DEGREE PROJECT IN ELECTRICAL ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2017

Smoothing and Mapping of an Unmanned Aerial Vehicle Using Ultra-wideband Sensors

ERIK STRÖMBERG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING

Smoothing and Mapping of an Unmanned Aerial Vehicle Using Ultra-wideband Sensors

Erik Strömberg

Examiner: Karl Henrik Johansson
KTH Supervisor: Jaehyun Yoo
Ericsson Supervisors: Lars AA Andersson, José Araújo
Thesis Opponent: Florian Curinga

Stockholm, October 2017

Master Degree Project in Automatic Control
School of Electrical Engineering
KTH Royal Institute of Technology

Abstract

Unmanned Aerial Vehicles (UAV), in particular the four-rotor quadrotor, are gaining wide popularity in research as well as in commercial and hobbyist applications. Maneuvrability, low cost, and small size make quadrotors an attractive option to full-scale, manned helicopters while opening up new possibilities. These include applications where full-scale helicopters are unsuitable, such as cooperative tasks or operating indoors.

Many UAV systems use the Global Positioning System (GPS), IMU (Inertial Measurement Unit) sensors, and camera sensors to observe the UAV's state. Depending on the application, different methods for observing the states are suitable. Outdoors, GPS is available and widely used and in cluttered environments on-board cameras can be the best choice. Controlled lab environments often use external cameras to track the quadrotor. Most applications make use of the IMU in their implementations, most commonly the gyroscope for attitude estimation.

In this thesis, several external ultra-wideband (UWB) radio sensors are used to measure the distance between individual sensors and a quadrotor. The range measurements are fused with acceleration and angular velocity measurements from an Inertial Measurement Unit to estimate the quadrotors position and attitude. An ultra-wideband sensor is cheap and does not require line-of-sight or heavy equipment mounted on the quadrotor. The drawback of UWB-based positioning is that it requires the assumption of known sensor locations in order to calculate the distance between sensor and UAV. This thesis aims to remove this assumption by estimating the quadrotor's and the sensors' position simultaneosly using the Smoothing and Mapping (SAM) technique.

The Georgia Tech Smoothing and Mapping (GTSAM) framework provides the incremental Smoothing and Mapping implementation, used for estimation of both the quadrotor's position and attitude, and the sensors' position. The Inertial Measurement Unit is handled by the state of the art IMU factor, included in GTSAM.

The system is evaluated with and without prior knowledge of the sensor positions, using recorded flight data from a Crazyflie quadrotor and six Loco Positioning Node sensors. The results show that the system is able to track the UAV's position and attitude with acceptable errors. The error in estimated sensor position is too large to be satisfactory, Based on the results several topics for future work are proposed.

Sammanfattning

Obemannade luftfarkoster (UAV), i synnerhet den fyra rotorer försedda quadrotorn, har vunnit en bred populäritet inom så väl forskningen som för kommersiella och hobbyapplikationer. Manövrerbarhet, låg kostnad och en liten storlek gör quadrotorerna till ett attraktiv alternativt till fullskaliga, bemannade helikoptrar. Samtidigt öppnar de upp för nya applikationer där konventionella luftfarkoster inte är lämpade, såsom att samarbeta eller flyga inomhus.

Beroende på applikation är olika metoder för att lokalisera quadrotorn lämpliga. Utomhus är det globala positions systemet (GPS) tillgängligt och vida använt, och i utrymmen med hinder passar det bra med kamerasensorer som monteras på farkosten. I kontrollerade miljöer, såsom i laboratorium, används ofta externa kameror för att följa quadrotorn. De flesta implementationer har gemensamt att de använder mätningar från en tröghetsmätningsenhet (IMU).

I det här arbetet används flera externa ultrabredbandiga radioenheter (UWB) som sensorer för att mäta avståndet mellan sensorerna och quadrotorn. Avståndsmätningarna läggs samman med uppmätt acceleration och vinkelhastighet från en IMU för att uppskatta quadrotorns position och orientering. Radiosensorn är billig, kräver inte fri sikt mellan sensor och quadrotor och ej heller annan tung utrustning monterad på quadrotorn. Nackdelen är att sensorernas positioner måste vara kända för att kunna beräkna avståndet. Det här arbetet har som mål att ta bort behovet genom att uppskatta quadrotorns och sensorernas vardera positioner samtidigt genom att använda metoden utjämning och kartläggning (SAM).

Mjukvaruramverket Georgia Tech Smoothing and Mapping (GTSAM) tillhandahåller en inkrementell utjämning och kartläggningsimplementation som här används för att uppskatta quadrotorns position och orientering samt sensorernas positioner. Tröghetsmätningsenheten hanteras av den toppmoderna IMU-faktorn som ingår i GTSAM.

Systemet utvärderas såväl utan som med tidigare vetskap om sensorernas positioner. Inspelad data är inhämtad från verkliga flygningar med quadrotorn Crazyflie och sex stycken Loco Positioning Node-sensorer. Resultaten visar att systemet kan estimera position och orientering med godtagbart fel när sensorernas initiala position är okänd. Felet i den uppskattade sensorpositionen är däremot för stort för att vara tillfredsställande. Baserat på dessa resultat dras en slutsats och flera områden för fortsatta studier föreslås.

Acknowledgements

I would like to thank my supervisors Lars Andersson and José Araújo, at Ericsson, for their support and invaluable guidance as well as for the opportunity to do my thesis with them. Thanks to Karl Henrik Johansson accepting to be my examiner, my supervisor Jaehyun Yoo for his advice, Antonio Adaldo for helping me with hardware related grievances, Arnaud Taffanel at Bitcraze for prompt firmware support, and Paul Rousse for helping me with lab issues. Also thanks to Wolfgang Hönig at USC and Michael Hamer at ETH Zürich for taking their time with my questions and remarks, and Joakim Blikstad for finding two critical bugs in my code. Thanks to the Smart Mobility Lab, and especially Jonas Mårtensson, for hosting me and my quadrotors.

Contents

Chapter 1

Introduction

Unmanned Aerial Vehicles (UAV), such as the quadrotor, have gained increased popularity in both research and among the public. Their popularity stems from their maneuverability, small size, and low cost, making them more accessible than a full-scale helicopter. UAV applications include inspection of inaccessible structures, such as wind turbines or bridges, and surveying.

Due to the quadrotor's above mentioned traits, they are suitable for indoors applications, where manned helicopters clearly are not. This opens up for applications such as transportation of goods in large warehouses, or traveling between different indoor areas. This, however, poses another problem. In order to autonomously fly and navigate the aircraft, it's position is commonly used as a reference. Outdoors, this can be solved with a Global Positioning System (GPS) receiver, which, however, requires line-of-sight with the GPS satellites. Indoors, infrared (IR) camera systems for localization are common in research. Though they are very accurate, they are also expensive, suffers from occlusion and are not suitable for large areas.

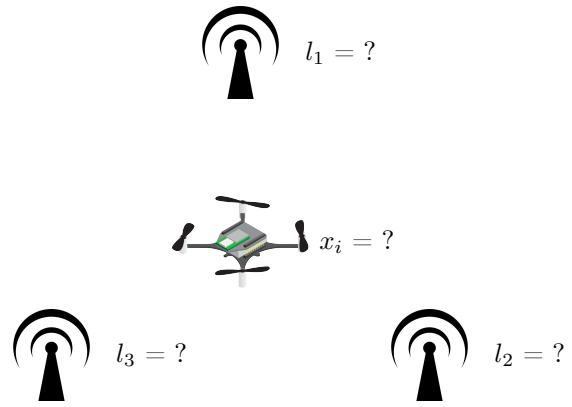


Figure 1.1: The problem visualized with three sensors of unknown position. Quadrotor image courtesy of Bitcraze AB

This thesis will investigate a solution consisting of several ultra-wideband (UWB) transceivers, acting as stationary sensors in order to localize and track

the quadrotor. In current applications of UWB-based systems the exact location of the sensors have to be known in advance, limiting the accuracy of the system to that of the initial position measurements of the sensors. The aim in this thesis is to remove this drawback. See Figure 1.1 for a visual description of the problem. In literature, these types of problems are called Simultaneous Localization and Mapping (SLAM). The idea is to learn the location of both the UWB sensors and the agent without prior knowledge of either, enabling the design of a system with simple deployment and low cost. The SLAM problem will be solved by using the Smoothing and Mapping (SAM) technique.

1.1 Motivation

Depending on the choice of sensor, different ways to use a quadrotor is enabled. External IR cameras are useful to use as a position and attitude reference in a lab, GPS works for coarse navigation outdoors, and lasers for object avoidance. By using new kinds of sensors for tracking, localization and SLAM can open up for new ways of using a quadrotor.

Concerning radio-based systems for localization and SLAM; the systems work indoors, they do not require heavy equipment to be added to the quadrotor, and they are fairly cheap. In particular, the UWB sensor offers high resolution, due to its high bandwidth. Below are a few use-cases which UWB based systems, like the one detailed in this thesis enables.

1.1.1 Use-Cases

Case: Initialize anchor locations

The location ofUWB sensors have to have known in order to infer the quadrotor's position from their measurements. Having a system where the sensor locations are self-initializing enables faster deployment and possibly higher accuracy. In a big warehouse where there is no GPS coverage for localizing quadrotors and an IR camera system would be both expensive and suffer from occlusion, an UWB solution is a viable alternative.

For example, in a warehouse UAVs could take the roles of pickers, i.e. fetch items from a list and bring to a boxing station where the items are packaged to be shipped to a customer. Amazon is one company experimenting with using UAVs to deliver parcels to customers [1].

UAVs could also be tasked to bring deliveries from distribution trucks to the appropriate place in the warehouse. This would solve the problem with having autonomous distribution vehicles and no driver to unload them. An advantage of using UAVs for such tasks is to not occupy the floor area which could be used by both autonomous ground vehicles and people.

Placing and measuring the position of a large set of sensors would be a tedious and hard task, especially if the layout of the location would change often. The solution would be to have the UAVs initialize the sensors while flying. Potentially a subset of the sensors could be initialized manually, allowing the UAVs to have an initial position reference while mapping out the rest of the sensors.

Case: Relay estimated anchor locations from off-board estimator to on-board estimator

The current on-board state estimator needs to be initialized with the measured positions of all involved sensors. These are transmitted to the quadrotor from the host computer at start-up and then remain constant.

The suggested improvement would be to let the system described in this thesis run on the host computer and continuously update the sensor positions which are used in the on-board estimator. This would allow for the positions to be improved throughout the flight without having to run the state estimator off-board, which would induce latency from the wireless channel. The improved positions could also be shared with other quadrotors that use the same set of UWB sensors.

Case: Apply to other devices, such as Microsoft Hololens

UAVs are not the only object that can benefit from being localized indoors. During the last years, several platforms for virtual and augemented reality were launched, such as the Microsoft HoloLens, HTC Vive and Oculus Rift. The Hololens uses a built-in depth camera while both the Vive and Rift rely on external IR-cameras for tracking.

Since UWB sensors do not suffer from occlusion, they could be a good alternative for so-called room scale VR-sessions for the mentioned devices.

When used in ones home, these sensors might not be desirable to have permanently mounted, and placing and measuring their position for each session is tedious. Being able to initialize these by moving the device about in the room would thus make deployment easier and the technology more accessible.

1.2 Problem Formulation

The problem addressed in this thesis is:

> Design, implement and evaluate a system for localizing Crazyflie [2] quadrotors using self-calibrating UWB sensors.

The problem can be broken down into the following three parts:

- Gathering adequate testing and evaluation data, together with ground truth from mocap, e.g. [3]
- Finding a method for creating the necessary factor graph to be used with Georgia Tech's GTSAM framework [4]
- Tuning and evaluating the resulting estimator

1.3 Contributions

In this thesis, the sensor and robot combination of Inertial Measurement Unit (IMU) and UWB senors, and a quadrotor, was investigated together with the GTSAM framework. The pre-integrated IMU Factor was used together with six UWB sensors in order to track a quadrotor.

A system is presented which takes range and IMU inputs and fuse them to estimate the quadrotors position and attitude as well as the sensors' positions in space, without the need for prior assumption of sensor position.

1.4 Prior Art

The SLAM problem was introduced in [5] as the issue of relating a robot's pose (coordinate frames) to features with expected uncertainties was addressed. The first SLAM implementations used an Extended Kalman Filter (EKF) and were due to [6] and [7]. By modeling the SLAM problem as a graph of constraints, [8] is able to formulate a global optimization problem.

The graph optimization problem, in reference to SLAM was shown to be a fast alternative to EKF SLAM in [9], which solves an offline-SLAM problem, which was named Square Root Smoothing and Mapping. In [10], iSAM improves the Square root SAM technique such that it works online in an incremental fashion. In [11], iSAM2 is introduced which offers additional improvements by introducing the Bayes tree data structure for fast inference of new measurements. This is the method used in this thesis.

A recent survey over the current state of SLAM, current challenges and the future of SLAM can be found in [12].

The topic of localization using UWB ranging is treated thoroughly in [13]. Early results from UWB-supported IMU localization of people includes [14] and [15], both of which estimates the position of a person with three degrees of freedom. In [16], the technique is extended to six degrees of freedom.

Early results from quadrotor localization includes [17], where the quadrotor is localized using IMU and GPS. Localization in GPS-denied areas, such that indoors, is solved with a monocular camera and off-board computation in [18]. The image processing is moved on-board in [19], removing the induced latency and limitations of having to transfer images to a base station. These solutions have come a long way, which [20] show with a monocular camera and aggressive flight.

