dot{a}$	Time derivative of a.
$\hat{a}$	Estimate of a.
$\mathbf{p}_a^{c,b}$	Position vector pointing from b to c, expressed in coordinates a.
$\mathbf{C}_{a,b}$	The 3×3 rotation matrix (or direction cosine matrix) transforming from coordinates b to a, i.e. $\mathbf{p}_a = \mathbf{C}_{a,b}\mathbf{p}_b$.
$\mathbf{a}^\times$	Cross product matrix of vector $\mathbf{a}$.
$\mathbf{a}'$	Transposed of $\mathbf{a}$.
$\mathbf{I}$	Identity matrix.

Indices

w	World frame
b	Blimp frame
m	Motor frame
t	Tangential frame
j	Time iterator
k	Actuation iterator

Acronyms and Abbreviations

ASL	Autonomous Systems Lab
CAD	Computer Aided Design
COB	Center of Buoyancy
COG	Center of Gravity
DOF	Degrees of Freedom
ETH	Eidgenössische Technische Hochschule
EKF	Extended Kalman Filter
FMU	Flight Management Unit
GN	Gauss-Newton Method
IMU	Inertial Measurement Unit

LMA	Levenberg-Marquardt Algorithm
NED	North-East-Down
POA	Point of Attack
PX4	Pixhawk IMU Device
QGC	QGroundControl
RMS	Root Mean Square
SD	Steepest-Descent Method
i.e.	id est *(Lat.)*, meaning "that is to say"
e.g.	exempli gratia *(Lat.)*, meaning "for example"
w.r.t.	with respect to

Chapter 1

Introduction

1.1 Motivation

Blimps are ideal vehicles for being used as an aerial advertising medium. Especially eye-catching are specifically shaped blimps, e.g. blimps that have the shape of a mascot, car or any other commercial product (see figure 1.1). The arrangement of the actuators for those different shaped blimps varies a lot. In general, manual adjustment of the control algorithms is needed or the actuators have to be controlled manually. Our idea is to use some adequate sensors to determine the effect of the actuator on the movement of the blimp. When such an algorithm is available, the blimp hull can be manufactured independently on the actuation units which can then be attached to the blimp. The actuators' effect on the system is then automatically detected by the identification algorithm and this knowledge is used to control the system. This is illustrated by figure 1.2.

Figure 1.1: Custom shaped blimps are used for advertising (Blimpworks, 2014).

1.2 Related Work

This thesis is on estimation of the actuation configuration for a multi-actuated blimp. Various work on thruster configuration estimation can be found for underwater robot systems. Doniec et al. (2014) propose black-box model approach. They calculate an inverse model of the thrusters, i.e. a mapping from a desired rotation and position change of the robot to the thruster commands. After about 40 seconds of random walk, the inverse model is computed as the Moore-Penrose pseudoinverse of the collected input/output data. Van de Ven et al. (2005) show an overview of neural network control of underwater vehicles.

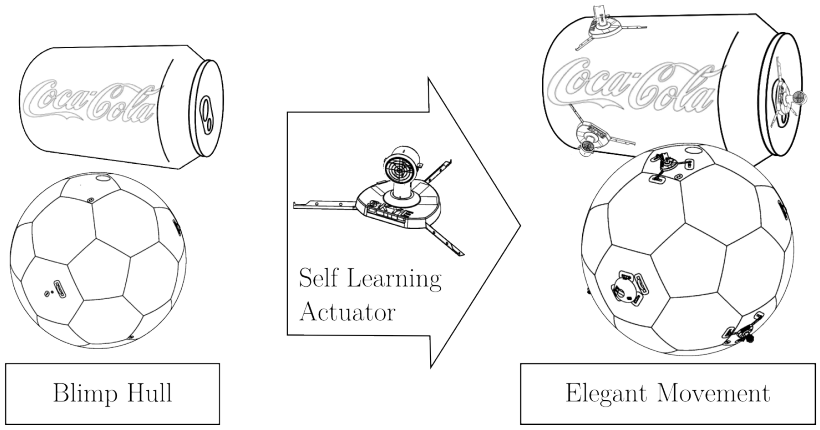

Figure 1.2: The goal is to find an algorithm, that allows to take any blimp hull (**left**) and attach the actuation units (**center**). The actuation configuration is then automatically detected, such that elegant movements can be performed without manual controller adjustments (**right**).

If a first principle-based model of the system is available, gray-box modelling may be applied. Many aerospace and robotics tasks are solved using physical meaningful models. To determine the sensor alignments after launch shock, Shuster et al. (1991) estimate the rotation vector between multiple star tracker devices using batch optimization. Bloesch et al. (2013) formulated a nonlinear least squares problem to estimate kinematic parameters of a legged robot. In depth studies using batch optimization have been made for intrinsic and extrinsic (stereo) camera calibration (Siegwart et al., 2011, chap. 4). In recent time, visual-inertia sensor calibration has found much attraction. Beside others, Hol (2011) showed both batch and recursive estimation (Kalman Filter) for this task.

1.3 Skye System Overview

The experimental part of this thesis is done with the Skye System (Burri et al., 2013). Skye is a spherical blimp (figure 1.3, right) where the center of gravity (COG) coincides with the center of buoyancy (COB). Four identical actuators (figure 1.3, left) are attached to the hull of Skye in a tetrahedral arrangement. Each of those actuator units consists of a static platform and part that can be rotated around the vertical axis. Mounted to the rotating part of the actuators there is an impeller which can generate thrust.

This setup gives Skye the ability to move and rotate in all directions with more or less the same efficiency (see Schaffner and Vuilliomenet, 2012, chap. 3).

1.3.1 Coordinate Systems

Different coordinate systems are used in this thesis to represent the blimp states and the actuation properties (see figure 1.3).

The world frame (w-frame) is a stationary North-East-Down (NED) frame (figure 1.3, bottom).

The blimp frame (b-frame) is a blimp body fixed frame (figure 1.3, right). When the camera is upright and pointing north, it matches a north-east-down (NED)

frame. Its origin coincides with the center of gravity of the blimp.

The motor frames (m^k-frame) (figure 1.3, left) are aligned to each actuator units $k = 1, ..., N$. When the actuator is placed on a horizontal surface, $\mathbf{e}^z_{m^k}$ is defined to point downwards. The coordinate system origin is set to the base of the actuator such that the z-axis coincides with the rotation axe of the actuator. $\mathbf{e}^x_{m^k}$ is defined to be pointing in the direction of the thruster when its orientation angle α^k is zero (see figure 1.4).

The tangential reference frames (t^k-frame) are used to express the estimated actuator configuration. If the true position and orientation of the motors is known (e.g. in simulation), t^k is the coordinate frame for the true motor position and orientation, whereas m^k is the frame for the estimated ones. If the true position is not known (e.g. for real data), any close estimate to the true is taken for t^k and serves as a common frame to compare multiple estimates. The use for this frame will become clear in section 4.1.

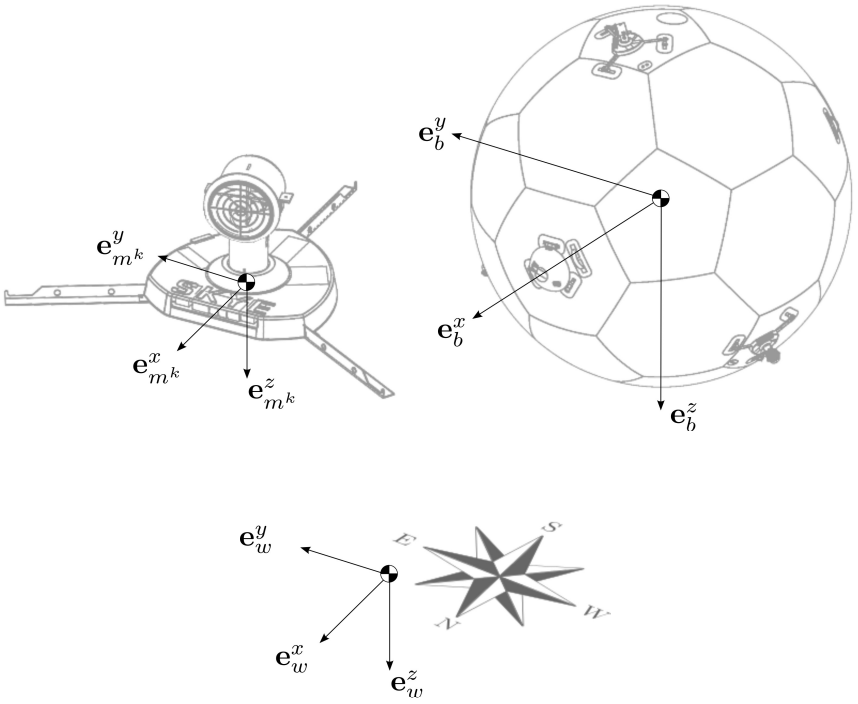

Figure 1.3: **Left**: Motor coordinate frame m^k. When the thruster is not rotated, it directs in $\mathbf{e}^x_{m^k}$ direction. $\mathbf{e}^z_{m^k}$ directs to the bottom. **Right**: Blimp coordinate frame b. $\mathbf{e}^x_b$ directs to the camera which is mounted at the front. $\mathbf{e}^z_b$ directs to the bottom.

Now as we have introduced the coordinate systems, we can formulate the force vector $\mathbf{F^k_{m^k}}$ generated by the actuation unit k expressed in its local motor frame m^k as

$$\mathbf{F^k_{m^k}} = \begin{bmatrix} F^k \cos(\alpha^k) \\ F^k \sin(\alpha^k) \\ 0 \end{bmatrix} \tag{1.1}$$

as it is shown in figure 1.4. α^k is the orientation angle around $\mathbf{e}^z_{m^k}$ and F^k is the thrust force magnitude.

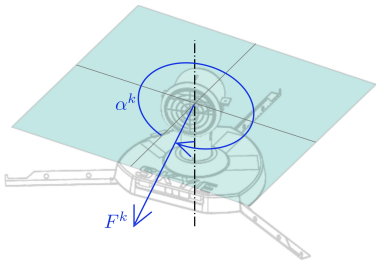

Figure 1.4: Motor orientation angle α^k and thrust force F^k.

1.3.2 Hardware and Software

The control hardware and software of Skye is based upon the framework from the Pixhawk Research Project (Pixhawk, 2014). The firmware of Skye is based on Pixhawk's PX4FMU device which includes an IMU. For operating Skye, a QGroundControl based interface is installed on a laptop which is connected via a XBee link to the PX4 autopilot. Piloting is possible by a 3D mouse from 3DConnexion (see Krebs and Ledergerber, 2012).

1.3.3 Sensors

The sensors used for this thesis are part of the PX4FMU. Although there are some more sensors on the device, the most relevant sensor is the MPU-6000 Six-Axis (Gyro + Accelerometer) MEMS MotionTracking™ Device. This sensor measures angular velocities around three axes as well as translational acceleration in three directions.
To get an understanding of the performance of the gyro of the MPU-6000, we measured the noise to be about $3.2 \cdot 10^{-4} rad/s$ RMS (see Appendix figure B.1).

1.3.4 Allocation

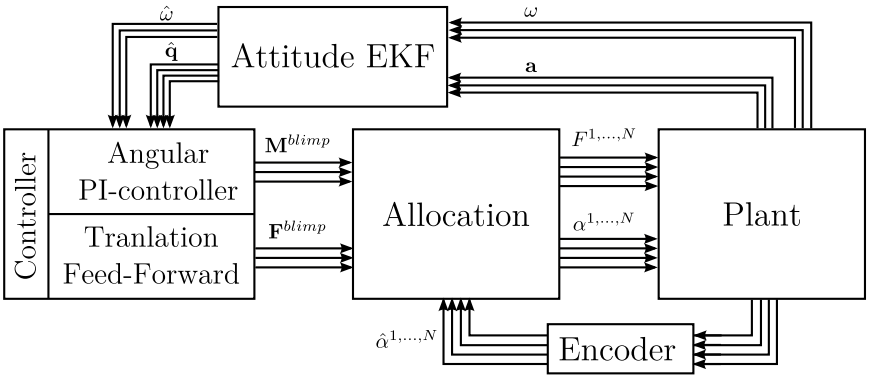

Figure 1.5: Skye control loop. The allocation transforms the blimp control force and moment to the actuator signals. Good knowledge about the actuator configuration is required to achieve precise control.

One characteristic feature of Skye is the way controlled flight is achieved. Figure 1.5 shows the control loop of Skye. Because the four actuators have 8DOF in total and the flying body has only 6DOF that are subject to control, the mapping between the controller output and the actuator inputs is not unique. This mapping is designed to

use minimal power consumption. This is solved as a constraint optimization problem. An analytical solution for the linear mapping has been derived by Schaffner and Vuilliomenet (2012) using Lagrangian multipliers. For the constraints of this optimization problem, it is crucial to know the position and orientation of the motors. Because Skye is over 2.5 m in diameter it is difficult to measure the actuation configuration precisely. The current system uses the actuator position from CAD data of Skye.

1.4 Problem Evaluation

The main goal of this thesis is the *Estimation of Actuation Configuration for a Multi-Actuated Blimp*. The need for a good estimate of the actuation configuration is mainly given by the allocation part of the controller as described before. Additional value can be achieved by estimating further properties, e.g. the offset between COG and COB. If this offset is known, it is possible to calculate the required taring weight apportionment to get any desired COG offset[1].

Optimization Method Selection
As it becomes clear by reading related work in section 1.2, there exist various methods for system calibration. In general, online estimation techniques are preferred for dynamically changing properties (state estimation). Offline estimation is preferred for estimating constant properties (parameter estimation). The actuation configuration is assumed to be constant over a significant period of the flight time. Therefore we will use offline estimation in this thesis.

Offline estimation means that we collect a batch of data and calculate the estimate of the actuation configuration out of this data. If the parameters change slowly over time, it is also possible to repeat the batch optimization after some time to consider these changes.

For the estimation of the actuation configurations, we assume the system states to be known. We will use the available information about the states from the online state estimator which is already implemented on the system.

Restriction to Spherical Blimps
For the experimental validation in this thesis, we will use the Skye System. Since Skye is a spherical blimp we can only verify the algorithm for spherical systems. Because of this, we decided to focus the limited scope of a semester thesis on just spherical blimps.

However, we designed the framework such that it can be expanded for the estimation of arbitrary actuator positions. Some aspects as observability need to be considered when expanding the optimization task. The thoughts outlined in section 2.2.4 will serve as a useful base.

System Model Selection
For bulky, but rotation symmetric systems as blimps, aerodynamic effects have much more influence on the translational movements than on the rotational movements. Effects like flow separation, drag crisis or the inertia of the surrounding fluid

[1] In the simulation (see section 3.3), such an algorithm is already implemented to place the taring weights.

mainly influence the translational movements. On rotational movements, aerodynamic effects have a much smaller impact. Since it is a difficult task to model aerodynamic effects and this would lead to many more unknown parameters, we decided to restrict ourselves on the investigation of the rotational movement.

Chapter 2

Optimization: Problem Formulation and Solution

Figure 2.1 gives an overview on the batch optimization problem formulation. On one hand, the real system is fed with some specific inputs and the system behaviour is measured by the available sensors. On the other hand, virtual sensor measurements are calculated for the same inputs using a parameterized system model. The optimization algorithm has to be designed to find the parameters that minimize the difference between the real and virtual system outputs.

In the following section, the system model and its parameters are described. Further, the problem is transformed into nonlinear least squares form which can be solved by the Levenberg-Marquardt algorithm. Finally, an analytical estimate for the parameter certainty is derived.

2.1 System Model

A system model for Skye has been stated in (Weichart, 2012) and is summarized in table 2.1.
It consists on the differential equations for the angular velocity $\boldsymbol{\omega}$, orientation quaternion $\mathbf{q}$, velocity $\mathbf{v}$, and position $\mathbf{p}$. Skye is assumed as a rigid body. Active forces and moments with respect to its center of gravity are concentrated as

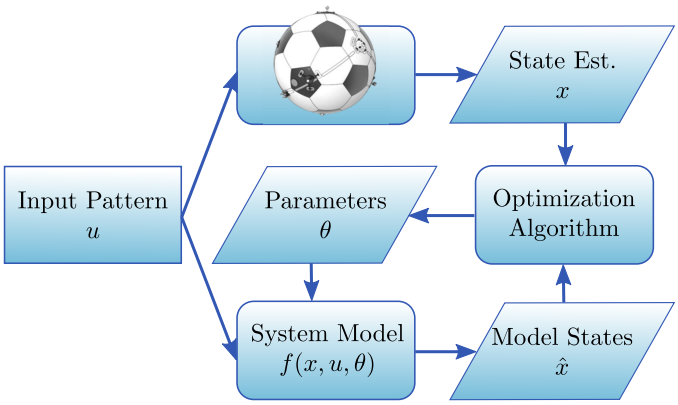

Figure 2.1: Scheme for optimization of parameterized system model.

Table 2.1: Equations of motion for Skye

$\dot{\boldsymbol{\omega}}_b^b$	$= \mathbf{J}_b^{-1}\left(\mathbf{M}_b - \boldsymbol{\omega}_b^b \times \mathbf{J}_b\boldsymbol{\omega}_b^b\right)$
$\dot{\mathbf{q}}_w^{b,w}$	$= \frac{1}{2}\mathbf{q}_w^{b,w} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}_b^b \end{bmatrix}$
$\dot{\mathbf{v}}_b^b$	$= \mathcal{M}_b^{-1}\mathbf{F}_b$
$\dot{\mathbf{p}}_w^{b,w}$	$= \mathbf{v}_w^b$

F and **M** respectively. The total mass $\mathcal{M}$ includes the rigid mass, helium mass as well as the virtual mass compensating for the inertia of the surrounding fluid. The moment of inertia (or inertia tensor) **J** does not contain a virtual part as long as sphere like blimps are considered.
For the optimization, only the angular acceleration $\boldsymbol{\alpha}$ is considered as outlined in section 1.4. Subsequently, a closer look at the corresponding equation is given.

$$\boldsymbol{\alpha}_b^b = \mathbf{J}_b^{-1}\left(\mathbf{M}_b - \boldsymbol{\omega}_b^b \times \mathbf{J}_b\boldsymbol{\omega}_b^b\right) \tag{2.1}$$

The change of angular velocity is given by the active moment **M** as well as the nutation term $\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}$.
The latter term influences the rotation axis such that the free body will asymptotically end in a rotation around its smallest or largest principle inertia axis. For a homogeneous sphere, the inertia tensor is diagonal with all its diagonal elements equal and the cross product in eq. (2.1) is therefore zero (because $\boldsymbol{\omega}$ and $\mathbf{J}\boldsymbol{\omega}$ are parallel). As Skye itself is not as perfect symmetric as a sphere (compare section 2.2.3) the nutation term will be considered although it is much smaller than the active moment.
The active moment **M** consists of an actuation term $\mathbf{M}^{actuation}$, a gravitation term $\mathbf{M}^{gravity}$, and an aerodynamic term $\mathbf{M}^{aero}$.

$$\mathbf{M}_b = \underbrace{\sum_{k=1}^{N}\left[\mathbf{C}_{b,m^k}\left(\mathbf{p}_{m^k}^{m^k,cog} \times \mathbf{F}_{m^k}^k\right)\right]}_{\mathbf{M}^{actuation}} - \underbrace{\left(\mathbf{p}_b^{cob,cog} \times (\mathbf{C}_{b,w} m\mathbf{g}_w)\right)}_{\mathbf{M}^{gravity}} + \mathbf{M}^{aero} \tag{2.2}$$

For the actuation term, the moment (cross product between position vector $\mathbf{p}^{cog,m^k}$ from COG to thruster's point of action and motor force $\mathbf{F}^k$) is first calculated in the local motor coordinate frames m^k and then transformed by the rotation matrix $\mathbf{C}_{b,m^k}$ to blimp coordinates b for each motor $k = 1, ..., N$.
The gravity term includes the offset $\mathbf{p}^{cob,cog}$ from COG to COB and the gravitation force $m\mathbf{g}$. The gravitation force is only known in world coordinates w and hence is transformed by the rotation matrix $\mathbf{C}_{b,w}$ from world coordinates w to blimp coordinates b.
The aerodynamic term can be modelled as aerodynamic friction. According to Kundu et al. (2012), the wall shear stress is usually expressed in terms of the dimensionless skin friction coefficient c_f as

$$\tau_{wall} = \frac{1}{2} c_f \rho_{air} v^2. \tag{2.3}$$

The fluid velocity v at the wall is composed by the angular and translational velocity of the blimp as well as the movement of the fluid itself. For simplification, we assume the the fluid to be at rest. The effect of the aerodynamic moment is dominated by the rotational movement of the blimp. Because the blimp is symmetric, a pure translation would not lead to a moment. If we assume the translational velocity

much smaller than the angular velocity, it can be neglected and the wall shear stress can be integrated over the spherical hull.

$$\begin{aligned}\|\mathbf{M}^{aero}\| &= \int_A \xi \tau_{wall} dA \\ &= \frac{1}{2} c_f \rho_{air} \int_A \xi (\xi\omega)^2 dA \\ &= \frac{1}{2} c_f \rho_{air} \omega^2 \int_{-r}^{r} (r^2 - z^2)^{\frac{3}{2}} \sqrt{r^2 - z^2} 2\pi dz \\ &= \frac{1}{2} c_f \rho_{air} \omega^2 \frac{32}{15} \pi r^5 = \frac{16}{15} c_f \rho_{air} \pi r^5 \omega^2\end{aligned} \tag{2.4}$$

Nevertheless, this does not consider the drag of all components which are attached on the hull (e.g. actuation units or handles) which will have a significant effect on the aerodynamics.

Measurements showed that the total aerodynamic drag is more than a magnitude smaller than the usual actuation moments ($\|\mathbf{M}^{aero}(\omega = 0.8\,\text{rad/s})\| \approx 0.2\,\text{Nm} \Leftrightarrow \|\mathbf{M}^{actuation}_{max}\| \approx 15\,\text{Nm}$). Therefore we neglect this term and do not further bother about its direction. In section 4.3.3 we will show by simulation results that it is indeed valid to neglect this term.

2.2 Parameterization

Inserting eq. (2.2) into eq. (2.1), we get the model for the angular acceleration as

$$\mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) = \tag{2.5}$$
$$\mathbf{J}_b^{-1} \left(\sum_{k=1}^{N} \left[\mathbf{C}_{b,m^k} \left(\mathbf{p}_{m^k}^{m^k,cog} \times \mathbf{F}_{m^k}^{k} \right) \right] - \left(\mathbf{p}_b^{cob,cog} \times (\mathbf{C}_{b,w} m \mathbf{g}_w) \right) - \boldsymbol{\omega}_b^b \times \mathbf{J}_b \boldsymbol{\omega}_b^b \right)$$

Every variable in eq. (2.5) must be classified as being part of the system state $\mathbf{x}$, the input $\mathbf{u}$, the parameter $\boldsymbol{\theta}$ or as a (constant) known variable[1]. The variables of the system model can be classified as shown in table 2.2.

Table 2.2: Classification of all variables of the system model equation. All variables which are subject to estimation are introduced as parameter.

Motor position	$\mathbf{p}_{m^k}^{m^k,cog}$	param
Motor orientation	$\mathbf{C}_{b,m^k}$	param
Blimp COG offset	$\mathbf{p}_b^{cob,cog}$	param
Blimp inertia tensor	$\mathbf{J}_b$	param
Blimp mass	m	param
Actuator force	$\mathbf{F}_{m^k}^{k}$	input
Blimp orientation	$\mathbf{C}_{b,w}$	state
Blimp angular velocity	$\boldsymbol{\omega}$	state
Gravitation	$\mathbf{g}_w$	known

In order to state a feasible problem, it must be possible to detect all parameters of the system model. As the observability analysis in section 2.2.4 shows, the parameters in table 2.2 are not independently observable. Therefore it is necessary

[1] In this thesis we only use time invariant variables. Continuous time variables can be added to batch optimization e.g. by using spline functions as shown in (Furgale et al., 2012).

to reduce the number of parameters and replace those parameters that can easily be measured as known. An updated classification of the variables will be shown in table 2.3.

There are several ways of how to represent the parameters. We will show our choice in the sections below.

2.2.1 Motor Position

In general, the motor position is a vector in the three dimensional space. It points from the blimp's COG to the thruster's point of attack.

$$\mathbf{p}^{m^k,cog} = \begin{bmatrix} p_1^{m^k,cog} \\ p_2^{m^k,cog} \\ p_3^{m^k,cog} \end{bmatrix} \in \mathbb{R}^3 \tag{2.6}$$

If we restrict the blimp hull to a perfect sphere, the ideal motor position vector space reduce to those vectors with length r. The radius r is the distance from the blimp's COG to the thruster's point of attack. It is equal for all motors and the motor position expressed in the local motor frame m^k (see figure 1.3) reduces to

$$\mathbf{p}_{m^k}^{m^k,cog} = \begin{bmatrix} 0 \\ 0 \\ -r \end{bmatrix} \in \mathbb{R}^3 \tag{2.7}$$

The actual position on the hull is then simply given by the motor orientation, which is explained next. The radius r be measured by an appropriate measuring device. As we will show in section 2.2.4, a scale offset of the now 'known' motor position leads to a scale offset of the estimates about the inertia tensor $\mathbf{J}$ and COG offset $\mathbf{p}^{cob,cog}$.

2.2.2 Motor Orientation

Since the motor thrust is known in motor coordinates m^k, a transformation into blimp coordinates is necessary to calculate the dynamics of the blimp. The coordinate transformation is described by a *special orthogonal group in the three dimensional space* SO(3). The SO(3) can either be minimal represented by 3 parameters (e.g. Euler angles, rotation vector, Gibbs-Rodriquez parameters) or nonminimal represented by $p > 3$ parameters and $p - 3$ constraints (e.g. quaternions, rotation matrix). While minimal representations always suffer from a singularity (representation is not *bijective*), nonminimal representations generally do not contain a singularity.

Here, a trade-off as to be made. Either, one specific motor orientation which cannot be observed is introduced or a constraint optimization problem must be solved. Unconstraint optimization problems are much less demanding to solve. Therefore it is more convenient to choose a parameterization such that the motor orientation is not on the singularity.
If a rough estimate of the motor orientation can be made, this can be used to introduce an intermediate coordinate frame, such that the true motor orientation will not be on the singularity with very heigh probability. Therefore, we go with minimal representation for this problem[2].

[2] An alternative implementation with nonminmal representation (quaternions) using either hard or soft constraints showed worse performance than using minimal representation.