Estimation of the full state of a quadrotor using IMU and UWB is pioneered in [21], where a time-of-arrival approach is used for UWB ranging to be used in an on-board EKF. This work was improved in [22] by eliminating the need for the quadrotor to transmit to the UWB sensors, enabling simultaneous multi-agent localization. Compared to this thesis, [21] and [22] assume that the sensors' locations are prior knowledge and their uncertainty are not modeled in the estimator. In [23], the assumption of prior sensor location knowledge is removed, by proposing a self-initializing system, however, both vision and GPS data are used, as contrary to the system proposed in this thesis.

There are published results of sensor location self-initialization using different techniques, such as ultrasound, UWB and Wi-Fi, one recent paper that uses Wi-Fi round-trip time distance measurements is [24].

Pre-integrating several IMU measurements and combining these into a motion constraint was first proposed in [25], for use with cameras sensors for localizing humans. Improvements on this, by doing the integration on the manifold of the rotation group, was done by [26].

To the best of our knowledge, no results are published on the use of pre-integrated IMU measurements fused with UWB range measurements for a quadro-

tor doing incremental smoothing.

1.5 Outline

A background with the necessary theory is given in Chapter 2. There, all important equations are developed, and the hardware and software is presented. The implementation is then described in Chapter 3, where the specific cases of the previously developed equations are shown together with the system architecture. In Chapter 4, the experimental setup and results are shown, together with a discussion of the results' implications. Finally, conclusions are drawn in Chapter 5 as well as the proposed future work.

Chapter 2

Bakground

2.1 Localization and Mapping

In order to be able to control a robot's position, the position has to be observed or estimated, this will hereinafter be refered to as localization. Localization often make use of a map in order to relate the sensor measurements to known objects or features. This map can either be known in advance or it will have to be constructed during operation. The latter problem is what will be investigated and is often referred to as SLAM. The SLAM problem will in this thesis be solved with the SAM technique.

Throughout Section 2.1.1 to 2.1.3, the theoretical base for the thesis will be explained and relevant equations will be developed. First, an introduction to factor graphs will be given, followed by SLAM and SAM.

2.1.1 Factor Graphs and Probabilistic Modeling

A factor graph can be used as a way of modelling probability and probabilistic relationships. When solving the SLAM problem, factor graphs can be used to describe the optimization problem effectively.

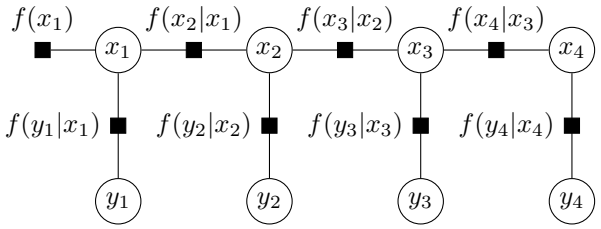


Figure 2.1: Factor graph of a hidden Markov model unrolled over four steps. The hidden states are $x_{1:4}$ and the observations are $y_{1:4}$.

Consider the factor graph in Figure 2.1, which represents a hidden Markov

model unrolled over four steps, with the associated joint probability function

$$f(x_1, \ldots, x_n, y_1, \ldots, y_n) = \prod_{i=1}^{n} f(x_i|x_{i-1}) f(y_i|x_i), \quad n = 4 \tag{2.1}$$

where the hidden states are x_i and the observations y_i [27]. The black squares in between states are called factors.

In order to formulate this as a SLAM problem, assume that $y_{1:n}$ represents landmarks, $\mathbf{L} = \{\mathbf{l}_j\}$, that $x_{1:n}$ represents poses, $\mathbf{X} = \{\mathbf{x}_i\}$, that for each factor between poses, there are control inputs, $\mathbf{U} = \{\mathbf{u}_i\}$, and that for each factor between landmarks and poses, there are measurements, $\mathbf{\Gamma} = \{\boldsymbol{\gamma}_k\}$. This new case is shown in figure 2.2.

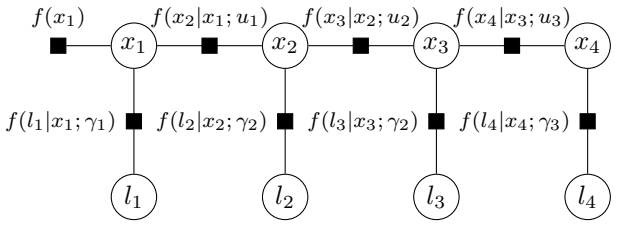


Figure 2.2: Factor graph of a simple SLAM problem where each landmark, l_i is seen once via the measurement γ_i. Neither x nor l are observed directly.

Further assume that the measurement and process noise are both zero-mean Gaussian distributed with covariance matrices $\mathbf{\Sigma}_k$ and $\mathbf{\Lambda}_i$ respectively, such that

$$\begin{aligned} f(x_1) &\propto p(x_1) &&\propto 1 \\ f(x_i|x_{i-1}, u_{i-1}) &\propto p(x_i|x_{i-1}, u_{i-1}) &&\propto \exp\left(-\tfrac{1}{2}\|f_i(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i\|^2_{\mathbf{\Lambda}_i}\right) \\ f(l_i|x_i, \gamma_i) &\propto p(l_i|x_i, \gamma_i) &&\propto \exp\left(-\tfrac{1}{2}\|h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) - \boldsymbol{\gamma}_k\|^2_{\mathbf{\Sigma}_k}\right) \end{aligned} \tag{2.2}$$

where $\propto$ denotes proportionality. Scale factors can be safely omitted since this will later be formulated as an optimization problem.

Generally, when a feature is detected it has to be associated with a landmark that is either already part of the map or it has to be added to the map [28]. In this thesis, however, the association of landmarks is assumed to be solved. This assumption can be made since the jth measurement for every iteration is always caused by the jth sensor and the sensors are the landmarks, see Section 2.3.1 for more details regarding this.

Using the assumptions of the middle column in (2.2) inserted into (2.1) and rewriting it as a maximum-a-posteriori (MAP) optimization problem yields

$$\{\mathbf{X}, \mathbf{L}\}^* = \underset{\{\mathbf{X},\mathbf{L}\}}{\operatorname{argmax}} \prod_{i=1}^{n} p(x_i|x_{i-1}, u_{i-1}) p(l_i|x_i, \gamma_i). \tag{2.3}$$

Taking the negative logarithm, thus transforming the maximization problem into a minimization problem, and inserting the rightmost column of (2.2) yields

$$\{\mathbf{X}, \mathbf{L}\}^* = \underset{\{\mathbf{X},\mathbf{L}\}}{\operatorname{argmin}} \left(\sum_{k=1}^{K} \|h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) - \boldsymbol{\gamma}_k\|^2_{\boldsymbol{\Sigma}_k} + \sum_{i=1}^{M} \|f_i(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i\|^2_{\boldsymbol{\Lambda}_i} \right) \tag{2.4}$$

which in Section 2.1.2 will be shown is the de facto standard SLAM formulation, when stated as an optimization problem [12].

2.1.2 SLAM - Simultaneous Localization and Mapping

In this section, (2.4) will be derived from a SLAM point of view and the equation will be explained and formulated as a least-squares problem, for which there exists efficient solvers.

In the absence of a map and the desire to localize itself, a robot will have to construct a map while localizing. The related problem is called SLAM. It is commonly referred to as a chicken-and-egg problem, since a map is needed to localize and the position is needed to create a map, i.e. create a spatial model that relates physical objects in the robot's environment to each other.

In order to state the SLAM problem mathematically, let $\mathbf{X} = \{\mathbf{x}_i\}$ denote a trajectory, $\mathbf{L} = \{\mathbf{l}_j\}$ denote the set of landmarks, i.e. the map, $\boldsymbol{\Gamma} = \{\boldsymbol{\gamma}_k\}$ the measurements and $\mathbf{U} = \{\mathbf{u}_i\}$ the inputs, such as wheel encoders, acceleration measurements or control input. Stated as a global opimization problem, the standard formulation of SLAM, as a MAP problem given measurements of some sort, as given in [12] is

$$\{\mathbf{X}, \mathbf{L}\}^* = \underset{\{\mathbf{X},\mathbf{L}\}}{\operatorname{argmax}}\ p(\{\mathbf{X}, \mathbf{L}\}|\boldsymbol{\Gamma}) = \underset{\{\mathbf{X},\mathbf{L}\}}{\operatorname{argmax}}\ p(\boldsymbol{\Gamma}|\{\mathbf{X}, \mathbf{L}\})p(\{\mathbf{X}, \mathbf{L}\}) \tag{2.5}$$

where $p(\boldsymbol{\Gamma}|\{\mathbf{X}, \mathbf{L}\})$ is the conditional probability of the measurements given the state or often *measurement probability* or *measurement likelihood*, $p(\{\mathbf{X}, \mathbf{L}\})$ is the *a priori probability* of $\{\mathbf{X}, \mathbf{L}\}$. This part of the SLAM problem is referred to as the back-end in [12]. The front-end refers to the part of the system that relates state to sensor data, i.e. motion and measurement models.

When the entire trajectory is included in the state $\mathbf{X}$, the problem is called *full SLAM*. In the case where only the current pose is estimated, it is referred to as *online SLAM* [28].

On explicit form, (2.5) is written, as shown in [12], as

$$\{\mathbf{X}, \mathbf{L}\}^* = \underset{\{\mathbf{X},\mathbf{L}\}}{\operatorname{argmin}} \sum_{k=1}^{K} \|h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) - \boldsymbol{\gamma}_k\|^2_{\boldsymbol{\Sigma}_k} \tag{2.6}$$

where $h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k})$ is, in this thesis, assumed to be a non-linear function relating the state to the measurements, such that $\boldsymbol{\gamma}_k = h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) + \mathbf{v}_k$ where $\mathbf{v}_k$ is the Gaussian, zero-mean prediction error. The prior is assumed to be known and thus treated as a constant and omitted in the optimization problem.

The function h_k is the observation model or measurement model. Here, the measurement noise is assumed to be zero-mean and distributed as a Gaussian. The norm subscript, Σ_k is the covariance matrix associated with $\mathbf{v}_k$ such that $\|e\|^2_\Sigma = e^T\Sigma^{-1}e$ is the squared Mahalanobis distance [29].

Equation (2.6) does not account for a motion model, i.e. a function which adds a constraint between states, the factors between different x in Figure 2.2.

The function could be linear or, as in this thesis, non-linear. Let this function be $\mathbf{x}_i = f(\mathbf{x}_{j-1}, \mathbf{u}_i) + \mathbf{w}_i$, where $\mathbf{w}_i$ is the Gaussian, zero-mean prediction error, with the associated covariance $\mathbf{\Lambda}_i$ [28]. Adding the motion model to (2.6) yields

$$\{\mathbf{X}, \mathbf{L}\}^* = \underset{\{\mathbf{X},\mathbf{L}\}}{\operatorname{argmin}} \left(\sum_{k=1}^{K} \|h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) - \boldsymbol{\gamma}_k\|^2_{\mathbf{\Sigma}_k} + \sum_{i=1}^{M} \|f_i(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i\|^2_{\mathbf{\Lambda}_i} \right) \tag{2.7}$$

which can then be seen as the optimization problem where we want to fit the states in $\{\mathbf{X}, \mathbf{L}\}$ given the control inputs and measurements, i.e. minimize the errors.

In (2.7), the norm

$$\|f_i(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i\|^2_{\mathbf{\Lambda}_i} \tag{2.8}$$

is called a model factor and

$$\|h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) - \boldsymbol{\gamma}_k\|^2_{\mathbf{\Sigma}_k} \tag{2.9}$$

is a measurement or landmark factor. Recall the concept of factors from Section 2.1.1.

What remains of the problem, on the algorithmic side, is finding adequate models for h_k and f_i, suitable sensors to supply γ_k, and an efficient optimization algorithm for solving the problem. The optimization will be further discussed in Section 2.1.3.

In order to fit the SLAM problem to an efficient non-linear least squares solver, such as the SAM framework [9], (2.4) will have to be linearized by finding the associated Jacobians [12]. SAM will be discussed in more detail in Section 2.1.3.

Consider a non-linear motion model $f_i(\mathbf{x}_{i-1}, \mathbf{u}_i)$, where $\mathbf{u}_i$ consists of the current control inputs and $\mathbf{x}_{i-1}$ is the previous, true pose. This model can be linearized by the Taylor expansion

$$f_i(\mathbf{x}_{i-1}, \mathbf{u}_i) \approx f_i(\hat{\mathbf{x}}_{i-1}, \mathbf{u}_i) + \mathbf{F}_i^{i-1}(\mathbf{x}_{i-1} - \hat{\mathbf{x}}_{i-1}) + \mathbf{G}_{i+1}^{i}\mathbf{w}_i \tag{2.10}$$

where $\hat{\mathbf{x}}_{i-1}$ is the previous pose prediction and $\mathbf{F}_i^{i-1}$ is the Jacobian of f_i evaluated at $\hat{\mathbf{x}}_i$, defined as

$$\mathbf{F}_i^{i-1} = \frac{\partial f_i(\hat{\mathbf{x}}_{i-1}, \mathbf{u}_i)}{\partial \hat{\mathbf{x}}_{i-1}} \tag{2.11}$$

as described in [28] and the Jacobian of f_i, evaluated at $\hat{\mathbf{x}}_i$ w.r.t. the *control inputs*, defined as

$$\mathbf{G}_i^{i-1} = \frac{\partial f_i(\hat{\mathbf{x}}_{i-1}, \mathbf{u}_i)}{\partial \mathbf{u}_i} \tag{2.12}$$

The term $\mathbf{x}_{i-1} - \hat{\mathbf{x}}_{i-1}$ is the prediction error, i.e. the difference between actual and estimated pose.