Because of its computationally cheap and for optimization sufficiently smooth formula to get the rotation matrix (see eq. (2.9)), we use Gibbs-Rodriquez parameters $\boldsymbol{\lambda}$. Starting from rotation angle φ and rotation axis $\mathbf{n}$, these parameters are defined as:

$$\boldsymbol{\lambda} = \mathbf{n} \tan(\varphi/2) \tag{2.8}$$

The singularity of the Gibbs-Rodriquez parameters lies at $\varphi = \pi$. The direction cosine transform is calculated as

$$\mathbf{C} = \mathbf{I} + \frac{2}{1 + \boldsymbol{\lambda}^\top \boldsymbol{\lambda}} \left(\boldsymbol{\lambda}^\times + \left(\boldsymbol{\lambda}^\times \right)^2 \right) \tag{2.9}$$

2.2.3 Inertia Tensor

The inertia tensor of a rigid body is calculated as

$$\mathbf{J} = \int_V \rho(x, y, z) \begin{bmatrix} y^2 + z^2 & -xy & -xz \\ -xy & z^2 + x^2 & -yz \\ -xz & -yz & x^2 + y^2 \end{bmatrix} dV \tag{2.10}$$

and is always symmetric. In its principle frame, the off-diagonal components are zero. We do not restrict the inertia tensor's principle frame to be aligned with the blimp frame b. Therefore $\mathbf{J}_b$ is represented by six parameters[3] $\boldsymbol{\vartheta} = (\vartheta_1, \ldots, \vartheta_6)'$.

$$\mathbf{J}_b = \begin{bmatrix} \vartheta_1 & \vartheta_4 & \vartheta_5 \\ \vartheta_4 & \vartheta_2 & \vartheta_6 \\ \vartheta_5 & \vartheta_6 & \vartheta_3 \end{bmatrix} \tag{2.11}$$

2.2.4 Observability

One way to analytically show observability of a system is shown by Hermann and Krener (1977) using Lie algebra. Here we do not show a proof of observability but argue which parameters are not observable and must therefore be excluded from the optimization problem.
For convenience, we state the parameterized system model eq. (2.5) again.

$$\mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) = \\ \mathbf{J}_b^{-1} \left(\sum_{k=1}^{N} \left[\mathbf{C}_{b,m^k} \left(\mathbf{p}_{m^k}^{m^k,cog} \times \mathbf{F}_{m^k}^{k} \right) \right] - \left(\mathbf{p}_b^{cob,cog} \times (\mathbf{C}_{b,w} m \mathbf{g}_w) \right) - \boldsymbol{\omega}_b^b \times \mathbf{J}_b \boldsymbol{\omega}_b^b \right)$$

The first important observation is, that the scale of the inertia tensor $\mathbf{J}$ is only jointly observable with the scale of the motor position vectors $\mathbf{p}^{m^k,cog}$ and the COG offset $\mathbf{p}^{cob,cog}$. If all those parameters have to be estimated, there exist infinite many solutions where the estimates are

$$\begin{aligned} \hat{\mathbf{p}}^{m^k,cog} &= \gamma \mathbf{p}^{m^k,cog} \\ \hat{\mathbf{p}}^{cob,cog} &= \gamma \mathbf{p}^{cob,cog} \\ \hat{\mathbf{J}} &= \frac{1}{\gamma} \mathbf{J} \end{aligned} \tag{2.12}$$

for any real number γ. The problem becomes observable, when the scale of one of those parameters is fixed, i.e. either the norm of one vector or the determinant of

[3] There exist multiple variants to represent the inertia tensor. One alternative would be to decompose it into the diagonal principle form and the corresponding coordinate transformation.

the tensor is assumed to be known. Note that this scale can be chosen arbitrary with no impact on the motor orientation $\mathbf{C}_{b,m^k}$.
Because we simplified the motor position in eq. (2.7) with the scalar radius r, it is sufficient to assume r as known to get a unique solution for all remaining parameters. Without this simplification, a constraint optimization problem has to be solved.

If one has a look at the COG offset term in eq. (2.5), it can be seen that the same scaling problem exists between $\mathbf{p}^{cob,cog}$ and m. Here again, a wrong estimate of the fixed value m would only result in a scale error of the jointly observable parameter $\mathbf{p}^{cob,cog}$ without impact on the remaining parameters[4].
The updated classification of the variables from table 2.2 is shown in table 2.3.

Table 2.3: Updated variable classification for the system model. Like this, all parameters are observable.

Motor orientation	$\mathbf{C}_{b,m^k}$	param
Blimp COG offset	$\mathbf{p}_b^{cob,cog}$	param
Blimp inertia tensor	$\mathbf{J}_b$	param
Actuator force	$\mathbf{F}_{m^k}^k$	input
Blimp orientation	$\mathbf{C}_{b,w}$	state
Blimp angular velocity	$\boldsymbol{\omega}$	state
Gravitation	$\mathbf{g}_w$	known
Blimp mass	m	known
Motor POA radius	r	known

2.3 Nonlinear Least Squares

As outlined above, our goal is to minimize the error between the measured and modelled angular acceleration. For normally distributed system model errors, the maximum-likelihood estimate of the parameters is equivalent to the solution of the least squares problem (Seber and Wild, 2003). In general, the cost function $S(\boldsymbol{\theta})$ for the ordinary nonlinear least squares problem is

$$S(\boldsymbol{\theta}) = \sum_{i=1}^{n} (y_i - f_i(\mathbf{x}_i, \boldsymbol{\theta}))^2 \tag{2.13}$$

which is the squared sum of the residual between observation y_i and function value $f_i(\mathbf{x}_i, \boldsymbol{\theta})$ for $i = 1, ..., n$ distinct observations. For our problem, we group the observations to the three dimensional angular acceleration measurement vector $\mathbf{y}_j$ and the corresponding system model function[5] $\mathbf{f}(\mathbf{x}_j, \boldsymbol{\theta})$ from eq. (2.5)

$$S(\boldsymbol{\theta}) = \sum_{j=1}^{n/3} \|\mathbf{y}_j - \mathbf{f}(\mathbf{x}_j, \boldsymbol{\theta})\|^2 \tag{2.14}$$

and we have $n/3$ distinct angular acceleration measurements $\mathbf{y}_j$ for time steps $j = 1, ..., n/3$. For being consistent with the literature, we substitute $\mathbf{x}_j$ into the function

[4] In contrast to the COG offset $\mathbf{p}_b^{cob,cog}$, the mass m can easily be measured on scales. Therefore we assume the scalar m as known.

[5] To solve the optimization problem, we do not distinguish between inputs $\mathbf{u}$ and states $\mathbf{x}$. In the remaining of this chapter, we will use $\mathbf{x}$ for states *and* inputs for better readability and consistency with the literature.

$\mathbf{f}_j(\boldsymbol{\theta})$, and use from now on

$$\mathbf{f}(\boldsymbol{\theta}) = \begin{bmatrix} \mathbf{f}(\mathbf{x}_1, \boldsymbol{\theta}) \\ \vdots \\ \mathbf{f}(\mathbf{x}_{n/3}, \boldsymbol{\theta}) \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{n/3} \end{bmatrix} \tag{2.15}$$

and finally get the expression for the ordinary least squares problem with the residual $\mathbf{r}$ as

$$S(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{f}(\boldsymbol{\theta})\|^2 = \|\mathbf{r}\|^2 \tag{2.16}$$

Linear Regression

Before we deal with the solution to the nonlinear least squares problem, we briefly review the linear case. Consider Draper and Smith (1998, chap. 5) for a profound treatment of linear regression. In linear regression, the parameter vector is usually denoted by $\boldsymbol{\beta}$ and the least squares problem for a linear function $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ with normally distributed error $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I}\sigma^2)$ is

$$S(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 = \|\mathbf{r}\|^2 \tag{2.17}$$

The least squares solution for $\boldsymbol{\beta}$ can directly be calculated as

$$\boldsymbol{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \tag{2.18}$$

where

$$\mathbf{V}(\beta) = (\mathbf{X}'\mathbf{X})^{-1}\sigma^2 \tag{2.19}$$

is the covariance matrix of the estimates. An estimate for σ is given by

$$\hat{\sigma}^2 = \frac{1}{n-p}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = \frac{1}{n-p}\|\mathbf{r}\|^2 \tag{2.20}$$

Numerical Methods for Nonlinear Regression

To find a solution for nonlinear regression, iterative schemes have to be applied (Seber and Wild, 2003). Linear Taylor expansion of the nonlinear function $\mathbf{f}(\boldsymbol{\theta})$ for $\boldsymbol{\theta}$ close to $\boldsymbol{\theta}_0$ gives

$$\mathbf{f}(\boldsymbol{\theta}) \approx \mathbf{f}(\boldsymbol{\theta}_0) + \mathbf{F}_0(\boldsymbol{\theta} - \boldsymbol{\theta}_0) \tag{2.21}$$

where $\mathbf{F}_0$ is the Jacobian of $\mathbf{f}(\boldsymbol{\theta})$ evaluated at $\boldsymbol{\theta}_0$

$$\mathbf{F}_0 = \left.\frac{\partial \mathbf{f}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}'}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0} \tag{2.22}$$

For the residual vector $\mathbf{r}(\boldsymbol{\theta})$, we get

$$\begin{aligned} \mathbf{r}(\boldsymbol{\theta}) &= \mathbf{y} - \mathbf{f}(\boldsymbol{\theta}) \\ &\approx \mathbf{r}(\boldsymbol{\theta}) - \mathbf{F}_0(\boldsymbol{\theta} - \boldsymbol{\theta}_0) \end{aligned} \tag{2.23}$$

and the cost function $S(\boldsymbol{\theta}) = \mathbf{r}(\boldsymbol{\theta})'\mathbf{r}(\boldsymbol{\theta})$ becomes

$$S(\boldsymbol{\theta}) \approx \mathbf{r}(\boldsymbol{\theta}_0)'\mathbf{r}(\boldsymbol{\theta}_0) - 2\mathbf{r}(\boldsymbol{\theta})'\mathbf{F}_0(\boldsymbol{\theta} - \boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)'\mathbf{F}_0'\mathbf{F}_0(\boldsymbol{\theta} - \boldsymbol{\theta}_0) \tag{2.24}$$

Following Seber and Wild (2003, chap. 2), the cost is minimized with respect to $\boldsymbol{\theta}$ when

$$\boldsymbol{\theta} - \boldsymbol{\theta}_0 = (\mathbf{F}_0'\mathbf{F}_0)^{-1}\mathbf{F}_0'\mathbf{r}(\boldsymbol{\theta}_0) \tag{2.25}$$

This leads to the Gauss-Newton algorithm, that is

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^i + \boldsymbol{\delta}_i \tag{2.26}$$

where

$$\boldsymbol{\delta}_i = (\mathbf{F}_i'\mathbf{F}_i)^{-1}\mathbf{F}_i'\mathbf{r}(\boldsymbol{\theta}^i) \tag{2.27}$$

Another iterative algorithm to find an optimal solution is the steepest-descent method. The gradient of the cost function $S(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{f}(\boldsymbol{\theta}))'(\mathbf{y} - \mathbf{f}(\boldsymbol{\theta}))$ with respect to $\boldsymbol{\theta}$ is

$$\frac{\partial S(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\mathbf{F}(\mathbf{y} - \mathbf{f}(\boldsymbol{\theta})) = -\mathbf{F}\mathbf{r}(\boldsymbol{\theta}) \tag{2.28}$$

From this follows the steepest-descent algorithm, that is

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^i + \mathbf{F}_i\mathbf{r}(\boldsymbol{\theta}^i) \tag{2.29}$$

Figure 2.2 illustrates the working principle of the steepest-descent method for two parameters.

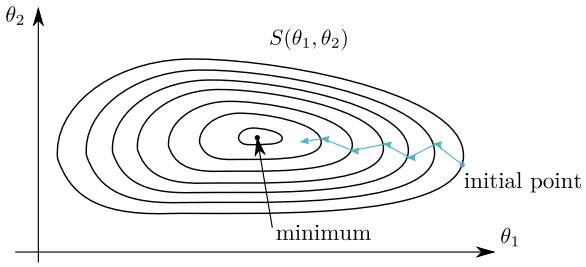

Figure 2.2: Starting at an initial point, the steepest-descent method iteratively proceeds into the negative direction of the gradient (down-hill) and finally converges at a (local) minimum.

2.4 Levenberg-Marquardt Algorithm

Based on ideas by Levenberg, Marquardt (1963) presented the Levenberg-Marquardt Algorithm (LMA). As stated by Marquardt, steepest-descent methods (SD) have *slow convergence after the first few iterations*, and Taylor series based methods, i.e. Gauss-Newton method (GN) suffer by *divergence of successive iterates*. The LMA performs an optimum interpolation between GN and SD and is more robust than GN and has faster convergence than SD. Using the notation introduced in the section above, LMA modifies the Gauss-Newton step eq. (2.27) to

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^i + (\mathbf{F}_i'\mathbf{F}_i + \lambda_i\mathbf{I})^{-1}\mathbf{F}_i'\mathbf{r}(\boldsymbol{\theta}^i) \tag{2.30}$$

where λ_i is a weighting factor between the GN and SD. There exist several possibilities to choose λ_i. Some implementations also use in eq. (2.30) the diagonal values of $\mathbf{F}_i'\mathbf{F}_i$ instead of $\mathbf{I}$, such that the method is not influenced by rescaling of $\boldsymbol{\theta}$.

Because of its convergence rate and robustness, LMA is an appropriate algorithm for many nonlinear least square problems. As described in chapter 3, we will use the MATLAB implementation of the LMA to solve our problem.

2.5 System Model Jacobian

In this section, we will derive the Jacobian for our system model eq. (2.5). In general, the Jacobian $\mathbf{F}(\boldsymbol{\theta})$ has dimension $n \times p$, where n is the number of observations and

p is the number of parameters. Analogue to our notation in eq. (2.14), we first can formulate the Jacobian $\mathbf{F}(\mathbf{x}_j, \boldsymbol{\theta}) \in \mathbb{R}^{3\times p}$ directly for the system model eq. (2.5) and then form the Jacobian for the batch optimization as

$$\mathbf{F}(\boldsymbol{\theta}) = \begin{bmatrix} \mathbf{F}(\mathbf{x}_1, \boldsymbol{\theta}) \\ \vdots \\ \mathbf{F}(\mathbf{x}_{n/3}, \boldsymbol{\theta}) \end{bmatrix} \tag{2.31}$$

The p columns of the Jacobian

$$\mathbf{F}(\mathbf{x}_j, \boldsymbol{\theta}) = \frac{\partial \mathbf{f}(\mathbf{x}_j, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}'}, \qquad j = 1, ..., \frac{n}{3} \tag{2.32}$$

consist of the derivatives of $\mathbf{f}(\mathbf{x}_j, \boldsymbol{\theta})$ with respect to the appropriate parameter. In the following, we use the shorthand notation $\mathbf{f}$ instead of $\mathbf{f}(\mathbf{x}_j, \boldsymbol{\theta})$.
We state again the system model from eq. (2.5) and calculate the derivative w.r.t. the parameters as following.

$$\mathbf{f} = \mathbf{J}_b^{-1} \left(\sum_{k=1}^{N} \left[\mathbf{C}_{b,m^k} \left(\mathbf{p}_{m^k}^{m^k,cog} \times \mathbf{F}_{m^k}^{k} \right) \right] - \left(\mathbf{p}_b^{cob,cog} \times (\mathbf{C}_{b,w} m \mathbf{g}_w) \right) - \boldsymbol{\omega}_b^b \times \mathbf{J}_b \boldsymbol{\omega}_b^b \right) \tag{2.33}$$