Linearizing the measurement model in the same way:

$$h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) \approx h_k(\hat{\mathbf{x}}_{i_k}, \hat{\mathbf{l}}_{j_k}) + \mathbf{H}_k^{i_k}(\mathbf{x}_{i_k} - \hat{\mathbf{x}}_{i_k}) + \mathbf{J}_k^{j_k}(\mathbf{l}_{j_k} - \hat{\mathbf{l}}_{j_k}) \tag{2.13}$$

results in the two Jacobians

$$\mathbf{H}_k^{i_k} = \frac{\partial h(\mathbf{x}_{i_k}, \mathbf{l}_{j_k})}{\partial \mathbf{x}_{i_k}} \tag{2.14}$$

and

$$\mathbf{J}_k^{j_k} = \frac{\partial h(\mathbf{x}_{i_k}, \mathbf{l}_{j_k})}{\partial \mathbf{l}_{j_k}}. \tag{2.15}$$

These Jacobians are time-variant and depend on the estimate and linearization point and in turn, the success of the linearization is dependent on these variables as well.

Now define the motion prediction error and the measurement prediction error, which in the filtering realm is often referred to as the innovation [28], as

$$\mathbf{a}_i \triangleq \mathbf{x}_i^0 - f_i(\mathbf{x}_{i-1}^0, \mathbf{u}_i) \tag{2.16}$$

and

$$\mathbf{c}_k \triangleq \boldsymbol{\gamma}_k - h_k(\mathbf{x}_{i_k}^0, \mathbf{l}_{j_k}^0). \tag{2.17}$$

Here, the superscript zero denotes a linearization point. Inserted into (2.4) it can be written as the linear least squares problem

$$\delta^* = \underset{\delta}{\operatorname{argmin}} \left(\sum_{i=1}^{M} \|\mathbf{F}_i^{i-1}\delta\mathbf{x}_{i-1} - \mathbf{I}\delta\mathbf{x}_i - \mathbf{a}_i\|_{\Lambda_i}^2 + \sum_{k=1}^{K} \|\mathbf{H}_k^{i_k}\delta\mathbf{u}_{i_k} + \mathbf{J}_k^{j_k}\delta\mathbf{l}_{j_k} - \mathbf{c}_k\|_{\Sigma_k}^2 \right) \tag{2.18}$$

where $\mathbf{I} \in \mathbb{R}^{d\times d}$ is the identity matrix, $d = \dim(x_i)$ and $\delta \in \mathbb{R}^n$ contains the pose and landmark variables [9].

2.1.3 SAM - Smoothing and Mapping

Before going into the details of SAM, it's valuable to know the difference between smoothing and filtering. A filter, for instance the EKF, will marginalize prior information at each iteration, making it inaccessible for future iterations [30]. The prior information at each iteration usually consists of the previous state, current control inputs, and measurements [28]. With smoothing, however, one is concerned with finding the most probable trajectory and map given the set of available data as a MAP problem, c.f. (2.5) [9].

The disadvantage with the filtering aspect is that any errors admitted to the estimate can't be altered at a later point. This results in an estimator where a bad linearization point can't be corrected later, even if more data exists. The advantages lies in its computational efficiency for some problems [28], where an EKF can be implemented on small embedded processors, such on the Crazyflie [31]. The EKF, when used to solve SLAM greatly reduces in efficiency when applied to problems with large maps, i.e. many landmarks, since the update complexity is quadratic with respect to the number of states and landmarks.

Smoothing on the other hand is advantageous in that during an update step, the entire graph, or a subset of it, can be re-linearized. Another key point with smoothing is that the graph (c.f. section 2.1.1) and especially its associated information matrix can be kept sparse, which means that a set of efficient methods can be used. The marginalization of an EKF-based solution, on the other hand, will cause so-called fill-in, such that the relevant matrices will no longer be sparse [30]. It is worth to point out that, according to [9], the case where every landmark is seen at every time step is the worst case scenario, in terms of fill-in. This is the exact case that is investigated in this thesis.

As mentioned in Section 2.1.2, what's missing from the solution to (2.7) is, apart from motion and measurement models, an efficient algorithm for solving

the optimization problem. Least-squares solvers such as the Gauss-Newton or Levenberg-Marquardt are often used.

In order to write (2.18) as a standard least-squares problem, the sparse matrix $\mathbf{A}$ and the column vector $\boldsymbol{b}$ have to be defined. The shape of these are dependent on the number of poses, landmarks, measurements, and the order of these. For $M = 3$ poses, $N = 2$ landmarks and $K = 4$ measurements, where landmark 1 and 2 are measured at pose 1, landmark 1 measured again at pose 2 and landmark 2 again at pose 3, the resulting $\mathbf{A}$ and $\boldsymbol{b}$ will be

$$\mathbf{A} = \begin{bmatrix} -\mathbf{I}_1 & & & & \\ \mathbf{F}_2^1 & -\mathbf{I}_2 & & & \\ & \mathbf{F}_3^2 & -\mathbf{I}_3 & & \\ \mathbf{H}_1^1 & & & \mathbf{J}_1^1 & \\ \mathbf{H}_2^1 & & & & \mathbf{J}_2^2 \\ & \mathbf{H}_3^2 & & \mathbf{J}_3^1 & \\ & & \mathbf{H}_4^3 & & \mathbf{J}_4^2 \end{bmatrix}, \; \boldsymbol{b} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}. \tag{2.19}$$

The size of $\mathbf{A}$ is $(Nd_x + Kd_\gamma) \times (Nd_x + Md_l)$, where d_x is the number of states, d_γ the dimension of the measurements and d_l the dimension of the landmarks. For the example in (2.19), with the state in three dimensions, range measurements in one dimension and landmarks in three dimensions, the size of $\mathbf{A}$ will be 10×15.

This leads to the least-squares formulation

$$\delta^* = \underset{\delta}{\operatorname{argmin}} \|\mathbf{A}\delta - \boldsymbol{b}\|_2^2. \tag{2.20}$$

For the SAM method to be useful for navigation, inference of measurements will have to be made incrementally and efficiently, as opposed to be done in batch, where all data is processed once. This is addressed by the incremental Smoothing and Mapping (iSAM) [10] algorithm, which was succeeded by iSAM2 [11].

2.2 Quadrotors

The quadrotor is a category of aerial vehicles which use four rotors to create lift and to maneuver, capable of vertical take-off and landing. They are commonly also referred to as quadrocopters, quadcopters, and simple quads. Another common name for a quadrotor is drone, which is a more general term describing Unmanned Aerial Vehicles (UAV), which may or may not be operated autonomously and may have a number of other motor and wing configurations.

Commons sensors to equip quadrotors with are IMU, cameras, and GPS, though some are also carrying pressure sensors and digital compasses. In research, it is common to use external IR cameras for state estimation and evaluation, though impressive results have been presented using only on-board cameras and IMU, for instance in [20].

This section will give an introduction to quadrotors in regards to geometry and the selected platform, the Crazyflie 2.0 in Figure 2.3.

Figure 2.3: The Bitcraze Crazyflie 2.0 quadrotor. Picture courtesy of Bitcraze AB

2.2.1 Formalism and Definitions

Coordinate System and Tait-Bryan angles

In the domains of aerial vehicles it is important to define and formalise the coordinate system in order to write equations unambiguous. While being in the air, the agent's motion has six degrees of freedom with axes usually denounced as the right handed coordinate system (x, y, z) and (ϕ, θ, ψ), also known as *roll*, *pitch*, and *yaw*; these are the X-Y-Z Tait-Bryan angles. In this thesis, the z-axis will be defined as pointing up. *Roll* is defined as the rotation about the x-axis, *pitch* as the rotation about the y-axis, and *yaw* the rotation about the z-axis, see Figure 2.4 for an informal visualization.

In this thesis, an additional representations of orientation will be used, other than the above mentioned Tait-Bryan angles, the rotation matrix.

Rotation Matrices

In order to rotate vectors between different frames of reference, such as the body-frame and inertial frame rotation matrices are used.

The rotation is applied by multiplying the vector to be rotated by the rotation matrix from the left:

$$\boldsymbol{v}_r = \mathbf{R}\boldsymbol{v}. \tag{2.21}$$

Several rotations can be applied at the same time such that

$$\boldsymbol{v}_2 = \mathbf{R}_2\boldsymbol{v}_1 = \mathbf{R}_2\mathbf{R}_1\boldsymbol{v}. \tag{2.22}$$

A useful property of the rotation matrix is that since it's orthogonal, i.e. $\det(\mathbf{R}) = 1$, its inverse is the same as its transpose, i.e.

$$\mathbf{R}^{-1} = \mathbf{R}^T \tag{2.23}$$

or more elegantly put as [32]

$$\mathbf{R}^T\mathbf{R} = \mathbf{I} \tag{2.24}$$

where $\mathbf{I} \in \mathrm{R}^{3\times3}$ denotes the identity matrix. From this follows that the difference in rotation between R_2 and R_1, with respect to some common coordinate frame O will be

$$\mathbf{R}_{21} = \mathbf{R}_{2O}\mathbf{R}_{O1} = \mathbf{R}_{O2}^T\mathbf{R}_{O1} \tag{2.25}$$

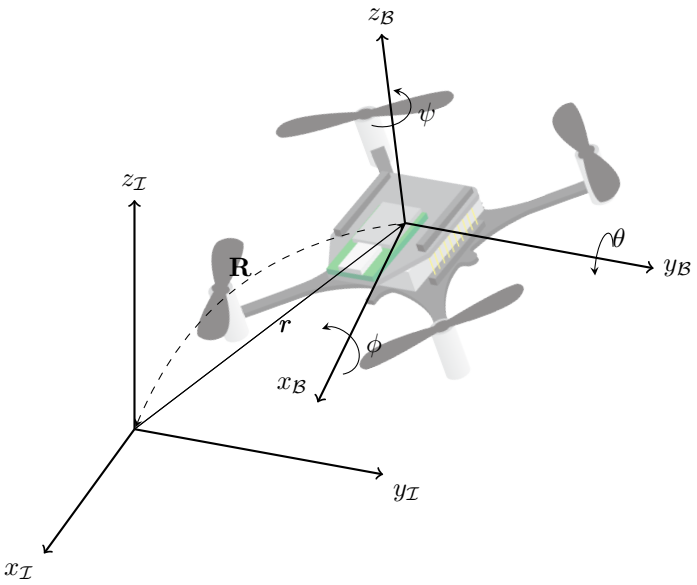


Figure 2.4: The coordinate system conventions. Quadrotor image courtesy of Bitcraze AB

where (2.23) has been used to express the inverse rotation. Here $\mathbf{R}_{O2}$ denotes the rotation from frame 2 to frame O.

The rotation matrix about the x-axis, i.e. the roll angle ϕ, is defined as

$$\mathbf{R}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}, \tag{2.26}$$

the rotation matrix about the y-axis, pitch θ, is defined as

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \tag{2.27}$$

and the rotation about the z-axis, yaw π is the defined as

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.28}$$

From (2.22) follows that we can express the three rotation matrices as one matrix

$$\begin{aligned} \mathbf{R} &= \mathbf{R}(\phi)\mathbf{R}(\theta)\mathbf{R}(\psi) \\ &= \begin{bmatrix} \cos(\psi)\cos(\theta) & -\cos(\theta)\sin(\psi) & \sin(\theta) \\ \cos(\phi)\sin(\psi) + \cos(\psi)\sin(\phi)\sin(\theta) & \cos(\phi)\cos(\psi) - \sin(\phi)\sin(\psi)\sin(\theta) & -\cos(\theta)\sin(\phi) \\ \sin(\phi)\sin(\psi) - \cos(\phi)\cos(\psi)\sin(\theta) & \cos(\psi)\sin(\phi) + \cos(\phi)\sin(\psi)\sin(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix} \end{aligned} \tag{2.29}$$

An important property of the rotation matrix to keep in mind is that it can be singular for certain rotations [33]. For instance, consider a $\pi/2$ rad rotation about the y-axis (pitch). This would yield a rotation matrix with a bottom right element of 0 and thus a singular matrix.

2.3 Sensors

The sensors that are used for state estimation are UWB and IMU. The IMU contains an accelerometer and a rate gyroscope. An introduction to each of the three are given in this section.

2.3.1 UWB - Ultra-wideband Ranging

In the absence of a Global Navigation Satellite System (GNSS) (GPS, GLONASS, etc.), such as when flying indoors, alternative systems have to be used. In academia, high-speed IR-camera-based motion capture systems are frequently utilized, see section 4.1.1, for obtaining an absolute position of a robot, for instance in the Flying Machine Arena at ETH Zürich [34]. Manufacturers include Qualisys and Vicon. These IR-systems are costly and cumbersome, why radio-based localization systems, such as UWB, pose an interesting option. However, whereas the costly motion capture systems can achive sub-cm or even sub-mm accuracy [35], UWB systems are generally less accurate. For instance, the DWM1000 UWB module used in this application has a claimed accuracy of 10 cm [36]. Some advantages of UWB, aside from low-power and cost, include high resolution, as the resolution is inversely proportional to the bandwidth [37], and low susceptibility to multipathing and shadowing [38].

The general idea is to have several UWB sensor nodes, hereinafter referred to as anchors, spread out in an area. The anchors then act as beacons to the quadrotor, which carries an UWB module. A distance measurement from each anchor to the quadrotor is used together with the anchors' positions to estimate the quadrotors's position in an estimator [21].