Derivative w.r.t Motor Orientation

As stated in section 2.2.2, the motor orientation is parameterized by Gibbs-Rodriquez parameters. The derivative of the direction cosine transform eq. (2.9) has been calculated with MUPAD and is stated in appendix A.2.
The derivative of the system model w.r.t. the Gibbs-Rodriquez parameters $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)'$ is

$$\frac{\partial \mathbf{f}}{\partial \lambda_i} = \mathbf{J}_b^{-1} \left(\frac{\partial \mathbf{C}_{b,m^k}}{\partial \lambda_i} \right) \left(\mathbf{p}_{m^k}^{m^k,cog} \times \mathbf{F}_{m^k}^{k} \right), \qquad i = 1, 2, 3 \tag{2.34}$$

Derivative w.r.t. Motor Position

Using the rules for derivative of the cross product in appendix A.1, the derivative of the system model w.r.t. the motor position $\mathbf{p}_{m^k}^{m^k,cog}$ is

$$\frac{\partial \mathbf{f}}{\partial (\mathbf{p}_{m^k}^{m^k,cog})'} = -\mathbf{J}_b^{-1} \mathbf{C}_{b,m^k} \left(\mathbf{F}_{m^k}^{k} \right)^{\times}, \qquad k = 1, ..., N \tag{2.35}$$

If the motor position is substituted by the radius r (see eq. (2.7)), the derivative of the system model w.r.t the POA radius r is simply

$$\frac{\partial \mathbf{f}}{\partial r} = \mathbf{J}_b^{-1} \mathbf{C}_{b,m^k} \left(\begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \times \mathbf{F}_{m^k}^{k} \right), \qquad k = 1, ..., N \tag{2.36}$$

Derivative w.r.t. Center of Gravity

The derivative of the system model w.r.t. the COG offset $\mathbf{p}_b^{cob,cog}$ is

$$\frac{\partial \mathbf{f}}{\partial (\mathbf{p}_b^{cob,cog})'} = -\mathbf{J}_b^{-1} \left(\mathbf{C}_{b,w} m \mathbf{g}_w \right)^{\times} \tag{2.37}$$

Derivative w.r.t. Inertia Tensor

The derivative of the system model w.r.t. the inertia tensor parameters $\boldsymbol{\vartheta}$ is according to the chain rule

$$\frac{\partial \mathbf{f}}{\partial \boldsymbol{\vartheta}'} = -\left(\frac{\partial \mathbf{J}_b^{-1}}{\partial \boldsymbol{\vartheta}'}\boldsymbol{\omega}^{\times}\mathbf{J}_b\boldsymbol{\omega} + \mathbf{J}_b^{-1}\boldsymbol{\omega}^{\times}\frac{\partial \mathbf{J}_b}{\partial \boldsymbol{\vartheta}'}\boldsymbol{\omega}\right) \tag{2.38}$$

The derivative of the inverse tensor has been calculated with MUPAD.

2.6 Confidence Region

For having a rough idea about the certainty of the parameters found by batch optimization, the covariance matrix of the estimates for nonlinear least squares is calculated. According to Seber and Wild (2003, chap. 2) the covariance can be calculated using first order Taylor series equivalent to the linear case eq. (2.19) as

$$\mathbf{V}(\boldsymbol{\theta}) = (\mathbf{F}(\boldsymbol{\theta})'\mathbf{F}(\boldsymbol{\theta}))^{-1}\sigma^2 \tag{2.39}$$

where an estimate for σ is given by

$$\hat{\sigma}^2 = \frac{1}{n-p}\mathbf{r}(\boldsymbol{\theta})'\mathbf{r}(\boldsymbol{\theta}) \tag{2.40}$$

To calculate the covariance of the estimated variables, we use the error propagation law as given by Siegwart et al. (2011, eq. 4.15)

$$\mathbf{V}_Y = \mathbf{G}_X\mathbf{V}_X\mathbf{G}_X' \tag{2.41}$$

where

$\mathbf{V}_X$ = covariance matrix of the input uncertainties;

$\mathbf{V}_Y$ = covariance matrix of the propagated uncertainties for the outputs;

$\mathbf{G}_X$ is the Jacobian matrix of any function $\mathbf{g}$ w.r.t. X;

Note that eq. (2.19) is basically the inverse formulation of eq. (2.41) where $\mathbf{V}_Y = \mathbf{I}\sigma^2$.

Using the estimate of the parameter covariance eq. (2.39) and the error propagation law eq. (2.41), we can calculate the covariance (and hence further statistics as 95% confidence region) of our estimates.

As an example, we calculate the variance of the motor position given the variance of the Gibbs-Rodriquez parameters $\boldsymbol{\lambda}$. The motor position, expressed in blimp coordinates b, is

$$\mathbf{p}_b^{m^k} = \mathbf{C}_{b,m^k}(\boldsymbol{\lambda})\mathbf{p}_{m^k}^{m^k} \tag{2.42}$$

The covariance matrix $\mathbf{V}(\boldsymbol{\lambda})$ is calculated by eq. (2.39) and hence we get the error propagation by eq. (2.41) as

$$\mathbf{V}(\mathbf{p}_b^{m^k}) = \left(\frac{\partial \mathbf{p}_b^{m^k}}{\partial \boldsymbol{\lambda}'}\right)\mathbf{V}(\boldsymbol{\lambda})\left(\frac{\partial \mathbf{p}_b^{m^k}}{\partial \boldsymbol{\lambda}'}\right)' \tag{2.43}$$

To express the variance in the tangential reference frame t^k (we will use that in chapter 4), we can apply coordinate transformation for the covariance matrix and yield

$$\mathbf{V}(\mathbf{p}_{t^k}^{m^k}) = \mathbf{C}_{t^k,b}\mathbf{V}(\boldsymbol{\lambda})\mathbf{C}_{t^k,b}' \tag{2.44}$$

Chapter 3

Implementation

To feed the batch optimization, data needs to be collected. The content of a dataset and its implementation in MATLAB are explained first in this chapter. Then, the methods used to generate these datasets from a real blimp and from simulation are shown.

3.1 Dataset

The MATLAB implementation of the batch optimization takes a dataset as the input and returns a set of optimal parameters estimated from the given dataset.
A dataset is designed to be self contained. Each dataset includes information about the structure of the system and a set of input vectors with their corresponding measurement vectors from the sensors.
In the current implementation, the structure contains the true values[1] for the motor positions and their coordinate system transformations, the blimp radius, the blimp mass and the inertia tensor as well as transformation matrices for the sensors.
The measurement data is stored in segments. A segment is a homogeneous sequence of continuously sampled actuator feedback vectors as well as measurement vectors. Because the real system exhibits transients upon changing inputs, a boolean is also stored for each data-point which denotes whether the system has reached steady-state.
With this technique it is possible to easily cut away parts of segments which have disturbances in them while still preserving transient information.

3.2 Real System

To record data from a real blimp, experiments where conducted on the Skye System. To excite the system with actuator inputs which are suitable for batch optimization, we expanded both QGROUNDCONTROL and the Skye specific PX4 firmware with new functionality.

3.2.1 Input Patterns

We tried a number of different approaches to input patterns. There are two main goals for the design of the input pattern:

[1] True values are only known for the simulation datasets. For datasets from the real blimp, the true values are not known and therefore replaced with values based on the CAD files of the blimp.

1. The input must be **persistent of excitation**. Basically, that means that thrust magnitude and direction of the input forces must be uncorrelated.
2. The input must be **applicable**. Because there are obstacles in the surroundings of the blimp (ground, hangar walls, etc.), the movements should be as small as possible.

Another aspect of the real system is that it takes some time until the actuators have reached the desired orientation and thrust. Those transients are very hard to estimate, mainly because the thrust motor controller has no feedback and exhibits very high variability in its dynamic behaviour. In addition to the unknown thruster dynamics, the hull also reacts to load changes with vibrations that influence the IMU measurements.

By combining these constraints on the inputs, we came up with the following pattern scheme:

(i) Apply input vector to system for a fixed amount of time

(ii) Stop thrust and turn actuators in the opposite direction

(iii) Apply same input vectors in reverse for the same amount of time

Using this scheme, the angular velocity of the system yields nearly zero after the pattern and the translational velocity remains small[2].
This enables to sequentially apply multiple of those patterns before the system must be repositioned.

For the input vectors themselves, pseudo random sequences would be most promising. However, the elaboration of the optimal input sequences goes beyond the context of this thesis and we tested the following options:

(a) Generate normally distributed angular acceleration vectors and use allocation to generate input vectors for the thrusters (this leads to angular acceleration only and no translations)

(b) Use fixed thrust and select thruster angles from a uniform random number distribution

(c) Generate normally distributed thrust vectors and select thruster angles from a uniform random number distribution

It turns out, that (a) is not persistent in excitation. This actually becomes clear when having a look at $\mathbf{M}^{actuation}$ from eq. (2.2).

$$\mathbf{M}^{actuation} = \sum_{k=1}^{N} \left[\mathbf{C}_{b,m_k} \left(\mathbf{p}_{m_k}^{m_k,cog} \times \mathbf{F}_{m_k}^{k} \right) \right] \tag{3.1}$$

The allocation is basically the inversion of $\mathbf{C}_{b,m_k}(\mathbf{p}_{m_k}^{m_k,cog})^{\times}$, i.e. a mapping $\mathbb{R}^{2N} \rightarrow \mathbb{R}^3$. If the allocation is used to feed $\mathbf{F}_{m_k}^{k}$, only a subspace of the domain of $\mathbf{F}_{m_k}^{k}$ is used and the problem becomes unobservable.

From the remaining options (b) and (c), it turns out that the variant with random thrust magnitude is better than the other one, because method (b) still only excites a sub-space of the input space. This is prevented by using fully random inputs provided by method (c).

[2]The translational velocity is not totally cancelled by the reverse thrust because the system has not the same orientation during the reverse input. But since the rotations remain small and translations are strongly damped, the final translational velocity remains low.

3.2.2 Input Generation

To avoid having to re-flash the firmware all the time, the inputs are generated inside QGroundControl and directly forwarded to the actuators by the firmware of the blimp. This way, all of the parameters like standard deviation and time intervals can be set from within the custom widget in the QGroundControl interface.

Figure 3.1: QGroundControl input generation interface. Time intervals and random number specifications can easily be changed. The transmission of the input sequence can be started and stopped.

Figure 3.2: Test setup for data generation. One operator initiates the input pattern at the ground station (not visible), the second operator repositions the blimp when it approaches too close to an obstacle after some input sequence.

3.2.3 Test Setup

Although the input patterns are designed to avoid large movements as good as possible, the blimp still needs to be catched after a number of such inputs. This leads us to the basic testing setup we used to record data:

An operator is located at the ground station and a second operator is near the blimp. At the ground station there is a button to initiate one such input pattern. The second operator at the blimp catches it as soon as it drifts too close to an obstacle (see figure 3.2).
With this setup it was possible to initiate an input patterns roughly every 6 seconds. This is including the time needed to reposition the blimp when it has drifted too much.

3.2.4 Data Acquisition

For accurate feedback from the blimp the firmware was extended to offer a mode where it will transmit the relevant sensor and actuator feedback data to the ground station.
The new mode collects and transmits the following data to the ground station:

1. raw gyro
2. raw accelerometer
3. angular acceleration from EKF
4. angular rate from EKF
5. orientation quaternion from EKF
6. thrust of each actuator (from thrust signal that is passed to the thrust motor controller)
7. orientation angle of each actuator (from motor encoder)

The actuator feedback loop runs at 25 Hz. Hence the whole telemetry message is transmitted at that rate. To avoid problems with noise aliasing the sensor signals are run through a resampling filter.
QGroundControl then writes these mavlink messages into a log-file that serves as an input for generating a dataset for the optimization with Matlab.

3.2.5 Data Selection

After a raw dataset has been recorded it needs to be preprocessed to be usable for our batch optimization code. Figure 3.3 shows a period of multiple segments of the dataset and figure 3.4 shows one segment in more detail.
First after importing a raw dataset, the regions where thrust transients are to be expected are automatically marked in the dataset. This process is implemented with generous margins to eliminate problems with the batch optimization.
Then the segments where the blimp has been catched and repositioned need to be cut from the dataset.
For this purpose a semi-automatic cutting tool was implemented (see figure 3.4). The tool first identifies the segments of interest and then displays each of them to the user for visual inspection. The user can then adjust the borders of the segments to cut away any visible disturbances.
Good and bad segments are easily distinguished by looking at the angular acceleration plot. An undisturbed system shows nearly constant angular accelerations

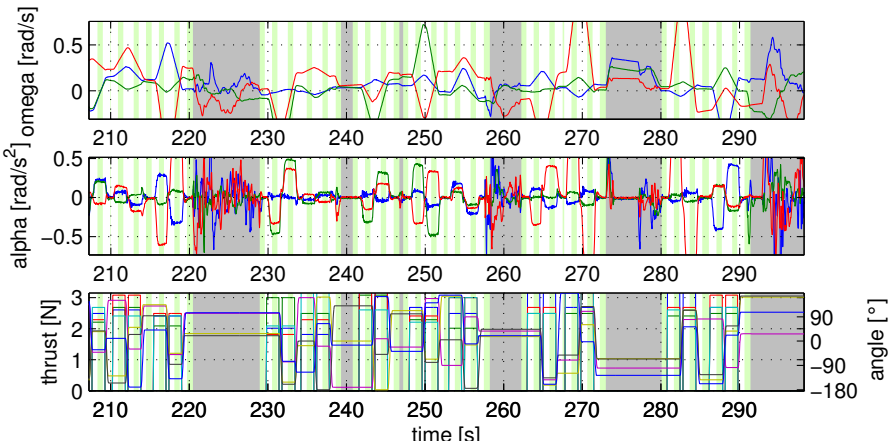

Figure 3.3: Except of dataset after transient marking. **Gray** regions are the values out interest (no inputs for a time longer than a threshold). **White** regions are the thrust transient parts which are discarded. **Green** regions are the remaining data which is used for batch optimization.