There are different ways of acquiring a range measurement, hereinafter referred to as ranging, from the sensors. Below, two different, general categories and their characteristics are briefly described.

Two-way ranging

Two-way ranging implies that both the anchor and the robot will transmit messages. Without having to synchronize the devices clocks', one can use the following scheme:

1. The robot transmits message 1, M_1, records the time, t_0.
2. The anchor receives message 1 and records the time of reception, t_1.
3. The anchor transmits message 2, M_2, and records the time of transmission, t_2.
4. The robot receives message 2 and records the time, t_3.
5. The anchor transmits, times t_1 and t_2 to the robot.

The robot now have all the data needed to calculate the distance as [39]

$$d = c\frac{t_3 - t_0 - (t_2 - t_1)}{2} \tag{2.30}$$

where c is the speed of light. Note that even if the clocks do not have to be synchronized, small errors in clock frequency, between the robot's and the anchor's clocks, will result in large errors in range [13].

A modification to the above ranging method was proposed in [21] as such:

1. The robot transmits M_1, records t_0
2. The anchor replies after a predetermined amount of time, δ
3. The robot records the time of reception, t_1
4. The anchor repeats its reply, another δ after the first reply
5. The robot records the time of the reception of the second reply, t_2

The robot can now estimate the anchor's time delay as

$$\hat{\gamma}_\delta = t_2 - t_1. \tag{2.31}$$

Applying a low-pass filter to several estimates $\hat{\gamma}_\delta$ in order to obtain a more stable estimate $\hat{\delta}$ the Time of Flight (TOF) measurement can be calculated as

$$d = c\frac{t_1 - t_0 - \hat{\delta}}{2} \tag{2.32}$$

A disadvantage with two-way ranging techniques is that the robot plays an active part in the communication scheme. This hinders effective multi-robot localization, such as in [40], as the update rate will have to be shared between robots [22]. Dividing the channel between robots could be implemented with Time Division Multiple Access (TDMA), as done by Bitcraze for the Crazyflie [41].

One-way ranging

One-way ranging implies that either the anchor or the robot acts as transmitter. In the case that the clocks of both transmitter and receiver are synchronized and there is no clock skew, the distance can be calculated as

$$d = c(t_b - t_a) \tag{2.33}$$

where t_a is the time of transmission from the anchor and t_b is the time of reception at the robot, this is called Time of Arrival (TOA) or TOF [39].

In the presence of non-ideal clocks, [22] proposes a Time Difference of Arrival (TDOA) ranging scheme where the anchors are synchronized but the robot's clock offset is cancelled out. A TDOA measurement is one where two anchors' signals are being compared, the general formula for a measurement between anchor i and j and the robot is

$$d_{i,j} = \frac{\|\boldsymbol{p} - \boldsymbol{p}_i\| - \|\boldsymbol{p} - \boldsymbol{p}_j\|}{c} + n_i - n_j \tag{2.34}$$

where $\boldsymbol{p}$ is the position of the robot, $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ is the position of anchor i and j, respectively, n_i, n_j is the measurement and channel noise of the two anchors. This way, the robot is not active in the communication and thus a large number of robots can be localized simultaneously without lowering the individual bandwidth.

2.3.2 IMU - Inertial Measurement Unit

The group consisting of accelerometer, gyroscope (e.g., a rate gyro) and magnetometer, sometimes referred to as digital compass, is often bundled in an IMU. These are lightweight, cheap and most quadrotor implementations rely on them [42].

In this thesis, two of these sensors are used, the accelerometer and the rate gyro.

Accelerometer

The accelerometer measures the acceleration of the body frame compared to the inertial frame. This acceleration can be modeled as

$$\boldsymbol{a}_{\text{IMU}} = \frac{1}{m}(\boldsymbol{F}_T - \boldsymbol{F}_g) + \boldsymbol{b}_a + \boldsymbol{\eta}_a \tag{2.35}$$

where $\boldsymbol{a}_{\text{IMU}}$ is the output acceleration, $\boldsymbol{F}_T$ the total external force and $\boldsymbol{F}_g = (0, 0, mg)^T$, the force due to gravitation, $\boldsymbol{b}_a$ models the bias and $\boldsymbol{\eta}_a$ models the measurement noise [43]. Accelerometer measurements tend to be corrupted by significant noise and bias. The bias may be dealt with by taking multiple samples from the sensor when stationary, before taking off, and treat the mean of those samples as the bias, thus assuming it's constant during the entire flight [43]. A more efficient way is to include the bias as a state in the estimator, thus allowing to compensate for time-variant bias [23]. Furthermore, the bias on the ground may be caused by an uneaven surface, which the quadrotor will not operate on, while the bias in the air might be caused by mechanical irregularities, such as an angled motor mount. These sources of bias will not be addressed without adding the bias to the estimator's state space. This is the reason for using the IMU factor, which is detailed in Section 2.4.

When using accelerometers, it is important to remember that when stationary, the accelerometer will measure $\boldsymbol{a}_{\text{IMU}} = (0, 0, \pm g)^T$, depending on how it's mounted. This has to be taken into consideration when modeling and using measurements [44].

Gyroscope

A gyroscope measures the angle with which a body is oriented. In small digital IMUs, the gyroscope is often implemented as a rate gyro, which measures the angular velocity in the body frame compared to the inertial frame. This rotational velocity can be modeled as

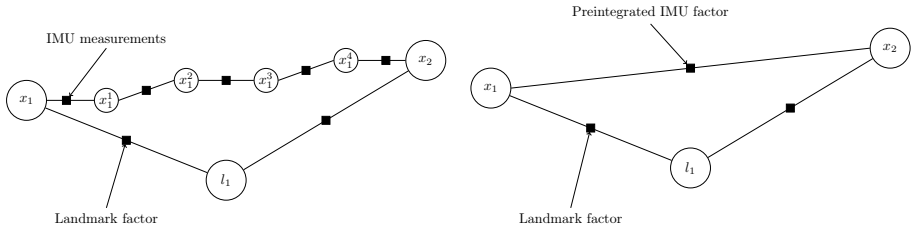
$$\boldsymbol{\Omega}_{\text{IMU}} = \boldsymbol{\Omega} + \boldsymbol{b}_\omega + \boldsymbol{\eta}_\omega \tag{2.36}$$

where $\boldsymbol{\Omega}_{\text{IMU}}$ is the output angular velocity, $\boldsymbol{\Omega} = (\dot{\phi}, \dot{\theta}, \dot{\psi})$ the actual angular velocity of the body in the body-frame, $\boldsymbol{b}_\omega$ the bias and $\boldsymbol{\eta}_\omega$ the measurement noise.

These sensors are generally more robust to noise, compared to accelerometers, however, the bias usually has a considerable drift [43].

2.4 IMU pre-integration

In Section 2.3.2, the accelerometer and gyroscope sensors were described. In this section, an effective way of using them for localization, based on the work in [25] and [26], will be briefly outlined. Please refer to the original works for in-depth discussions and results.



(a) (b)

Figure 2.5: Large circles denote states at iterations where there are landmark measurements. Small circles are inferences between landmark measurements. (a) Measurements are integrated and a state update is done for every IMU measurement. (b) IMU measurements are integrated and form a preintegrated IMU factor between states. States are only updated when there's a landmark measurement.

The accelerometer measures the acceleration of the body with respect to the inertial frame, hence, in order to use it to estimate velocity of translation it has to be integrated once or twice, respectively. The same holds for the gyroscope, in this case a rate gyro, which measures angular velocity and has to be integrated once to estimate attitude.

Both accelerometers and gyroscopes suffer from a time-variant bias. In crude applications, the bias can be estimated at start by averaging a number of measurements while standing still, and removing it from all subsequent measurements. Having the sensor bias as a state in the estimator, however, allows for it to be estimated just as the position or velocity, which bodes for more accurate bias correction. States that are not directly of interest, such as the bias, are called support variables, as opposed to the target variables, such as position and attitude.

In [25], the integration of the measurements is done when the measurement is received, as opposed to when it is used. For example, the IMU may run at well over 500 Hz, while the complementary sensor, such as a camera, GPS, or a radio-based sensor runs at at least an order of magnitude slower. Several integrated IMU measurements are then combined to form a single Preintegrated IMU Factor. In Figure 2.5a, for every IMU measurement a model factor, c.f. (2.8), is added to the factor graph. Only every fifth iteration adds a measurement factor, c.f. (2.9). In Figure 2.5b, however, the IMU measurements are used to construct a preintegrated IMU factor, which will be added to the factor graph, this is the strategy used in this thesis.

The advantage, from the viewpoint of this thesis, is two-fold; by integrating the measurement when they are received, computation time is saved during the more expensive update step, where the optimization takes place. Secondly, the number of factors in the factor graph is reduced, which decreases the computation time during the optimization.

2.5 Platforms

2.5.1 Crazyflie 2.0

The quadrotor used in this thesis' physical implementations is the Crazyflie 2.0 [2], see Figure 2.3, developed by Bitcraze AB [45] in Malmö, Sweden. It features a wide range of sensors, a small price tag, very low weight, and have completely open source and hardware.

The Crazyflie 2.0 includes the following sensor modules:

- 3-axis accelerometer (MPU-9250) [46]
- 3-axis rate gyro (MPU-9250)
- 3-axis magnetometer (MPU-9250)
- Barometer (LPS25H) [47]

An STM32F405 [48] MCU is used for main applications such as a state estimator and controllers (motor, attitude, position) [49].

The radio module used to communicate with the Crazyflie from a host computer is called Crazyflie PA, Figure 2.6 and is a 2.4 GHz radio rated at a 1000 m line-of-sight range when used with a Crazyflie 2.0 [50].

Figure 2.6: The Bitcraze Crazyradio PA. Picture courtesy of Bitcraze AB

Worth noting is that the Crazyflie does not have any available rotor speed information, due to the nature of the brushed motors used. The rotor speed information is popular to use for estimating the thrust [43]. The lack of rotor

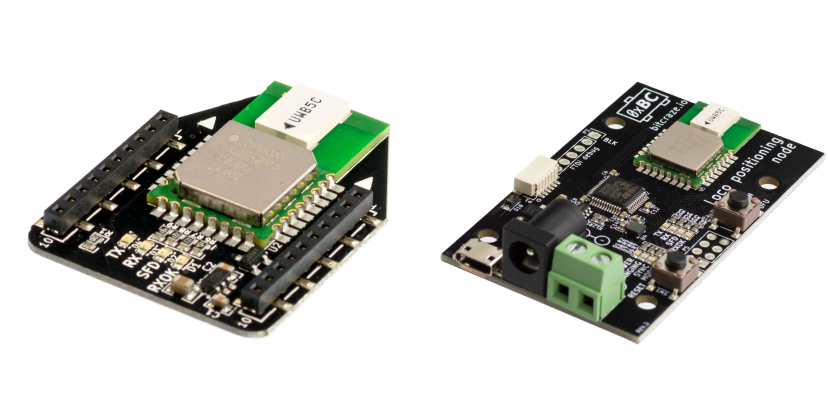
speed information is the main reason for relying heavily on the accelerometer for state estimation.

The Crazyflie quadrotor can be used with the Loco Positioning System (LPS) [51], also from Bitcraze AB. The system is a platform for UWB positioning.

For interfacing the Crazyflie with a PC, Wolfgang Hoenig's Robot Operating System (ROS) drivers [52] are used. They were introduced in [53] and include controllers, teleoperation cababilities and more.

2.5.2 Loco Positioning System

The LPS [51] is an UWB positioning system created by Bitcraze. The hardware consists of the Loco Positioning Deck, Figure 2.7a and the Loco Positioning Node, Figure 2.7b. The nodes acts as anchors and the deck is attached to the Crazyflie and lets the quadrotor communicate with the anchors. Both are equipped with Decawave DWM1000 UWB modules [36]. See Section 2.3.1 for details on Ultra-Wide Band sensors.



(a) Loco Positioning Deck (b) Loco Positioning Node

Figure 2.7: The Loco Position System's hardware. The UWB modules are recognized by their green PCB. Picture courtesy of Bitcraze AB

In this thesis, LPS is used with ROS, using [54] and Wolfgang Hoenig's ROS drivers for the Crazyflie [52], which was introduced in [53].

Further information and a tutorial for the system can be found in [51].

2.6 GTSAM - Georgia Tech Smoothing and Mapping

Georgia Tech Smoothing and Mapping (GTSAM) is a framework for Smoothing and Mapping [4], described in Section 2.1.3. An introduction to GTSAM is given in [55]. It's implemented in C++ with additional interfaces for MATLAB and the Python programming language. The framework, or library, contains tools for building a factor graph, defining factors depending on their properties, such as odometric (between poses) or measurements (between pose and landmarks).

Part of the framework is the preintegrated IMU factor, see Section 2.4 which is the result of the work in [25], [56] and [26].

In this thesis, version 4 of the framework is used.

Chapter 3

Smoothing and Mapping of an Unmanned Aerial Vehicle Using Ultra-wideband Sensors

As stated in Section 1.2, the proposed system will take IMU and UWB measurements and fuse them in order to output an estimate of the quadrotor's trajectory and UWB anchor locations. This is further explained in Section 3.1.1.

In this chapter, the different states and inputs will be detailed, followed by the proposed system architecture. Two models, one for motion and one for measurements will be described together with the outlier rejection.

All noise, throughout this section, will be assumed to be zero-mean and Gaussian, where not otherwise stated.

3.1 Localization and Mapping

3.1.1 Proposed System

Let the inertial frame of reference of the quadrotor be denoted by $\{\mathcal{I}\}$ and the body-centered frame of reference by $\{\mathcal{B}\}$. Both are right-handed systems and oriented as East, North, Up (ENU).

Let $\hat{\mathbf{x}}$ be the output from the motion model such that $\hat{\mathbf{x}}_i = f(\hat{\mathbf{x}}_{i-1}, \mathbf{u}_{i-1})$, where $\mathbf{x}_i$ denotes hte nine-dimensional navigational state (NavState) during the ith iteration such that

$$\mathbf{x}_i = \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{v}_i \\ \boldsymbol{\delta}_i \end{bmatrix} \tag{3.1}$$

where

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \ \boldsymbol{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}, \ \boldsymbol{\delta} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \tag{3.2}$$

recall Figure 2.4 for the coordinate frame convention.

The model inputs, $\tilde{\boldsymbol{u}}$, in this case accelerometer and rate gyro measurements, $\tilde{\boldsymbol{a}}$ and $\tilde{\boldsymbol{\omega}}$ respectively, are defined as

$$\tilde{\boldsymbol{u}} = \begin{bmatrix} \tilde{\boldsymbol{a}} \\ \tilde{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \tilde{a}_x & \tilde{a}_y & \tilde{a}_z & \tilde{\omega}_\phi & \tilde{\omega}_\theta & \tilde{\omega}_\psi \end{bmatrix}^T. \tag{3.3}$$

The ground truth acceleration and angular velocity are denoted as $\boldsymbol{a}$ and $\boldsymbol{\omega}$, respectively.

The accelerometer and rate gyro measurements have the associated, time-variant, bias

$$\boldsymbol{b}_i = \begin{bmatrix} \boldsymbol{b}_i^a \\ \boldsymbol{b}_i^g \end{bmatrix} \tag{3.4}$$

where the superscript a and g denotes accelerometer and gyro respectively.

The landmarks consist of the anchors. Let the state vector for the jth landmark related to the kth measurement be represented as

$$\mathbf{l}_{j_k} = \begin{bmatrix} l_{x_{j_k}} & l_{y_{j_k}} & l_{z_{j_k}} \end{bmatrix}^T \in \mathbf{L}_k, \tag{3.5}$$

the kth UWB measurement, $\boldsymbol{\gamma}$, be represented as

$$\boldsymbol{\gamma}_k = \begin{bmatrix} \gamma_{0_k} & \gamma_{1_k} & \dots & \gamma_{n_k} \end{bmatrix}^T \tag{3.6}$$

with n anchors, and $\hat{\boldsymbol{\gamma}}$ be the predicted measurement, i.e. the output from the measurement model, such that $\hat{\boldsymbol{\gamma}}_k = h(\mathbf{x}_{i_k}, \mathbf{l}_{j_k})$. The measurement model is derived in Section 3.1.4.

Let $\{\mathbf{X}, \mathbf{L}\}^*$, as in (2.6), denote the optimized state and landmark output from the estimator, in this implementation iSAM2 [11].

Figure 3.1 shows how the system is connected. The inputs are the IMU and UWB measurements. The measurements are passed to the IMU factor and UWB factor blocks, which creates the factors, c.f. (2.8) and (2.9), which are inserted in the graph. The graph and all operations done on it are contained in the iSAM2 block. Before a UWB Factor is inserted into the graph, outliers are removed via the Outlier Rejection block. Outlier rejection is discussed in Section 3.1.5. The outputs from iSAM2 are the optimized state and the anchor positions. These optimized variables are fed back to and used to create the next IMU and UWB factor.

3.1.2 Architecture

The proposed architecture, Figure 3.2, is based on what was devolped by Bitcraze as their Loco Positioning System [51], see Section 2.5.2, with the exception that their estimator runs on-board. A two-way ranging method (see Section 2.3.1 for different methods) will be used in order to estimate the distance between a sensor and the quadrotor. The implementation is not dependent on the number of sensors but Bitcraze states the lowest practical number to six in order to offer some redundancy above the theoretical minimum of four anchors, for localization in three dimensions [51].

The communication with the anchors, i.e. obtaining the range measurements, are handled through the quadrotor, which in turn sends the timestamped UWB and IMU measurements to the host computer. In Figure 3.2, it is assumed that the anchors send the range measurements to the quadrotor and it is not dependent on a specific ranging technique.

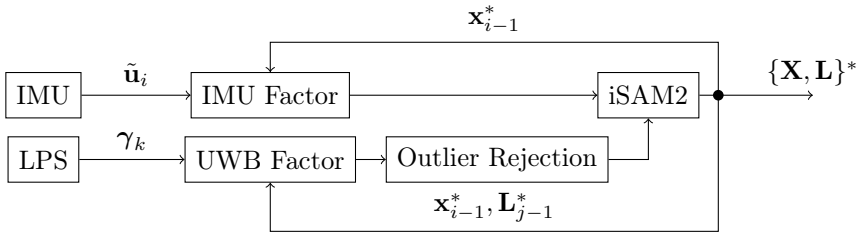


Figure 3.1: Proposed system as explained in Section 3.1.1. The output from the IMU block are the IMU measurements, $\tilde{\boldsymbol{u}}_i$, the output from the UWB block are the UWB range measurements, $\boldsymbol{\gamma}_i$. The blocks labeled IMU Factor and UWB Factor create the factors that are inserted into the graph, which is contained in iSAM2. The outputs from the iSAM2 block are the optimized state and anchor positions.

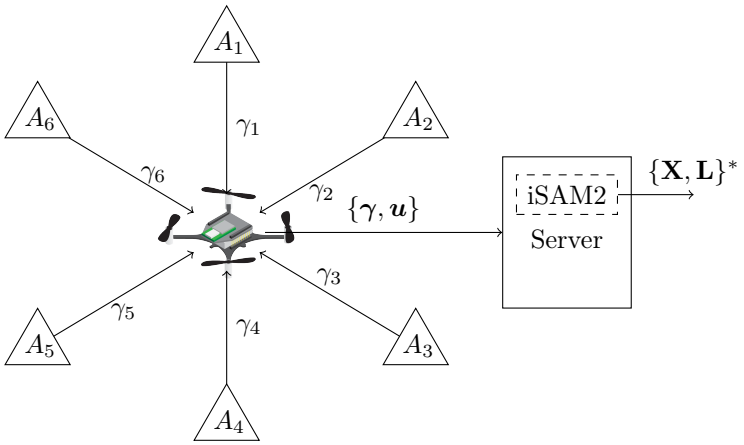


Figure 3.2: Proposed architecture as explained in Section 3.1.2. A_i denotes the ith anchor and γ_i the associated range measurement. The set of range and IMU measurements is denoted $\{\Gamma, u\}$. The proposed estimator runs on the server along with software to communicate with the quadrotor.

3.1.3 Motion Model and the Preintegrated IMU Factor

The motion model, c.f. $f(.)$ in (2.4), used in the proposed system is the model developed in [26], and the pre-integrated IMU factor, see Section 2.4. The IMU factor is considered a black-box module, i.e. only the inputs and outputs are considered.

This model is not designed for a quadrotor in flight, but for a walking person. It was chosen for simplicity. The same holds for the Jacobians, c.f. (2.11) and (2.12). The downside of this is the possibility of worse model performance, than if a quadrotor model would be used. The equations will not be restated here, readers are referred to the original article.

The following discussion is based on [26].

The kinematic equations which relates the IMU measurements to the states,

i.e. the motion model, are

$$\hat{\mathbf{R}}_i = \mathbf{R}^*_{i-1}\text{Exp}(\boldsymbol{\omega}^{\{\mathcal{B}\}}_{i-1}\Delta_i), \tag{3.7a}$$

$$\hat{\boldsymbol{v}}_i = \boldsymbol{v}^*_{i-1} + \boldsymbol{a}^{\{\mathcal{I}\}}_{i-1}\Delta_i, \tag{3.7b}$$

$$\hat{\mathbf{x}}_i = \mathbf{x}^*_{i-1} + \boldsymbol{v}^*_{i-1}\Delta_i + \frac{1}{2}\boldsymbol{a}^{\{\mathcal{I}\}}_{i-1}\Delta_i \tag{3.7c}$$

where $\hat{\mathbf{R}}_i$ is the attitude expressed as a rotation matrix, c.f. (2.29), which transforms a vector from body-centered frame of reference to the inertial frame of reference and Δ_i the time difference between sample i and $i+1$. $\mathbf{R}^*_{i-1}$, $\boldsymbol{v}^*_{i-1}$, and $\mathbf{x}^*_{i-1}$ is the optimized attitude, velocity and position, respectively wheras $\hat{\mathbf{R}}_i$, $\hat{\boldsymbol{v}}_i$, and $\hat{\mathbf{x}}_i$ are the predicted states. Equation (3.7b) calculates the velocity given an accelerometer measurement and the previous velocity and (3.7c) calculates the position, given the previous position, previous velocity and an accelerometer measurement. The Exp-function in (3.7a) is a transformation from the tangent space to the manifold. For further details on Lie algebra och Riemannian geometry, a review is given in [26]. The first order approximation of Exp is given as

$$\text{Exp}(\boldsymbol{\delta}) = \mathbf{I} + [\boldsymbol{\delta}]_\times \tag{3.8}$$

where $\mathbf{I} \in \mathbb{R}^{3\times 3}$ is the identity matrix and $[\hat{\boldsymbol{\delta}}]_\times$ denotes the skew-symmetric matrix of $\hat{\boldsymbol{\delta}}$ such that $\boldsymbol{a}\times\boldsymbol{b} = [\boldsymbol{a}]_\times\boldsymbol{b}$ where $\boldsymbol{a},\boldsymbol{b} \in \mathbb{R}^{3\times 1}$ [57].

The IMU measurements are modeled as

$$\tilde{\boldsymbol{\omega}}^{\{\mathcal{B}\}}_i = \boldsymbol{\omega}^{\{\mathcal{B}\}}_i + \boldsymbol{b}^g_i + \boldsymbol{\eta}^g_i, \tag{3.9a}$$

$$\tilde{\boldsymbol{a}}^{\{\mathcal{B}\}}_i = (\mathbf{R}^*_{i-1})^T\left(\boldsymbol{a}^{\{\mathcal{I}\}}_i - \boldsymbol{g}\right) + \boldsymbol{b}^a_i + \boldsymbol{\eta}^a_i \tag{3.9b}$$

where $\boldsymbol{\eta}_g$ and $\boldsymbol{\eta}_a$ is the additive white noise of the rate gyro and the accelerometer respectively, affecting the measurement, and $\boldsymbol{g}$ is the gravitational constant. In this thesis, however, the accelerometer measurements are given in body-coordinates, where (3.9b) becomes

$$\tilde{\boldsymbol{a}}^{\{\mathcal{B}\}}_i = \boldsymbol{a}^{\{\mathcal{I}\}}_i - (\mathbf{R}^*_{i-1})^T\boldsymbol{g} + \boldsymbol{b}^a_i + \boldsymbol{\eta}^a_i. \tag{3.10}$$

The bias variation is modeled as

$$\dot{\boldsymbol{b}}^g(t) = \boldsymbol{\eta}^{bg}, \quad \dot{\boldsymbol{b}}^a(t) = \boldsymbol{\eta}^{ba} \tag{3.11}$$

where $\boldsymbol{\eta}^{bg}$ and $\boldsymbol{\eta}^{ba}$ is the bias noise on the respective bias. The bias noise can be seen as how much the bias is allowed to change between state updates.

The parameters to the IMU factor that the user can adjust are listed in Table 3.1. The implementation of both the model and IMU bias are found in the GTSAM project's BitBucket repository [4].

Parameter	Description
$\boldsymbol{\eta}^a$	Accelerometer measurement noise
$\boldsymbol{\eta}^g$	Rate gyro measurement noise
$\boldsymbol{\eta}^{ba}$	Accelerometer bias noise
$\boldsymbol{\eta}^{bg}$	Rate gyro bias noise

Table 3.1: Variables used as parameters to the IMU factor

3.1.4 Measurement Model

Model

The measurement model is used to predict what each UWB anchor will measure as a function of the predicted state and the anchor locations.

As described in Section 2.3.1, the UWB sensors outputs a range in meters, which will have to be related to the quadrotor's reference system. The predicted measurement from the jth and the kth measurement is

$$\hat{\gamma}_{k_j} = h(\hat{\mathbf{x}}_{i_k}, \mathbf{l}_{k_j}) = \|\mathbf{l}_{k_j} - \hat{\boldsymbol{x}}_{i_k}\|_2 \tag{3.12}$$

where $\mathbf{l}_{k_j} \in \mathbb{R}^{3\times 1}$ denotes the position of the jth anchor (landmark) at the time of the kth measurement.

Jacobian

Recalling Section 2.1.2, the Jacobian of (3.12) with respect to the navigational state, $\mathbf{x}_i$, $\mathbf{H}_i$, needed for the SAM implementation, c.f. (2.18) is

$$\mathbf{H}_i = \begin{bmatrix} \frac{\partial h}{\partial x_i} & \frac{\partial h}{\partial y_i} & \frac{\partial h}{\partial z_i} \end{bmatrix}. \tag{3.13}$$

Taking the derivative of (3.12) yields

$$\frac{\partial h}{\partial x_i} = \frac{\partial}{\partial x}\left(\sqrt{(l_{j_x} - x)^2 + (l_{j_y} - y)^2 + (l_{j_z} - z)^2}\right) = \frac{x - l_{j_x}}{\|\mathbf{l}_j - \boldsymbol{x}\|_2} \tag{3.14}$$

for the x-component and the jth anchor. For simplicity, the time indices are dropped on the right-hand side since all variables evaluated belongs to to the same timestep. Taking the same derivative for all axes yields the Jacobian as

$$\mathbf{H}_i = \begin{bmatrix} \frac{x - l_{j_x}}{\|\mathbf{l}_j - \boldsymbol{x}\|_2} & \frac{y - l_{j_y}}{\|\mathbf{l}_j - \boldsymbol{x}\|_2} & \frac{z - l_{j_z}}{\|\mathbf{l}_j - \boldsymbol{x}\|_2} \end{bmatrix}. \tag{3.15}$$

Note that $\mathbf{H}_i$ is dependent on the position of the quadrotor and the anchors and will thus change between timesteps and iterations.