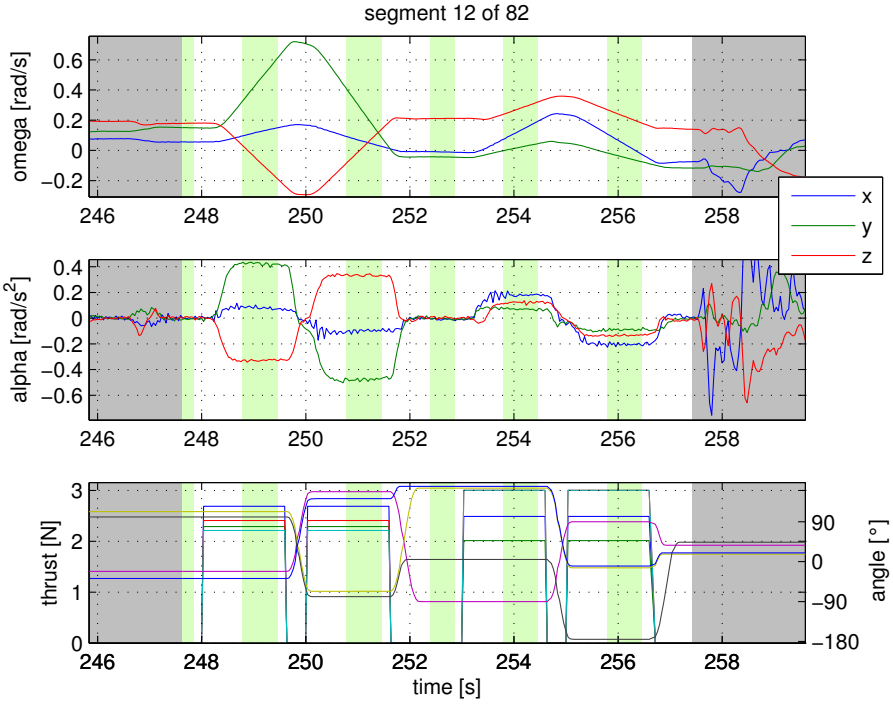

Figure 3.4: Cutting tool displaying one segment of the dataset. The **green** regions which are used for the batch optimization do only contain data with steady-state thrust. The **white** thrust transient regions are defined with a generous margin. As there is no feedback about the thrust motor, this data is discarded. Vibrations of the hull caused by load changes are also visible in these regions. Disturbances can be clearly seen in the **grey** regions.

during the periods where the thrusters are active.
Using this approach to select datapoints within a dataset, only about 50% of the raw data from the session has to be discarded. The result is a dataset with undisturbed system response including transients. When the described transient removal is applied, about 40% of the remaining data remains.
To summarize: for 10 seconds of transient-free data about 50 seconds of raw data has to be collected.

3.3 Simulation

A blimp simulator was implemented for several reasons:

- fast turn around time for generating new datasets
- disturbances (like sensor noise or aerodynamic drag) can be turned on and off
- different blimp configurations can be evaluated without having a new prototype

Combining those reasons with the properties of the data selection procedure (see section 3.2.5), the following requirements were considered for the simulation:

- include aerodynamic drag
- include sensor noise
- include thruster dynamics
- allow for easy reconfiguration of the blimp

The simulation was implemented as object oriented MATLAB code. Figure 3.5 gives an overview about the code architecture.
The blimp is built with objects that interact with each other. Each of these objects can have continuous and discrete states. The simulator then interacts with the objects over a standardized interface. At each time-step the simulator updates the states of the objects and then calls a function on each object to calculate the derivatives of the continuous states. Additionally, at the border of a discrete time step the simulator also calls a function on the objects to get the next discrete states.

3.3.1 Motion Equation

At the core of the blimp simulation is a the rigid body dynamics model. The simulator uses the same motion equation as described in table 2.1.

3.3.2 Mechanical Properties

To calculate the mechanical properties, the blimp structure is represented in a tree like data-structure (see figure 3.5). The root node of the tree is the actual rigid body object whose dynamics are simulated. Each node in the tree represents an assembly of parts.

With this methodology the blimp is actually an assembly of several parts. In general it will include a hull, the electronics unit and a number of actuation units. Each of these parts are assemblies by themselves. The electronics unit could include for example the cover, a camera and the PX4FMU, which simulates the sensors.
An actuation unit is also an assembly of at least a battery, a taring weight and an actuator which simulates the actuator dynamics and generates the forces.

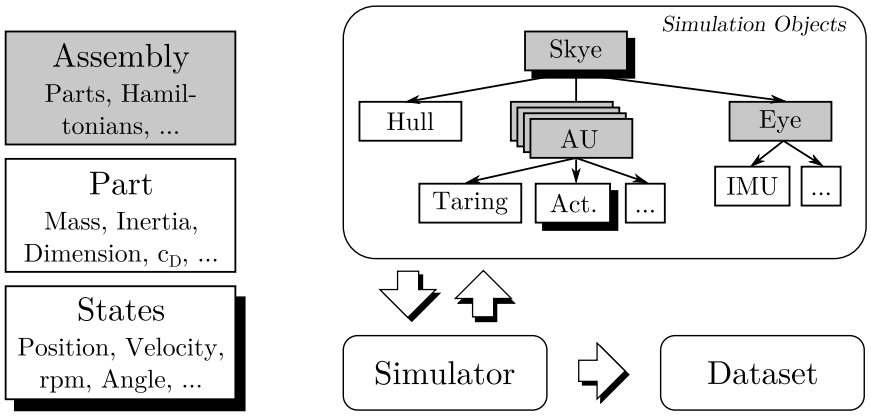

Figure 3.5: Tree like data-structure of simulation objects. Assembly parts (**gray**) have subparts (**white**) and hold additional information about their arrangement. Some parts include states (**shaded**). The simulator interacts with the simulation objects and generates a dataset.

This concept allows to have different types of components in a library which can then be used to build an arbitrary blimp.

After the blimp has been assembled, the mechanical properties of the parts at the leafs of the tree data-structure are recursively combined into their parent node. This way the mechanical properties of the root node are recalculated.
The taring routine uses the tree structure to calculate values for the taring weights located at the actuation units. After taring is done, the changed mass of the taring weights cause the tree to recalculate the mechanical properties of the root node.

3.3.3 Aerodynamic Drag

The aerodynamic effects are the most difficult part of modelling. Aerodynamic drag is usually separated into *form drag*, *wall friction drag*, *interference drag*, *lift-induced drag* and *wave drag* which are listed in decreasing importance for blimp modelling[3].

We modelled *form drag* for translational movements and *friction drag* for rotational movements as quadratic functions w.r.t. to the velocity. Further, a *virtual mass factor* was considered for the hull to consider the inertia of the surrounding fluid. *Form drag* was applied to the hull and the actuation units which are represented by cylinders with appropriate dimension. *Friction drag* was applied to the hull for rotational speed only. Together with the form drag of the actuation units, the friction drag of the hull acts as a damping moment on rotational movements.

Form drag coefficients[4] and virtual mass factor were introduced according to the literature (Kundu et al., 2012). The friction drag coefficient of the hull was determined such that the rotational resistance coincides with reference measurements.

[3] Wave drag describes compression effects of the fluid and is not relevant for subsonic speed. Lift-induced drag is also neglected because we consider blimps as symmetric profiles without considerable angle of attack. Interference drag is assumed low, as the actuation units and other components on the blimp are much smaller than the hull and is therefore also neglected.

[4] Blimps like Skye are close to the drag crisis point of spheres ($Re \approx 3e05$). Interpolation of the drag coefficient according to the actual velocity is used.

3.3.4 Thruster Dynamics

The actuator has been modelled according to test-bench measurements[5] of the thruster. The dynamic thruster model includes:

- thrust start-up delay
- thrust reaction delay
- minimum and start-up thrust thresholds
- second order thrust motor dynamics
- rotation actuator maximum speed
- second order rotation actuator dynamics

The thruster dynamics has been implemented as a blackbox model. The minimal parameters needed to describe the listed properties of the thruster dynamics are matched to the test-bench measurements using an empirical approach.
Because the thruster dynamics are not critical to our application this yields sufficient accuracy.

3.3.5 Inertial Sensors Model

Following the idea of the simulator framework the inertial sensors are also an object which is attached to the blimp assembly.
The tree-like framework of the simulation objects is bidirectional. While the mechanical properties of the leaves are used to determine the properties of the root, the motion of the rigid body at the root of the structure is distributed to each of the leaf nodes in their respective coordinate systems.
The sensor object then only has to take the already transformed accelerations, velocities and orientation and store them. In this process the sensor object also adds noise to the sensor values.

[5]The use of test-bench measurements are by courtesy of Daniel Meier.

Chapter 4

Results

In this chapter, the batch optimization results for the actuation configuration are shown for both simulation data and data from the real system. A statistical analysis of the results gives some reference for the estimation accuracy.

4.1 Comparing Results

Since the parameter space of the motor configuration is not very intuitive for error analysis, we transform the parameter results (and their variance) to a more intuitive form such that we can compare them. A good possibility is to express the actuation configuration in the tangential reference frames t^k as introduced in section 1.3.1. As the radius r is not estimated, the estimation error in $\mathbf{e}^z_{t^k}$-direction is marginal. The error of the actuator arrangement is expressed by an error distance e^k_{pos} for each motor k, which is approximated by the euclidean norm in the x-y plane of the tangential frame, and an error angle e_{angle}, which is the angle between x-axes of the 'true' and estimated motor coordinate systems $\mathbf{e}^x_{m^k}$ and $\mathbf{e}^x_{t^k}$ (see 4.1).
To express the certainty of the inertia tensor estimate, the relative error of the inertia tensor parameters will be used in figures 4.7 and 4.8.

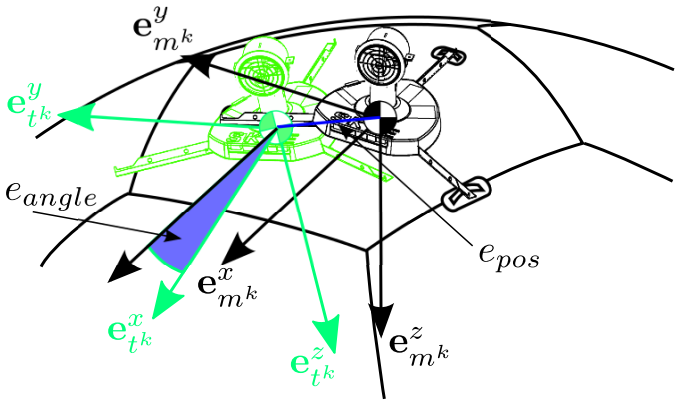

Figure 4.1: The *tangential frame* is the coordinate frame of the motor at its *true* position. If the true position is not known, an estimate is used. In any case, its $\{\mathbf{e}^x_{m^k}, \mathbf{e}^y_{m^k}\}$ plane will be tangential to the spherical hull such that an error of the arrangement can be approximated by the euclidean norm in the tangential coordinate frame.

4.2 Dataset Bootstrapping for Statistical Analysis

To be able to generate meaningful results we recorded a long dataset with Skye which contains about 850 seconds of undisturbed data samples. To improve the accuracy of statistical results bootstrapping is applied to this large dataset. Moving block bootstrapping is applied because the dataset consists of time series data. The block stride is selected such that the dataset is divided in about 16 blocks which leads to the certainty that the actual standard deviation lies within 25% the calculated standard deviation with a certainty of 95% (Wikipedia, 2014).
Unless otherwise stated, the presented statistical results imply an 25% error margin.

4.3 Simulation

For the following sections, a dataset has been generated from simulation of a blimp that is similar to Skye with a 1.37 m radius. The simulation includes aerodynamic drag as well as sensor noise. The dataset is 850 s long. The same input vectors than in the real dataset have been applied.

4.3.1 Convergence

In this section we are going to discuss the convergence properties of the batch optimization.

Convergence w.r.t. dataset length
In this thesis the datasets used for the batch optimization employ two metrics of length. Individual input vectors are applied to the system for a fixed amount of time.
The first metric of length is thus measured in number of different input vectors, *number of input vectors* for short. The second metric of length is the amount of data-samples used during each of the different input vectors, *number of samples per input vector* for short.
The number of input vectors is the more important length measure as will be shown later. It determines whether the parameters are observable at all.
The number of samples per input vector serves mainly to reduce the effect of the sensor noise on the results.

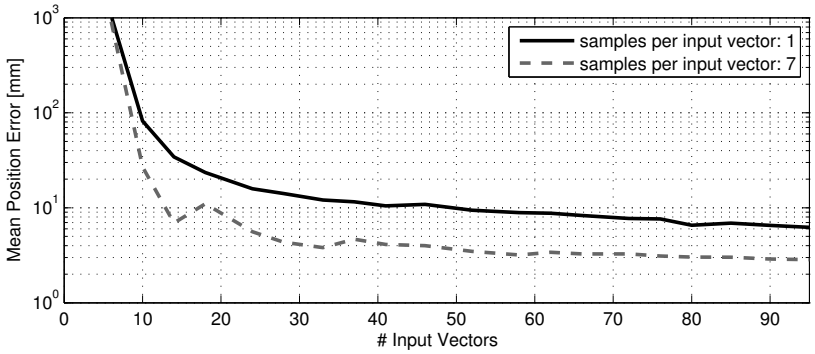

Figure 4.2: Influence of the number of different input vectors on the mean position estimate error of the four AUs e_{pos}. When using less than about 7 input vectors, no solution was found. Number of samples per input vector decreases the error by reducing the influence of sensor noise

Figure 4.2 shows the effects of different dataset lengths. Below about 7 different input vectors no solution could be found. The number of samples per input vector influences the error because the sensor noise is reduced with increasing number of samples per input vector.

Convergence w.r.t. initial parameter estimate
The impact of the initial parameter estimate on the convergence of the batch optimization has been tested. In order to test convergence, we generated a grid of initial parameters which differ from the true actuation configuration by the following two metrics:

- Position deviation of each actuator on the hull surface
- Rotation angle deviation of each actuator

The algorithm first rotates the actuator around its rotation axis by the given amount and then displaces it in a random direction on the hull surface by the requested amount.
For a grid of different position and angle deviations the failure rate was determined with ten samples each. Figure 4.3 shows the result of this test. It can be seen that an initial position offset of *all* actuators in any random direction[1] is acceptable up to 1 m. Initial orientation offsets of the motors are acceptable up to 120 °.
A similar result for the convergence region is found when using datasets from real measurements (compare figure 4.4). This outcome is as expected and means that the simulation to generate the dataset describes the real system well enough to test the batch optimization.