Following (2.18) the Jacobian with respect to the landmarks, $\mathbf{l}_{j_i}$, $\mathbf{J}_i^j$ is defined as

$$\mathbf{J}_i^j = \frac{\partial h}{\partial \mathbf{l}_j} \in \mathbb{R}^{1\times 3}. \tag{3.16}$$

where the derivative with regards to the anchor's x-component yields

$$\frac{\partial h}{\partial \mathbf{l}_j} = \frac{\partial}{\partial \mathbf{l}_j}\|\mathbf{l}_j - \boldsymbol{x}\|_2 = \frac{l_{j_x} - x}{\|\mathbf{l}_j - \boldsymbol{x}\|_2} \tag{3.17}$$

for the x-component. Analogously along the other axes yields the Jacobian for the jth anchor and ith iteration as

$$\mathbf{J}_i^j = \left[\frac{l_{j_x} - x}{\|\mathbf{l}_j - \boldsymbol{x}\|_2} \quad \frac{l_{j_y} - y}{\|\mathbf{l}_j - \boldsymbol{x}\|_2} \quad \frac{l_{j_z} - z}{\|\mathbf{l}_j - \boldsymbol{x}\|_2} \right]. \tag{3.18}$$

3.1.5 Outlier Rejection

In order to avoid injecting range measurements which deviate from the expected, an outlier rejection step is required. The outlier rejection is done after the measurement is received but before it is added to iSAM, see Figure 3.1.

In order to detect an outlier, the proposed system will use the Mahalanobis distance. The Mahalanobis distance, D, can be seen as a covariance weighted error [29] and is defined as

$$D = \sqrt{(\hat{\gamma}_i - \gamma_i)\, \mathbf{C}_i\, (\hat{\gamma}_i - \gamma_i)^T} \tag{3.19}$$

where γ_i and $\hat{\gamma}_i$ are the measurements and predicted measurements, respectively. Since the measurements are scalar, 3.19 becomes

$$D = \sqrt{(\hat{\gamma}_i - \gamma_i)^2\, C_i} \tag{3.20}$$

where D is scalar.
In (3.20),

$$C_i = \mathbf{H}_i \mathbf{P}_i \mathbf{H}_i^T + Q \tag{3.21}$$

where $\mathbf{H}$ is the Jacobian from (3.15), $\mathbf{P}$ is the marginal covariance of the state and represents the uncertainty of state $\mathbf{x}_i$, and $Q = q^2$ is the measurement covariance, where q is the UWB measurement noise standard deviation. The marginal covariance is calculated by iSAM2 in GTSAM.

If the Mahalanobis distance is smaller than a certain threshold δ, the associated measurement is considered an inlier, as opposed to an outlier if $D \geq \delta$.

3.2 Parameters and Constants

The parameter and constants, such as noise values and sampling frequencies are shown in table 3.2. All iSAM parameters are left as default, except for the relinerization parameter which decides how often iSAM2 should relinearize. That parameter is set to 1, so that iSAM is forced to relinearize each iteration. The Anchor Noise Parameters depend on the mode of operation. If the anchors are initialized at their ground truth positions, the set of parameters marked "known" is used. If they are initialized as unknown, the other set of parameters are used. See the description of the experiments, Section 4.1 for a description of the two modes of operation.

Parameter (dimensions)	**Value**	**Comment**
Sensor Noise Parameters σ		
Accelerometer measurement noise (x, y, z)	0.1178 m/s^2	20 % of the datasheet value [46]
Rate gyro measurement noise (ϕ, θ, ψ)	1.7500×10^{-4} rad/s	10 % of the datasheet value
Accelerometer bias noise (x, y)	0.1178 m/s^2	20 % of the datasheet value
Accelerometer bias noise (z)	0.1571 m/s^2	"
Rate gyro bias noise (ϕ, θ, ψ)	1.8 rad/s	Hand tuned
UWB measurement noise, q	0.5 m	"
NavState Noise Parameters		
Prior position noise (x, y, z)	0.0001 m	Start position as origin
Prior velocity noise (v_x, v_y, v_z)	0.01 m/s	
Prior rotation noise (diagonal of $\mathbf{R}$)	0.002 rad	
Anchor Noise Parameters		
Prior anchor noise, unknown (x, y)	10 m	When the anchors are initialized as unknown
Prior anchor noise, unknown (x, y)	1.5 m	"
Prior anchor noise, known (x, y, z)	0.1 m	When the anchors are initialized as known
iSAM parameters		
Relinearization skip	1	Forces iSAM to relinearize each iteration
Average Sampling Frequencies		
IMU sampling frequency	91.64 Hz	For one dataset
UWB sampling frequency	31.08 Hz	"
Outlier Rejection		
Mahalanobis distance threshold	1.0	Hand tuned

Table 3.2: Variables used as parameters to the IMU factor. Noise parameters are expressed as their standard deviation, σ

Chapter 4

Experimental Setup, Results, and Discussion

In this chapter the validation of the proposed system is presented. The experimental setup will be detailed, followed by the results together with an interpretation and a discussion of their significance.

The quadrotor flights and related data collection were conducted in the Smart Mobility Lab [58] at KTH.

4.1 Description of Experiments

Two experiments will be performed. In the first experiment the anchors' locations will be unknown. This experiment most resembles a SLAM-case with known correspondences between range measurements and landmarks, anchors, in this case. The anchors' are initialized at 1.5 meters above the xy-plane's origin, as $(0.0, 0.0, 1.5)$, the reason for the z-axis being initalized higher up is to avoid getting a solution that is mirrored in the z-axis.

In the second experiment, the anchors, c.f. (3.5), are initialized at a known position, i.e. the position is manually measured before launch. This most resembles the existing EKF-solutions, e.g. [21] where the anchors are measured accurately. Unlike in the EKF implementations, the positions are modeled with an uncertainty and the anchors' estimated location within the landmark set are allowed to move to better fit the measurements.

A case where a subset of the anchors are unknown was a attempted without satisfying results and will not be presented here.

One dataset was chosen for the experiment on the merits of good flight performance. The robot used the on-board EKF estimate as a position and attitude reference for the control system during the flight. During the flight, the UWB and IMU measurements, and Motion capture system (mocap) position measurements are recorded. The latter used for validation, see Section 4.1.1.

The dataset is 61.2 seconds long and includes a vertical take-off to 1.5 meter, followed by a one meter step in the negative y-direction, a two meter step in the positive y-direction, a one meter step in the positive x-direction and finally a step in the negative y-direction. After each step the quadrotor hovers before

continuing with the next step. The duration of the hover is set to five seconds, but depends on the time it takes for the quadrotor to perform the step.

The position and attitude estimates shown in Section 4.2.1 and 4.3.1 are presented in two ways. The first is what will herinafter be called the instantaneous result. It is instantaneous in the sense that they correspond to the pose that the quadrotor had during the corresponding iteration. Due to the nature of the optimization based localization, the kth estimate can change during the ith iteration, where $i \geq k$, unlike in a filter-based technique where an estimate will not change after it has been calculated. In other words, the instantaneous estimate shown will be the result from the first iteration where it was included, i.e. the 40th estimate will be the value that a robot would see during the 40th iteration. The second way is what will be refered to as the the final result. The final result is the position and attitude that was estimated for each discrete time step as seen in the last time step.

The execution is done on a desktop computer with an Intel Xeon E3-1225 v3 processor with a clock rate of 3.20 GHz.

4.1.1 Ground Truth

Ground truth is obtained from an external mocap with centimeter precision [3]. The mocap system consists of 12 wall-mounted IR-cameras which tracks four spherical reflectors mounted in an asymmetric fashion on the quadrotor.

4.1.2 Anchor Placement

Anchor	**x [m]**	**y [m]**	**z [m]**
1	0.529	-3.55	1.46
2	0.911	-3.44	0.13
3	2.87	-0.317	2.15
4	2.56	-0.0502	0.152
5	0.0320	2.83	1.66
6	0.0755	2.68	0.154

Table 4.1: The anchors' initial position for the first dataset

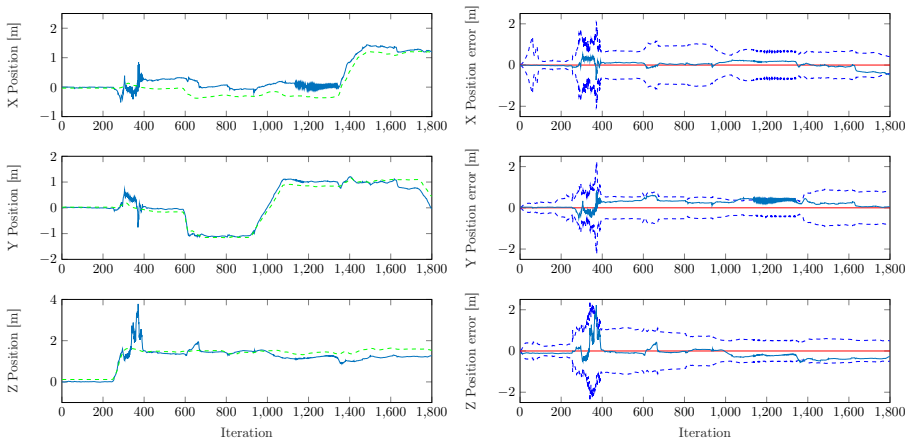
Six anchors were used with three of them placed close to the floor and the other three on tripods as high as possible. The anchors' measured position is shown in Table 4.1. The anchors were measured using mocap, where the anchors are positioned vertically, the side with the UWB module pointing up, recall Figure 2.7b. The measured position corresponds to the top of the anchor, where the mocap markers were placed.

The quadrotor starts at the origin, pointing in the positive x-direction, with the positive y-direction to the left and z is pointing upwards, recall Figure 2.4 for the coordinate system convention.

4.2 Experiment 1: Unknown Starting Position of Anchors

4.2.1 Results

Position and Attitude

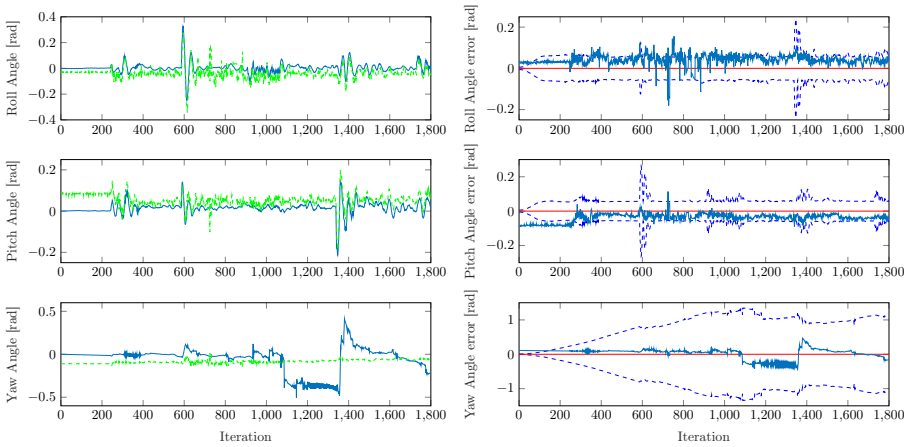


(a) Position (b) Error in Position

Figure 4.1: Instantaneous position results for the case with unknown anchor starting positions. (a) The estimated position, in solid blue, compared to the position measured with the mocap, in dashed green. (b) The error in estimated position, in solid blue, compared to the position measured with the mocap. The 3σ line is dashed blue. The solid red line is at zero, for reference.

The estimated position and its error, as the difference between the estimate and the mocap measurement, are shown in Figure 4.1. In the error plot, the error is shown together with what will be referred to as the 3σ-line or region. The standard deviation $\sigma = \sqrt{\text{diag}\,(\mathbf{P})}$ with $\mathbf{P}$ being the marginal covariance matrix, associated with the estimated state, recall Section 3.1.5. This is a rule of thumb which is used to evaluate if the noise model is adequate and how the noise is propagated through the system. If the estimate routinely exceeds the 3σ bounds or is much larger, the system is said to be overconfident. The covariance in Figure 4.1 accounts for the rather large error in the region between iterations 200 and 400. The error stays within the bounds for the most part, with an exception in the z-axis just before the 400^{th} iteration. Thus, the model is adequate. The error settles after the 400^{th} iteration. Seeing that the quadrotor takes off at iteration 240, and the error settles at around iteration 395, the system settles after 155 iterations, which equals around five seconds. The estimate, in particular the y-axis, has a positive bias, which the noise model does not account for.

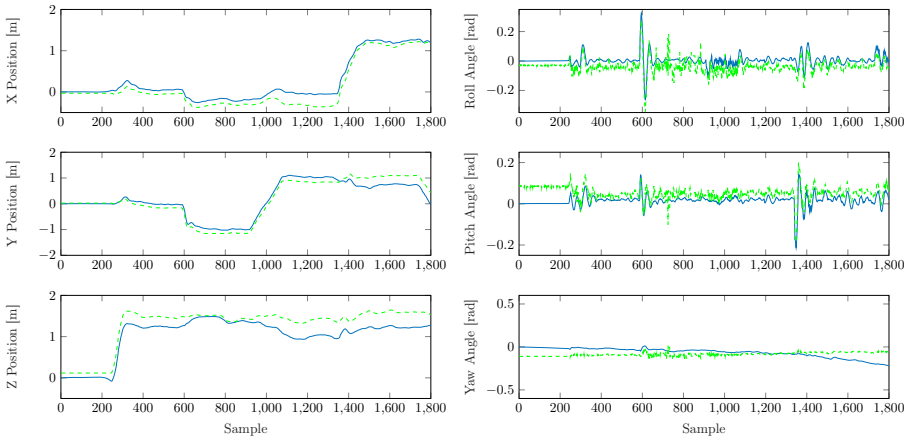
The attitude estimate and its error, as the difference between the estimate and the mocap measurement, are shown in Figure 4.2. Most notably, there is considerable bias in both roll and pitch angles. The error, however, is for the most part contained in the 3σ-area. The error in the yaw-angle is large. This



(a) Attitude (b) Error in Attitude

Figure 4.2: Instantaneous attitude results for the case with unknown anchor starting positions, ordered as roll, pitch, and yaw from top to bottom. (a) The estimated attitude, in solid blue, compared to the attitude measured with the mocap, in dashed green. (b) The error in estimated attitude, in solid blue, compared to the attitude measured with the mocap. The 3σ line is dashed blue. The solid red line is at zero, for reference.

is presumably due to the quadrotor keeping the same heading throughout the entire flight. Furthermore, the uncertainty doesn't grow unbounded, as shown by the 3σ-lines.



(a) Final Position (b) Final Attitude

Figure 4.3: Final position and attitude results with unknown anchor positions. (a) The estimated position, in solid blue, compared to the position measured with the mocap, in dashed green. (b) The final attitude estimate, in solid blue, compared to the position measured with the mocap, dashed green. The 3σ line is dashed blue.

In Figure 4.3, the *final* estimates are shown for both the position and attitude. Here, final refers to the estimate in each *sample* as estimated at the end of the sequence. This is contrary to the instantaneous estimates shown in Figure 4.1 and 4.2, where the result was shown per iteration. In this case, the estimate does not only depend on past estimates, but also incorporates future estimates and measurements. Most notably, the fast dynamics in the position estimate in the region from sample 300 to 400 is smoothed out completely.

Examining the attitude, the roll and pitch angles remain similar, but the yaw angle's erratic behavior is also smoothed out. Still, there's a drift in the estimate unlike that which is observed in the other angles and in the position.

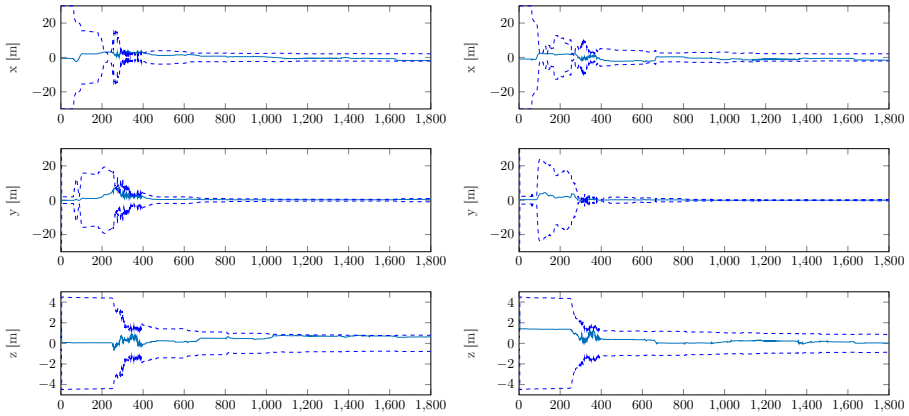
Anchor Position

The errors in the estimated anchor positions are shown in figure 4.4. The final position, error and error norm, denoted $\mathbf{r}$, is presented in Table 4.2. The error is formed by comparing the estimate with the measured anchor locations, Table 4.1. For the most part, the estimates stay within the 3σ limits. Notable exceptions are Anchor 5, where the x-axis estimate is out of bounds between iteration 400 and 1000 and Anchor 6's y-axis which exhibits high confidence but still has an error of close to 40 cm. Other than that, the noise model is adequate. Considering the errors, most apparent in Table 4.2, they are for Anchors 1, 2, and 6 over 1 meter. Studying Anchor 1's x-axis, the error takes a turn for the worse at around iteration 1630. The same holds for Anchor 4 at iteration 1400. Thus, it is hard to say if the anchor estimates would grow worse or better, with more flight time. Furthermore, note that Anchor 4 and 6 has a negative z-estimate. The Anchors can not be placed in such a way, but this limitation is not reflected in the implementation since such a hard limitation could cause the underlying optimization algorithm to exhibit unpredictable behavior. This is a question of the model not being adequate for the anchors.

It is important to remember that the anchor estimates are the effect of trying to fit the anchors and the trajectory to the measurements, and that the measurements are dependent on the anchor positions. Because of this, it can't be said for certain that the anchor positions are bad, since they support a fairly good trajectory, as seen in Figure 4.3. However, their reusability could be a metric of their quality, i.e. if a second quadrotor could reuse the same anchor estimates and complete a successful flight.

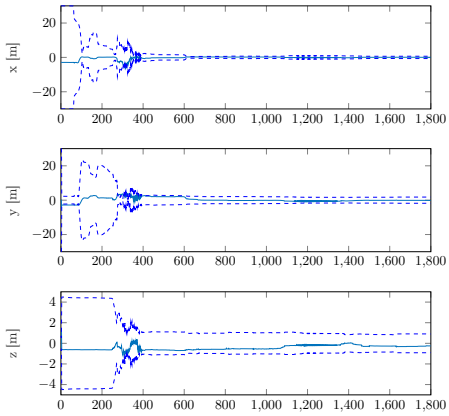
	Position			**Error**			
Anchor	**x [m]**	**y [m]**	**z [m]**	**x [m]**	**y [m]**	**z [m]**	**r [m]**
1	-1.34	-3.09	2.09	-1.87	0.462	0.626	2.03
2	-0.689	-3.42	0.157	-1.60	0.0262	0.0244	1.60
3	2.90	-0.468	1.90	0.0361	-0.151	-0.247	0.292
4	2.47	-0.464	-0.211	-0.0809	-0.414	-0.363	0.557
5	0.0728	2.84	1.32	0.105	0.0130	-0.347	0.363
6	1.00	2.31	-0.409	1.08	-0.377	-0.563	1.28

Table 4.2: The anchors' final position and errors for the case with unknown anchor positions. The rightmost column is the norm of the error.

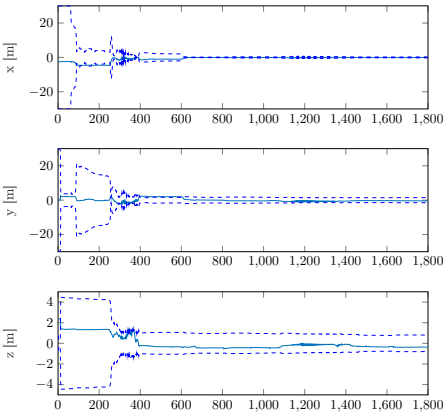


(a) Anchor 1

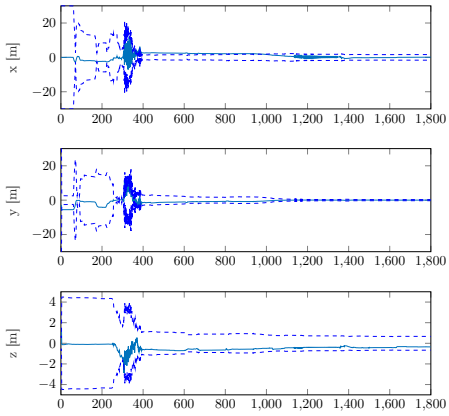
(b) Anchor 2



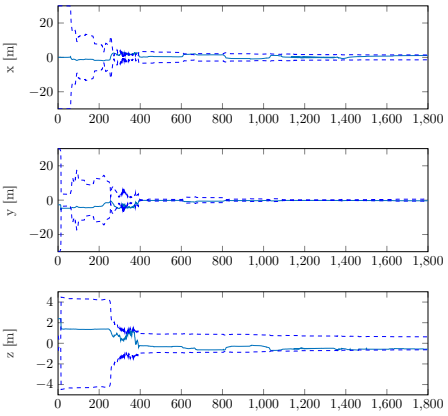
(c) Anchor 3



(d) Anchor 4



(e) Anchor 5



(f) Anchor 6

Figure 4.4: The estimate of the anchor positions for the case with unknown anchor position. The estimate is in solid blue. The dashed blue lines show the 3σ region.

Execution Time

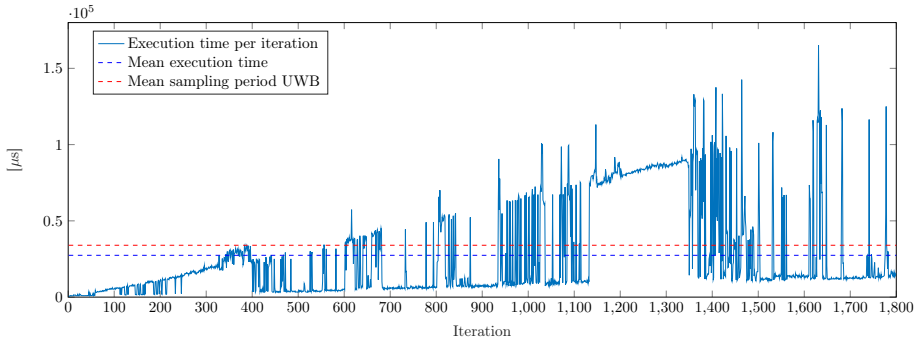


Figure 4.5: Execution time for each iteration for the case with unknown anchor starting positions, shown in solid blue together with the mean in dashed blue and the mean sampling period of the UWB measurements in dashed red.

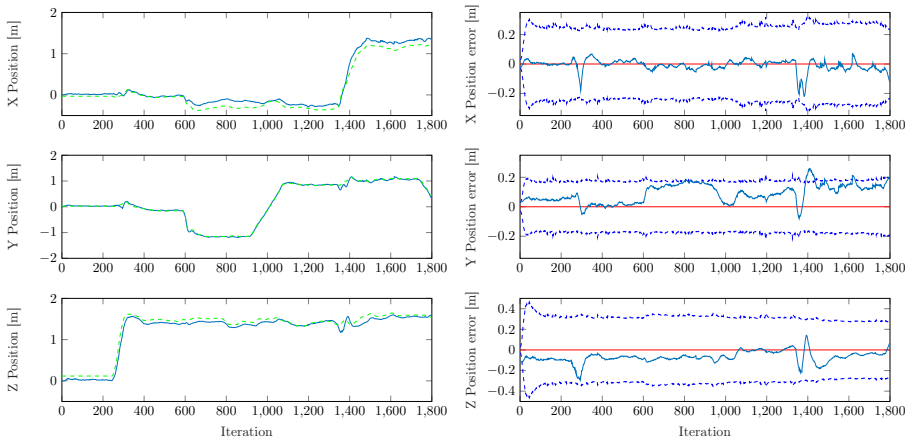
The execution time, in micro seconds, for each iteration is shown in Figure 4.5, together with the mean time between each UWB measurement, the latter seen as the time limit per iteration. Although the mean is below the limit, individual iterations starts to regularly exceed the limit after iteration 602, which occurs after 16.5 seconds of execution time or 20.6 seconds of flight time. The entire 61.2 second flight amounts to 49.3 seconds of CPU time. Most noticeably is that the peak time increase with the number of iterations, which presumably has to do with the graph containing more entries. Furthermore, it was observed while testing that large errors give rise to longer execution times. The long execution times observed in the region between iteration 1150 and 1350 could be traced to the large errors in yaw angle, see Figure 4.2, and the oscillatory behaviour in the x-position estimate, see Figur 4.1. The base execution time is similar throughout the sequence. Recall that iSAM2 is configured to allow relinearization at each iteration, which affects the execution time negatively. No effort has been made to streamline the code, and the execution involves reading from the hard drive.

4.3 Experiment 2: Known Starting Position of Anchors

4.3.1 Results

Position and Attitude

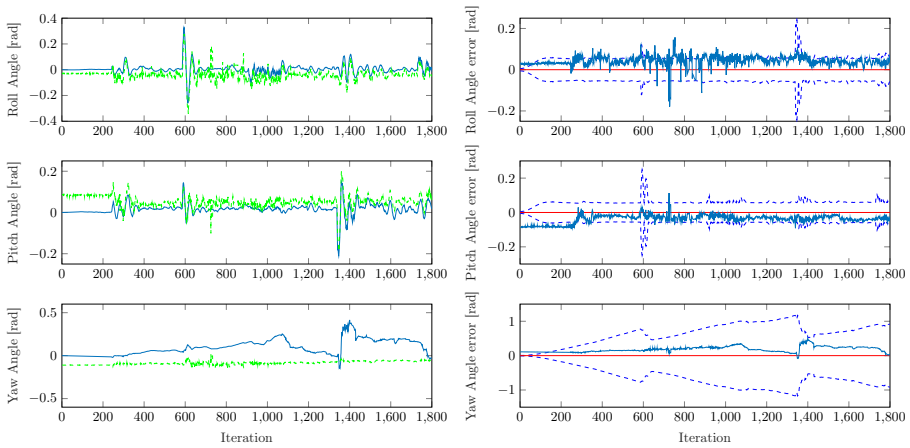
The estimated position and the position error, as the difference between the estimate and the mocap measurement, are shown in Figure 4.6. The covariance accounts for the error at close to all iterations, except for the y-position at around iteration 1400. The model is deemed adequate. The error is consistantly small, with the biggest errors occuring during the take-off and at around iteration 1400. This is the expected behavior.