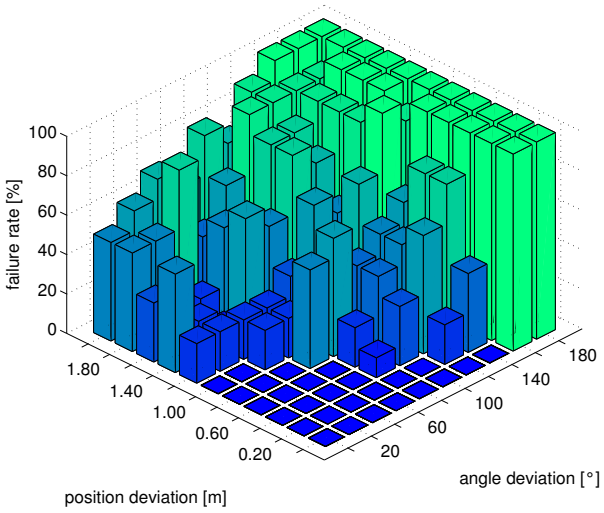

Figure 4.3: Failure rate using simulation data and varying initial guess about actuation configuration.

The convergence region depends on the 'surface' of the residual norm $\|\mathbf{r}\|$ from eq. (2.16). If there are no local minima between the initial guess and the global minimum, the optimization algorithm will find the global minimum.

[1] The simulated blimp has a hull radius of 1.37 m and therefore the distance between two of the tetrahedrally arranged actuation units is 2.62 m.

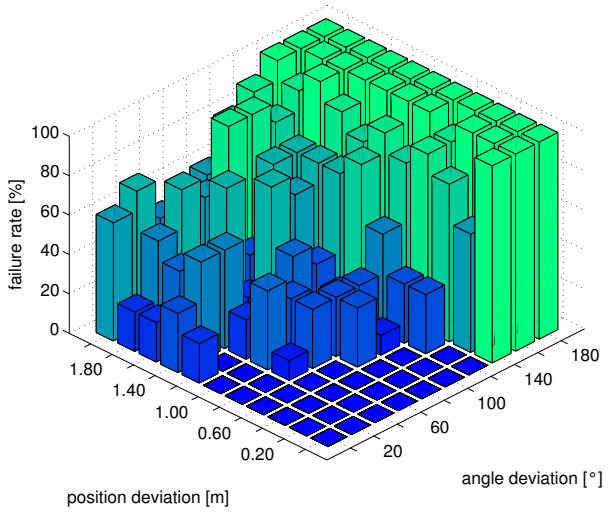

Figure 4.4: Failure rate using experiment data and varying initial guess about actuation configuration.

Figure 4.5 shows the contour of the residual norm if one of the four actuation units is deviated in position or angle. It can be seen that there is no local minimum near the global minimum. This is only the case if the configuration of the remaining actuation units is known. As is has been shown in figure 4.3, the optimization algorithm does not converge to the global minimum if *all* actuation units are deviated.

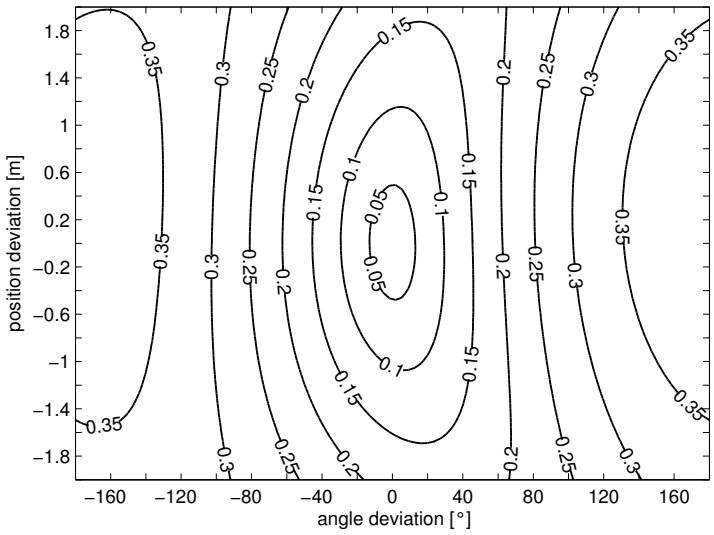

Figure 4.5: Residual norm plot for varying position and angle deviation of one actuation unit. The configuration of the other three actuation unit is set to the true value. There are no local minima within the close region to the global minimum.

4.3.2 Estimation Confidence

Methods
In this thesis two methods have been used to determine the confidence of the parameters. The first method is outlined in section 2.6 using the covariance matrices. After the batch optimization has found a set of optimal parameters, the covariance matrix of the parameters is calculated by the optimization residual. The covariance of the results is then calculated using error propagation and transformations. We will refer to this covariance as the *calculated covariance*.
The second method does not rely on any assumptions about covariance matrices and measures the parameter variance directly from multiple experimental results. We will refer to this covariance as the *measured covariance*. To get enough experimental results, bootstrapping as described in section 4.2 is used to generate 16 different datasets.

Confidence from covariance matrix
As introduced in section 4.3.1 there are two metrics of dataset length which influence the result in different ways. The same applies to the confidence calculated from the covariance matrix. When using multiple samples per input vector the covariance matrix gets over-confident. This means that the *calculated covariance* is lower than the *measured covariance*. It is intuitively clear, that if we apply multiple times the same input vectors, we will not get more information about the actuation configuration. The data samples are not independent. When using just one sample per input vector, the *calculated* and *measured covariance* match.

Confidence from multiple results
As outline above the covariance of the parameters has also been measured by evaluating multiple datasets generated from the same system and calculating the covariance for each of the parameters.

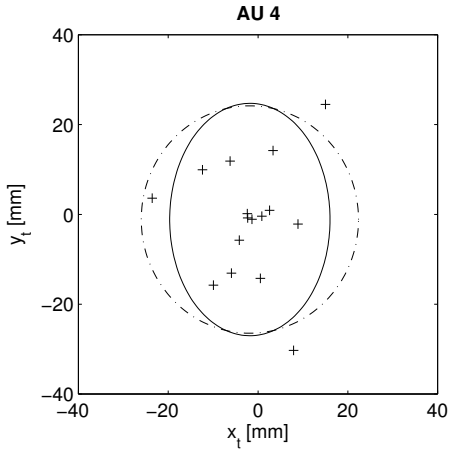

Figure 4.6: Distribution of actuator position estimates. 16 datasets with 24 input vectors each and one sample per input vector. + represent the results used to measure the variance. **dashed**: 95% confidence interval calculated from covariance matrix. **solid**: 95% confidence interval calculated from multiple results. The result is taken from actuation unit number 4. See Appendix figure B.3 for the remaining actuation units.

In figure 4.6 the *calculated covariance* is compared to the *measured covariance*. The covariances[2] are illustrated with circles that represent the region in which the parameters reside with 95% certainty. The size of the circles is determined with eq. (4.1).

$$radius = 1.96 \cdot \sqrt{variance} \tag{4.1}$$

4.3.3 Estimation Performance

To test the performance of our optimization framework, we defined five test cases.

Default A simulation of a blimp that is comparable to the real Skye system. The positions of the actuation units are taken from CAD data of Skye. All 21 parameters as listed in table 2.3 are estimated. This is the same test case as in the previous sections.

1 AU Same blimp as in default, but only one of the AUs is estimated (AU 4). The remaining AUs are not powered at all. 12 parameters estimated.

5 AU Different blimp with 5 AUs. Configuration for all AUs estimated. 24 parameters.

No Drag Same blimp as in default, but simulation without aerodynamic drag. This is used to see how much drag influences the result accuracy.

COG Same blimp as in default, but with large additional weight at AU4 position. COG to COB offset is 10cm and yields a moment in the same magnitude as the actuation moment. This is used to estimate the error introduced by large COG shifts.

For each of these test cases the RMS error for 4 different parameter measures has been calculated. The samples for the RMS calculations are generated with the same method as described in section 4.2. The analyzed parameters are:

1. Position error of AU 4 in the tangential frame
2. Angle error of AU 4 around the z-axis in the tangential frame
3. COG offset error in the blimp frame
4. Inertia tensor parameter vector error in percent

Figure 4.7 shows the results of this analysis.

[2] Actually just the diagonal entries of the covariance matrix are used to draw the ellipses.

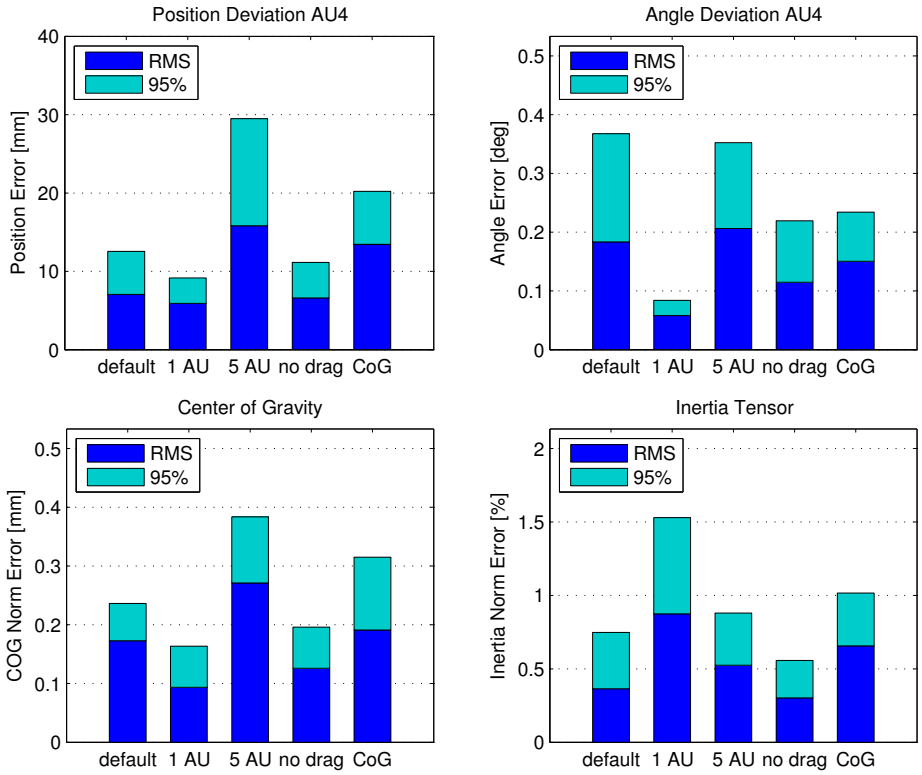

Figure 4.7: Results for simulator

Here is a list of significant observations:

- The position estimation is accurate within centimeters
- The angle error is lower than 0.5 °
- The center of gravity estimation is accurate within millimetres

A variation of the system model parameters by the scale of their RMS accuracy induces a change of the angular acceleration of 0.2% on average.
Further observations include:

- When feeding the same number of data-points to the batch optimization, the estimation error is higher when more parameters need to be estimated (default $\leftrightarrow$ 1 AU $\leftrightarrow$ 5 AU)
- Omitting aerodynamic drag from simulation does not improve the results by much. This supports our argumentation that we can neglect the aerodynamic effects on the rotations.
- Although a big COG shift decreases the accuracy of the actuator positions, the COG itself is still very accurately estimated. The increased uncertainty in the parameter estimates using big COG shift is due to the system model assumption that the actuation units are placed spherical around the COG. This assumption only holds if the COG shift is zero. However, a system like used in this simulation with a COG shift of 10 cm is not applicable for Skye anyway as the moment released by the COG offset exceeds the actuation power.

- For the 1 AU case the inertia tensor error is bigger because a single actuator can excite the system only around two axes. This yields to an estimate of the inertia tensor that is much less accurate.

4.4 Experiments

The data in this section was recorded with the Skye blimp as described in section 3.2. For statistics, instead of calculating the RMS error we calculated the standard deviation of the results of multiple data sets. Standard deviation was chosen over RMS error because coming up with a bias-free ground-truth is hard. This will be discussed in more detail in section 4.5.
By using the standard deviation as a estimation performance metric we implicitly assume that the mean of all estimations lies at the true value. With this setup it is not possible to measure estimator bias. We will show a different performance measure in section 4.4.2 which shows that the actuator configuration from the estimation result is more accurate than the default model we used for the simulation.

4.4.1 Estimation Performance

In figure 4.8 the results of the real dataset are compared to the results of the simulation dataset. The variance of the parameters is larger than in the simulation, but in the same order of magnitude. E.g., the estimate of the actuator positions is accurate within 2 cm.

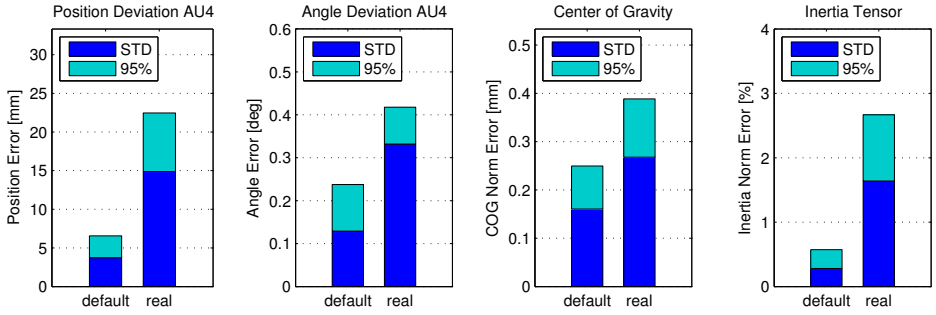

Figure 4.8: Results for real system compared with the (default) simulation

4.4.2 Simulation Improvement

To test the influence of the optimized parameters on the system dynamics, one part of the dataset has been chosen for validation. *First*, the same inputs as in the real dataset were used for a simulation with the actuator configuration known from CAD data of the prototype[3]. *Second*, the inputs were applied to a simulation with the optimized actuator configuration.
In figure 4.9, the angular acceleration of the real dataset is compared to the two simulation cases. During the steady state sections of the thrusters (green shaded), the error of the angular acceleration is reduced by more than a factor of two. As the transient behaviour of the thrusters was not included in the optimization, the according parts of the simulation was not improved much.
Nevertheless, as the thruster dynamics is not considered in control so far, the knowledge about it would not help to increase the control performance.

[3] This information about actuator configuration is recently used on Skye system for control.

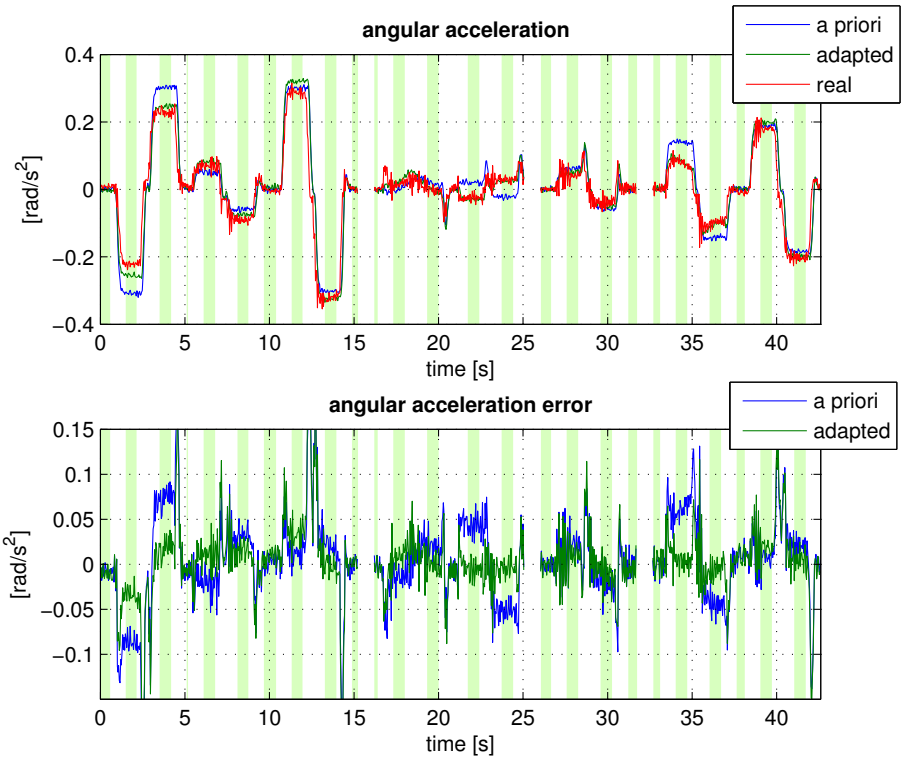

Figure 4.9: Validation section of the dataset. **Top**: Blimp angular acceleration of the real dataset (**red**), simulation with (a priori) CAD knowledge about actuation configuration (**blue**), and simulation with (adapted) optimized actuation configuration knowledge (**green**). **Bottom**: Blimp angular acceleration difference between real dataset and the two simulation cases. During thruster steady-state (**green shaded**), the simulation error using the adapted actuation configuration (**green**) is smaller than using the a priori estimate (**blue**). During the thruster transients (**plain**), the error of both simulations is larger.

4.5 Ground Truth

For the experiments, the true actuator configuration is not available. To get at least some reference values, we measured the position of the actuation units with a Leica Viva TS15 laser tracking system.
The setup is shown in figure 4.10. With the tracking system, only three actuators

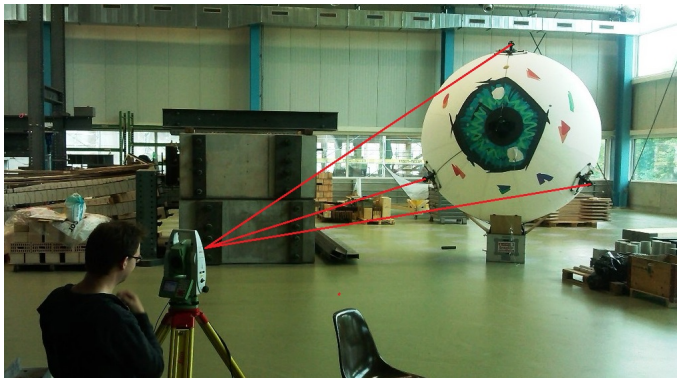

Figure 4.10: Setup with the Leica tracking system. Three of the four actuation units can be seen at once. To track the relative position of all of them, the blimp was rotated such that another combination of three actuators was visible.

can be seen at once. To track all of the actuators, multiple measurements from different views have been made. As a detectable feature, the center of the actuator's rotor has been used (see figure 4.11).

Figure 4.11: The center of the rotor has been used for tracking.

The actuator had to be rotated such that this point was visible for the current view. The resulting offset to the actuators z-axis was considered in the optimization problem, which was formulated to calculate the distance between the actuator units. This means only the relative arrangement of the actuators was measured as 'ground-truth'[4]. The relative difference between the distance calculated from the Leica measurements and the distance calculated from our work are between 0.7 and 3.8%. Nevertheless, a measurement of the actuator arrangement in the geometric space is disadvantaged to an estimate found using the onboard sensors for the use in the controller because the latter ones are also used for control.

[4] The absolute configuration of the actuators with respect to the IMU (which is essential for the control) would require the precise measurement of the position and orientation of the IMU too.

Chapter 5

Conclusion

In this thesis we showed a working method to determine the actuation configuration of a spherical blimp. On a real system like Skye, the presented method can determine the actuator configuration with a centimeter accuracy. Additionally hard to measure parameters like the inertia tensor and the offset between COG and COB can be determined as well. To get to this point we showed a possible parametrization of a blimp system model. We then broke down which sensors to use and briefly examined which input vectors can be used to excite the system for identification. We then implemented a MATLAB framework including a modular blimp simulator to test several different optimization algorithms. The final implementation uses Levenberg-Marquardt method for parameter optimization which showed fast and robust convergence.

5.1 Application

There are two immediate applications for the results of this thesis. The estimated actuation configuration can be implemented in a configuration chain that updates the allocation matrix. The estimate of the COG offset can be used to calculate the optimal placement of the taring weights.

Automatic Allocation Generation
The procedure for automatic allocation generation can be realized as follows:

1. Prepare Skye for flight and place it in a big open space
2. Start input sequence in user interface and record feedback data
3. Run MATLAB batch optimization algorithm
4. Upload actuation configuration data via user interface to Skye
5. Recalculate allocation matrix on firmware

Using the recent input pattern, step 2. must be repeated iteratively and the system has to be caught when it drifts towards an object. To get enough data for accurate configuration estimation, at least about 16 different inputs are recommended. It takes about two minutes to record this amount of data. This is mainly due to the transient behaviour of the trusters.

Automatic Taring Calculation
If the displacement between the center of gravity and the center of buoyancy $\mathbf{p}^{cob,cog}$ is known, the required mass for the taring weights m^{tar^k} at the given positions $\mathbf{p}^{tar^k}$ can be calculated to eliminate the COG offset. This can be achieved by solving the following linear system of equations:

$$\begin{bmatrix} \mathbf{p}^{tar^1} & \mathbf{p}^{tar^2} & \cdots \\ 1 & 1 & \cdots \end{bmatrix} \begin{bmatrix} m^{tar^1} \\ m^{tar^2} \\ \vdots \end{bmatrix} = \begin{bmatrix} m^{blimp}\mathbf{p}^{cob,cog} \\ m^{taring\ total} \end{bmatrix} \tag{5.1}$$

$m^{taring\ total}$ can be determined by measuring the uplift of the blimp.

5.2 Acknowledgement

We would like to express our gratitude to Prof. Dr. Roland Yves Siegwart for enabling this thesis and his commitment for the Skye project. We would also like to thank our supervisors Kostas Alexis and Markus Achtelik for their profound advice and their critical view to our ideas.
Further we want to express our gratitude to Dominik Werner and his team at the ETH D-BAUG Bauhalle for providing us a great hangar to fly Skye. And finally our thanks goes to all those people who pushed the Skye project to where it is today. Especially we thank Lukas Gasser, Miro Käch, Daniel Meier and Andreas Schaffner for their dedication in enabling this unique aerial platform.

Appendix A

Derivatives

A.1 Derivative of Cross Product

The derivative of a cross product can be calculated analogue to matrix derivatives. The skew symmetric cross matrix is

$$\mathbf{a}^{\times} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{A.1}$$

and the appropriate derivatives are therefore

$$\frac{\partial}{\partial \mathbf{a}} (\mathbf{a} \times \mathbf{b}) = -\mathbf{b}^{\times} \tag{A.2}$$

and

$$\frac{\partial}{\partial \mathbf{a}} (\mathbf{b} \times \mathbf{a}) = \mathbf{b}^{\times} \tag{A.3}$$

A.2 Derivative w.r.t. Gibbs-Rodriquez Parameters

Derivative of the rotation matrix eq. (2.9) w.r.t. the Gibbs-Rodriquez parameter:

$$\frac{\partial \mathbf{C}}{\partial \lambda_1} = \begin{pmatrix} \frac{4\,\lambda_1\left(\lambda_2{}^2+\lambda_3{}^2\right)}{\nu}^2 & \frac{2\,\lambda_2}{\nu}+\frac{4\,\lambda_1\,\lambda_3-\lambda_1\,\lambda_2}{\nu^2} & \frac{2\,\lambda_3}{\nu}-\frac{4\,\lambda_1\,(\lambda_2+\lambda_1\,\lambda_3)}{\nu^2} \\ \frac{2\,\lambda_2}{\nu}-\frac{4\,\lambda_1\,(\lambda_3+\lambda_1\,\lambda_2)}{\nu^2} & \frac{4\,\lambda_1\left(\lambda_1{}^2+\lambda_3{}^2\right)}{\nu^2}-\frac{4\,\lambda_1}{\nu} & \frac{4\,\lambda_1\,(\lambda_1-\lambda_2\,\lambda_3)}{\nu^2}-\frac{2}{\nu} \\ \frac{2\,\lambda_3}{\nu}+\frac{4\,\lambda_1\,(\lambda_2-\lambda_1\,\lambda_3)}{\nu^2} & \frac{2}{\nu}-\frac{4\,\lambda_1\,(\lambda_1+\lambda_2\,\lambda_3)}{\nu^2} & \frac{4\,\lambda_1\left(\lambda_1{}^2+\lambda_2{}^2\right)}{\nu^2}-\frac{4\,\lambda_1}{\nu} \end{pmatrix} \tag{A.4}$$

$$\frac{\partial \mathbf{C}}{\partial \lambda_2} = \begin{pmatrix} \frac{4\,\lambda_2\left(\lambda_2{}^2+\lambda_3{}^2\right)}{\nu^2}-\frac{4\,\lambda_2}{\nu} & \frac{2\,\lambda_1}{\nu}+\frac{4\,\lambda_2\,(\lambda_3-\lambda_1\,\lambda_2)}{\nu^2} & \frac{2}{\nu}-\frac{4\,\lambda_2\,(\lambda_2+\lambda_1\,\lambda_3)}{\nu^2} \\ \frac{2\,\lambda_1}{\nu}-\frac{4\,\lambda_2\,(\lambda_3+\lambda_1\,\lambda_2)}{\nu^2} & \frac{4\,\lambda_2\left(\lambda_1{}^2+\lambda_3{}^2\right)}{\nu^2} & \frac{2\,\lambda_3}{\nu}+\frac{4\,\lambda_2\,(\lambda_1-\lambda_2\,\lambda_3)}{\nu^2} \\ \frac{4\,\lambda_2\,(\lambda_2-\lambda_1\,\lambda_3)}{\nu^2}-\frac{2}{\nu} & \frac{2\,\lambda_3}{\nu}-\frac{4\,\lambda_2\,(\lambda_1+\lambda_2\,\lambda_3)}{\nu^2} & \frac{4\,\lambda_2\left(\lambda_1{}^2+\lambda_2{}^2\right)}{\nu^2}-\frac{4\,\lambda_2}{\nu} \end{pmatrix} \tag{A.5}$$

$$\frac{\partial \mathbf{C}}{\partial \lambda_3} = \begin{pmatrix} \frac{4\,\lambda_3\left(\lambda_2{}^2+\lambda_3{}^2\right)}{\nu^2}-\frac{4\,\lambda_3}{\nu} & \frac{4\,\lambda_3\,(\lambda_3-\lambda_1\,\lambda_2)}{\nu^2}-\frac{2}{\nu} & \frac{2\,\lambda_1}{\nu}-\frac{4\,\lambda_3\,(\lambda_2+\lambda_1\,\lambda_3)}{\nu^2} \\ \frac{2}{\nu}-\frac{4\,\lambda_3\,(\lambda_3+\lambda_1\,\lambda_2)}{\nu^2} & \frac{4\,\lambda_3\left(\lambda_1{}^2+\lambda_3{}^2\right)}{\nu^2}-\frac{4\,\lambda_3}{\nu} & \frac{2\,\lambda_2}{\nu}+\frac{4\,\lambda_3\,(\lambda_1-\lambda_2\,\lambda_3)}{\nu^2} \\ \frac{2\,\lambda_1}{\nu}+\frac{4\,\lambda_3\,(\lambda_2-\lambda_1\,\lambda_3)}{\nu^2} & \frac{2\,\lambda_2}{\nu}-\frac{4\,\lambda_3\,(\lambda_1+\lambda_2\,\lambda_3)}{\nu^2} & \frac{4\,\lambda_3\left(\lambda_1{}^2+\lambda_2{}^2\right)}{\nu^2} \end{pmatrix} \tag{A.6}$$

where

$$\nu = \lambda_1{}^2 + \lambda_2{}^2 + \lambda_3{}^2 + 1 \tag{A.7}$$

Appendix B

Additional Plots

B.1 Sensor Specification

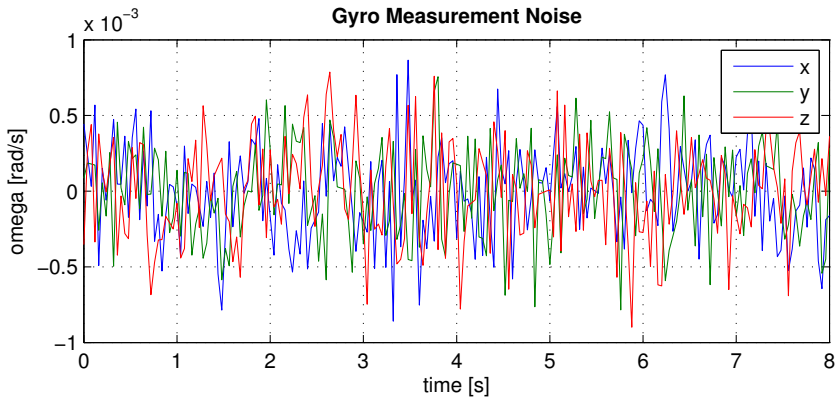

Figure B.1: Measurement noise of MPU-6000 gyro. The RMS error is about $3.2 \cdot 10^{-4} rad/s$.

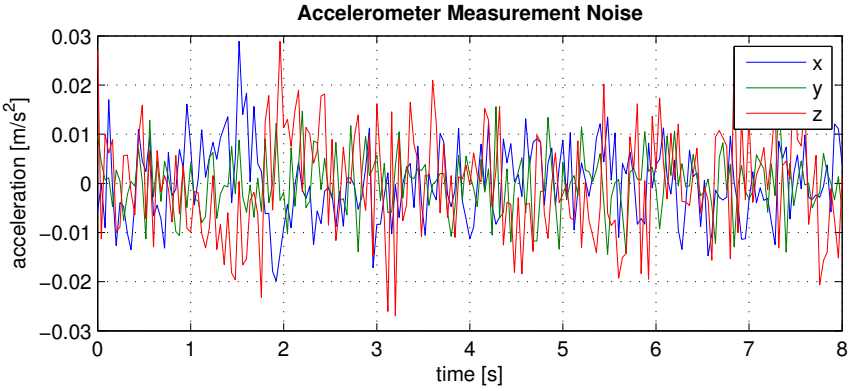

Figure B.2: Measurement noise of MPU-6000 accelerometer. The RMS error is about $8.2 \cdot 10^{-3} m/s^2$.

B.2 Actuation Unit Position Estimate Variance

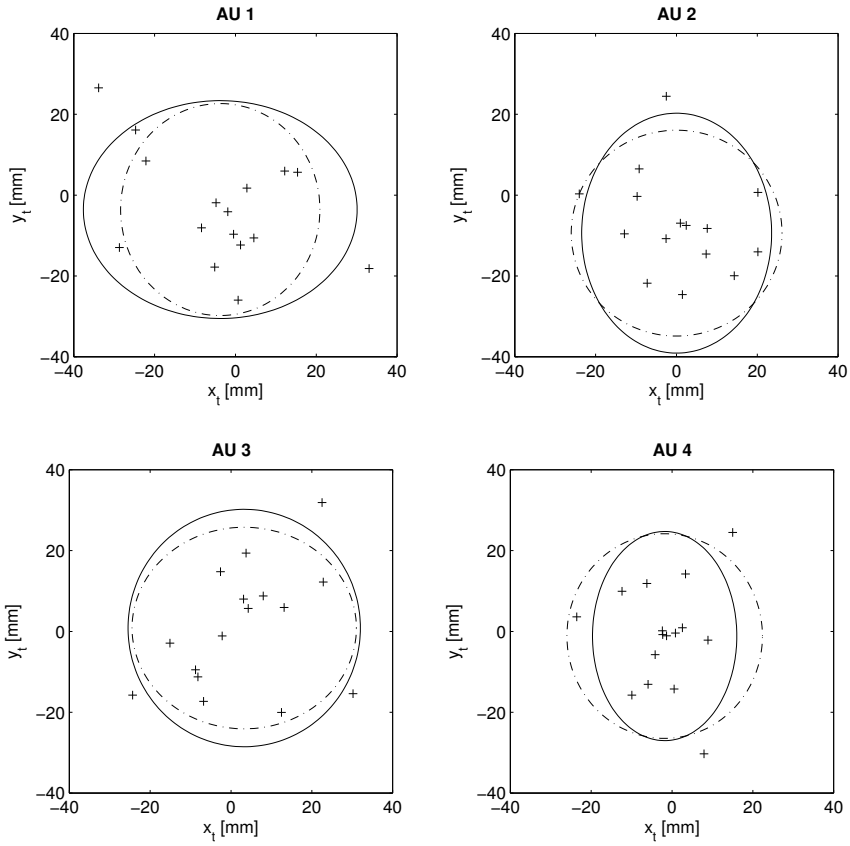

Figure B.3: Distribution of actuator position estimates from simulation dataset. 16 datasets with 24 input vectors each and one sample per input vector. + represent the results used to measure the variance. **dashed**: 95% confidence interval calculated from covariance matrix. **solid**: 95% confidence interval calculated from multiple results.

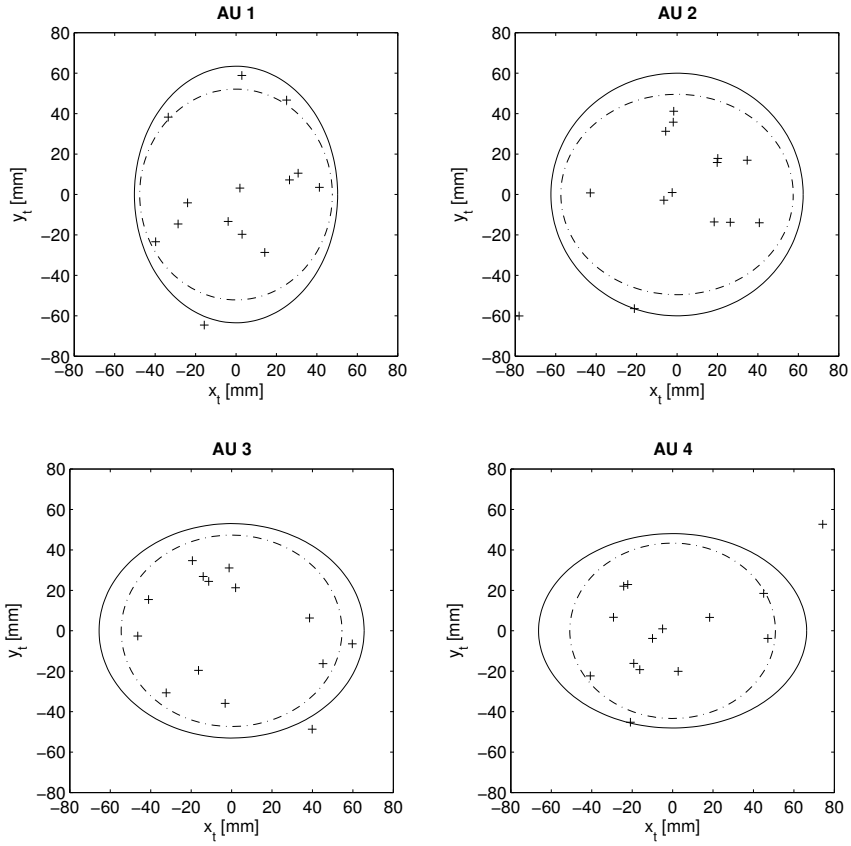

Figure B.4: Distribution of actuator position estimates from real dataset. 16 datasets with 24 input vectors each and one sample per input vector. + represent the results used to measure the variance. **dashed**: 95% confidence interval calculated from covariance matrix. **solid**: 95% confidence interval calculated from multiple results.

B.3 Parameter Estimates

Table B.1: Optimization result of the parameters as defined in section 2.2, table 2.3 for simulation data. Results have been calculated for 16 distinct datasets. **Mean** is the mean parameter estimate of the samples with standard deviation **Std1**. The mean over the expected standard deviation from the optimization residual is given as **Std2**.

Variable	Parameter	Mean	Std1	Std2	Unit
AU 1 Orientation	λ_1^1	-1.742	0.0092	0.0125	[−]
	λ_1^1	-0.177	0.0027	0.0050	[−]
	λ_1^1	0.306	0.0058	0.0058	[−]
AU 2 Orientation	λ_1^2	1.731	0.0092	0.0124	[−]
	λ_1^2	-0.175	0.0044	0.0051	[−]
	λ_1^2	-0.306	0.0053	0.0060	[−]
AU 3 Orientation	λ_1^3	0.001	0.0030	0.0030	[−]
	λ_1^3	-0.177	0.0020	0.0034	[−]
	λ_1^3	0.000	0.0019	0.0022	[−]
AU 4 Orientation	λ_1^3	-0.001	0.0018	0.0037	[−]
	λ_1^3	1.002	0.0038	0.0059	[−]
	λ_1^3	-0.001	0.0018	0.0036	[−]
Inertia Tensor	J_1	14.574	0.0239	0.0361	kg m^2
	J_2	15.564	0.0219	0.0379	kg m^2
	J_3	15.475	0.0350	0.0391	kg m^2
	J_4	0.072	0.0245	0.0210	kg m^2
	J_5	0.215	0.0173	0.0203	kg m^2
	J_6	0.047	0.0085	0.0199	kg m^2
Center of Gravity	$\mathbf{p}_{b,x}^{cob,cog}$	-0.102	0.1114	0.1530	mm
	$\mathbf{p}_{b,y}^{cob,cog}$	-0.025	0.1502	0.1493	mm
	$\mathbf{p}_{b,z}^{cob,cog}$	-0.003	0.0990	0.1478	mm

Table B.2: Optimization result of the parameters as defined in section 2.2, table 2.3 for real data. Results have been calculated for 16 distinct datasets. **Mean** is the mean parameter estimate of the samples with standard deviation **Std1**. The mean over the expected standard deviation from the optimization residual is given as **Std2**.

Variable	Parameter	Mean	Std1	Std2	Unit
AU 1 Orientation	λ_1^1	-1.842	0.0390	0.0465	[−]
	λ_2^1	-0.253	0.0134	0.0155	[−]
	λ_3^1	0.270	0.0161	0.0161	[−]
AU 2 Orientation	λ_1^2	1.727	0.0330	0.0414	[−]
	λ_2^2	-0.018	0.0136	0.0125	[−]
	λ_3^2	-0.335	0.0180	0.0212	[−]
AU 3 Orientation	λ_1^3	-0.010	0.0079	0.0091	[−]
	λ_2^3	-0.183	0.0093	0.0114	[−]
	λ_3^3	-0.006	0.0054	0.0044	[−]
AU 4 Orientation	λ_1^4	-0.104	0.0088	0.0097	[−]
	λ_2^4	0.985	0.0168	0.0209	[−]
	λ_3^4	-0.097	0.0092	0.0107	[−]
Inertia Tensor	J_1	14.709	0.1356	0.2695	kg m^2
	J_2	17.218	0.1759	0.3285	kg m^2
	J_3	16.405	0.1685	0.4151	kg m^2
	J_4	0.512	0.1351	0.1366	kg m^2
	J_5	-0.060	0.1261	0.1460	kg m^2
	J_6	0.234	0.1267	0.1342	kg m^2
Center of Gravity	$\mathbf{p}_{b,x}^{cob,cog}$	0.495	0.2842	0.2695	mm
	$\mathbf{p}_{b,y}^{cob,cog}$	-0.706	0.2679	0.2285	mm
	$\mathbf{p}_{b,z}^{cob,cog}$	-0.215	0.3171	0.3384	mm

Bibliography

Blimpworks. `http://rcblimp.net`, 2014. [Online; Retrieved 11-June-2014].

Michael Bloesch, Marco Hutter, Christian Gehring, Mark A Hoepflinger, and Roland Siegwart. Kinematic Batch Calibration for Legged Robots. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2542–2547, May 2013. ISBN 9781467356428.

M. Burri, L. Gasser, M. Kach, M. Krebs, S. Laube, A. Ledergerber, D. Meier, R. Michaud, L. Mosimann, L. Muri, C. Ruch, A. Schaffner, N. Vuilliomenet, J. Weichart, K. Rudin, S. Leutenegger, J. Alonso-Mora, R. Siegwart, and P. Beardsley. Design and control of a spherical omnidirectional blimp. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1873–1879, Nov 2013.

Marek Doniec, Carrick Detweiler, and Daniela Rus. Estimation of thruster configurations for reconfigurable modular underwater robots. In Oussama Khatib, Vijay Kumar, and Gaurav Sukhatme, editors, *Experimental Robotics*, volume 79 of *Springer Tracts in Advanced Robotics*, pages 655–666. Springer Berlin Heidelberg, 2014. ISBN 978-3-642-28571-4.

Norman R. Draper and Harry Smith. *Applied Regression Analysis*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, New Jersey, 3rd edition, 1998. ISBN 978-0-471-17082-2.

Paul Furgale, Timothy D. Barfoot, and Gabe Sibley. Continuous-time batch estimation using temporal basis functions. *2012 IEEE International Conference on Robotics and Automation*, pages 2088–2095, May 2012. doi: 10.1109/ICRA.2012.6225005.

R. Hermann and Arthur J. Krener. Nonlinear controllability and observability. *Automatic Control, IEEE Transactions on*, 22(5):728–740, Oct 1977.

Jeroen Hol. *Sensor Fusion and Calibration of Inertial Sensors , Vision , Ultra-Wideband and GPS*. PhD thesis, Linköping University, 2011.

Matthias Krebs and Anton Ledergerber. Human machine interface for operating a blimp. Bsc thesis, ETH Zurich, 2012.

Pijush K. Kundu, Ira M. Cohen, and David R. Dowling. *Fluid Mechanics*. Elsevier Science, 5th edition, 2012. ISBN 9780123821003.

Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2): 431–441, 1963.

Pixhawk. Pixhawk research project. `https://pixhawk.ethz.ch`, 2014. [Online; Retrieved 11-June-2014].

Andreas Schaffner and Nicolas Vuilliomenet. Actuation chain design and optimization. Bsc thesis, ETH Zurich, 2012.

George A. F. Seber and C. J. Wild. *Nonlinear Regression*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, New Jersey, 2003. ISBN 978-0-471-47135-6.

Malcolm D. Shuster, Daniel S. Pitone, and Gerald J. Bierman. Batch Estimation of Spacecraft Sensor Alignments I. Relative Alignment Estimation. *Journal of Astronautical Sciences*, 39:519–546, 1991.

Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. The MIT Press, 2nd edition, 2011. ISBN 0262015358, 9780262015356.

Pepijn W.J. van de Ven, Colin Flanagan, and Daniel Toal. Neural network control of underwater vehicles. *Engineering Applications of Artificial Intelligence*, 18(5): 533–547, August 2005.

Johannes Weichart. Agile spherical blimp modeling and simulation environment. Bsc thesis, ETH Zurich, 2012.

Wikipedia. Margin of error — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Margin_of_error`, 2014. [Online; Retrieved 11-June-2014].