(a) Position (b) Error in Position

Figure 4.6: Instantaneous position results for the case with known anchor starting positions. (a) The estimated position, in solid blue, compared to the position measured with the mocap, in dashed green. (b) The error in estimated position, in solid blue, compared to the position measured with the mocap. The 3σ line is dashed blue. The solid red line is at zero, for reference.

There is a small positive and negative bias in the y and z-position, respectively.

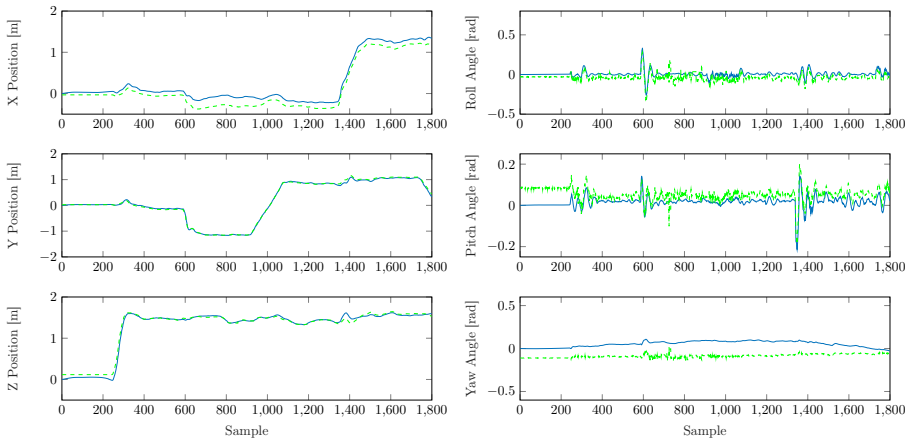


(a) Attitude (b) Error in Attitude

Figure 4.7: Instantaneous attitude results for the case with known anchor starting positions, ordered as roll, pitch, and yaw from top to bottom. (a) The estimated attitude, in solid blue, compared to the attitude measured with the mocap, in dashed green. (b) The error in estimated attitude, in solid blue, compared to the attitude measured with the mocap. The 3σ line is dashed blue. The solid red line is at zero, for reference.

The attitude estimate and the estimated attitude's error, as the difference

between the estimate and the mocap measurement, are shown in Figure 4.7. Likewise with the unknown case, shown in Figure 4.2, there is considerable bias in both roll and pitch angles. The error, however, is for the most part contained in the 3σ-area. The estimate in roll and pitch angle is very similar to the unknown case. This is expected since the known starting position of the anchors should make the range measurements better, but not affect the attitude measurements since these come from the IMU.



(a) Final Position (b) Error in Final Position

Figure 4.8: Final position and attitude results with known anchor positions. (a) The estimated position, in solid blue, compared to the position measured with the mocap, in dashed green. (b) The final attitude estimate, in solid blue, compared to the position measured with the mocap, dashed green. The 3σ line is dashed blue.

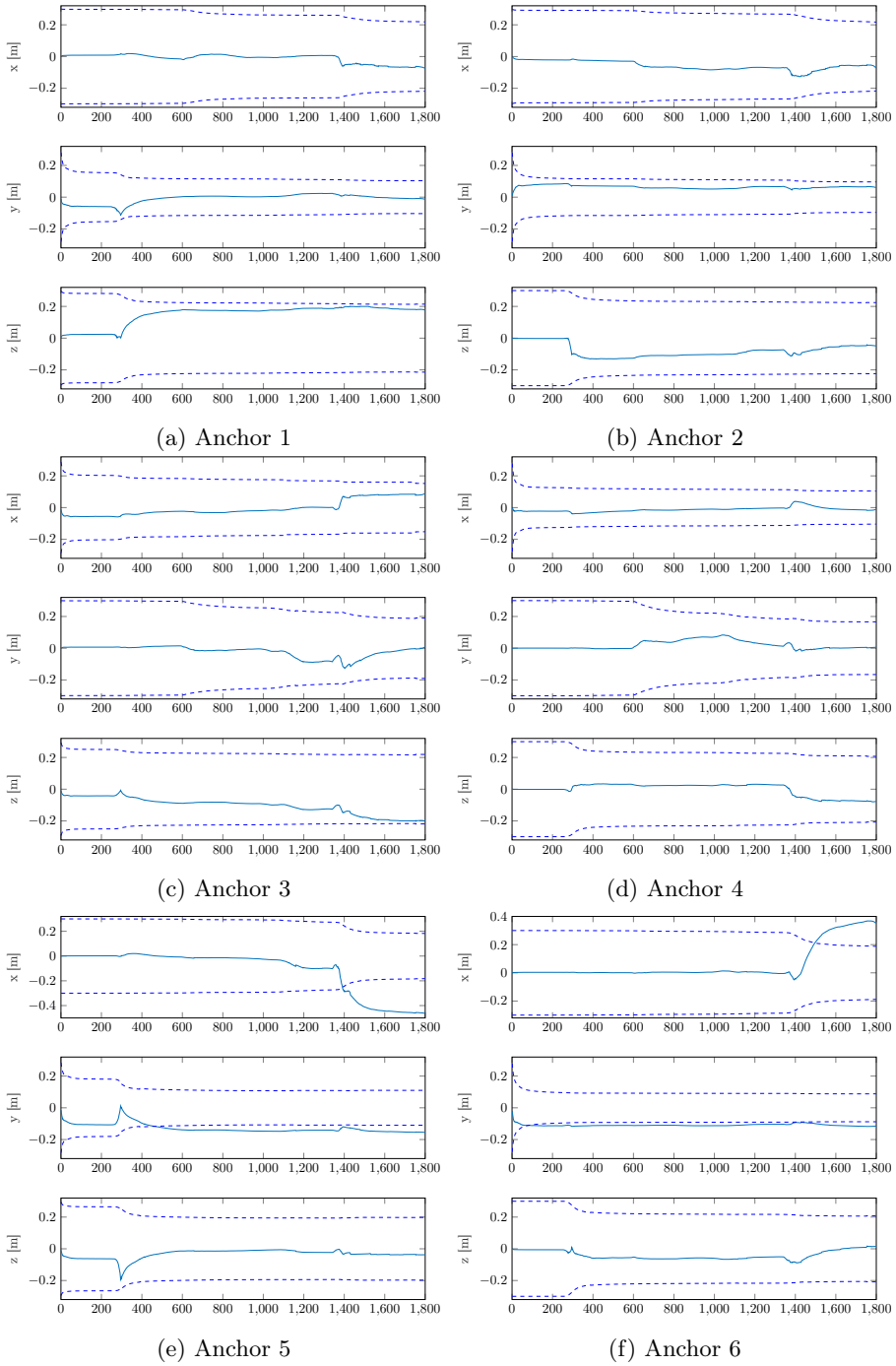
In Figure 4.8, the final estimates are shown for both the position and attitude. As expected, there is little difference between the instantaneous, Figure 4.6, and the final results, since the estimated anchor positions do not change much during the sequence. However, as in the case with unknown anchor positions, Figure 4.8, the faster dynamics have been supressed, most notably around sample 1400.

The behavior in attitude is again similar to the unknown case. There is little difference in roll and pitch while the yaw angle estimate has been smoothed out. The final yaw estimate has a similar shape as in the unknown case, but in the known case it more closely follows the true yaw angle.

What is distressing about the final estimates, is that in all three dimensions, the estimated position is slightly farther from the ground truth than in the instantaneous result. This is likely due to the anchor positions worsening throughout the execution, as will be shown next, Figure 4.9.

Anchor Position

The anchor estimate errors are shown in Figure 4.9. The final position, error and error norm, denoted $\mathbf{r}$, is presented in Table 4.3. The anchor positions, in this case, are initialized at the ground truth which means that they ideally



(a) Anchor 1 (b) Anchor 2

(c) Anchor 3 (d) Anchor 4

(e) Anchor 5 (f) Anchor 6

Figure 4.9: The anchors' final position and errors for the case with unknown anchor positions. The rightmost column is the norm of the error.

should not move at all, or at least end up in the same point. This is not the case, as can be seen in the figure. In the unknown case, the argument was made that

the anchors positions, although the errors were large in some cases, improved the trajectory. Judging by the final estimates, shown in Figure 4.8, which are worse than in the instantaneous, Figure 4.6, the anchor positions can only be said to be worse than the initial.

This test shows that the estimate has much to gain by improving the anchor model, which clearly does not adequately describe the anchors' behavior. This is, however, somewhat expected since the measurement model shown in Section 3.1.4 is linear and simple.

For the most cases, the noise model accounts for the error. Exceptions are the x and y-axis of anchor 5 and 6.

	Position			**Error**			
Anchor	**x [m]**	**y [m]**	**z [m]**	**x [m]**	**y [m]**	**z [m]**	**r [m]**
1	0.455	-3.57	1.64	-0.0747	-0.0135	0.176	0.192
2	0.838	-3.38	0.0841	-0.0733	0.0601	-0.0483	0.106
3	2.96	-0.307	1.95	0.0897	0.0095	-0.201	0.220
4	2.54	-0.0438	0.0755	-0.0108	0.0064	-0.0764	0.0774
5	-0.495	2.67	1.63	-0.463	-0.155	-0.0383	0.490
6	0.275	2.57	0.170	0.351	-0.115	0.0163	0.369

Table 4.3: The anchors' final position and errors for the known case. The rightmost column is the 2-norm of the error, i.e. the distance from the true position.

Execution Time

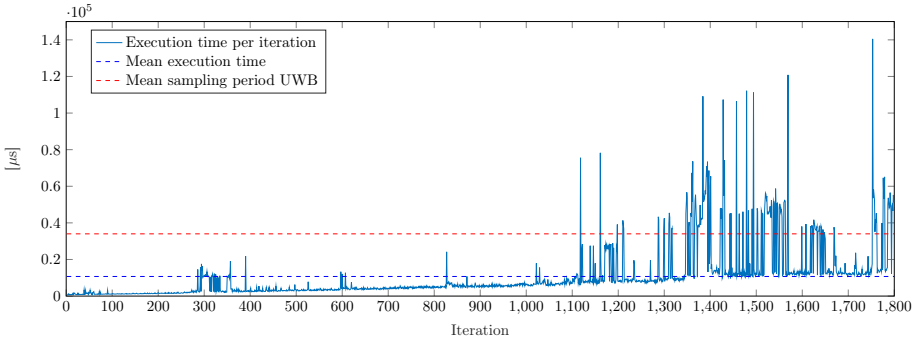


Figure 4.10: Execution time for each iteration for the case with known anchor starting positions, shown in solid blue together with the mean in dashed blue and the mean sampling period of the UWB measurements in dashed red.

The execution time, in micro seconds, for each iteration is shown in Figure 4.10. Likewise with the unknown case, Figure 4.5 the mean is below the limit, with individual iterations exceeding the limit. Individual iterations starts to regularly exceed the limit after iteration 1118, almost twice as long into the sequence as in the known case. The entire 61.2 second flight amounts to 19.3

seconds of CPU time. The shorter average execution time, compared to the unknown case is likely due to the fact that the relative errors are smaller in the known case. Likewise with the unknown case, the execution time seems to increase with the number of factors in the graph.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Comprehensive background of the area of SLAM and SAM along with a theoretical foundation has been given, from which an introduction to the associated problem was presented.

A system that implements UWB localization for a UAV was presented. The system fused IMU measurements with UWB ranges, using state of the art IMU integration and smoothing techniques. Results were shown where the system was evaluated with real flight data offline, with both unknown and known prior anchor positions.

The system estimated the quadrotor's position and attitude with acceptable errors. The errors in the estimated anchor positions were too large to be acceptable.

To conclude, the measurement model does not adequately describe the measurements and has to be improved for the system to be adopted for the previously discussed use cases.

5.2 Future Work

5.2.1 Purpose-built Models

Both the measurement model and the motion model used in this thesis leave room for improvements. The measurement model is linear and does not take nonlinearities such as multipathing or bias into consideration.

The motion model is not specifically designed for a quadrotor, by adapting the IMU factors to use a quadrotor model the overall system performance should improve. This requires, amongst other things, derivation of Lie-algebraic functions and the relevant Jacobians. A quadrotor model was initially implemented for this thesis, but the attempt was abandoned in favor of the IMU factor.

5.2.2 Multiple Robots

The only part of the current system that obstruct a multi-robot operation is the two-way UWB ranging algorithm, as discussed in Section 2.3.1. By implementing a one-way UWB ranging algorithm, it is straightforward to adapt the system for multiple robots. Multi-robot operation opens up for a wide variety of use cases, where only a single robot would not be enough, such as in a warehouse.

5.2.3 On-board Exteroceptive Sensing

The current system does not offer any way of doing collision avoidance, since the agent does not have any sensors able to measure the distance to objects from the quadrotors frame of reference, so called exteroceptive sensors. Adding this would allow for flights in cluttered environments. The system can be extended with more sensors by only specifying a measurement model and noise model. For a tiny quadrotor such as the crazyflie, small laser sensor modules are available.

Bibliography

[1] "Amazon prime air," https://www.amazon.com/Amazon-Prime-Air/b?node=8037720011, [Online; accessed 2017-06-25].

[2] "Crazyflie 2.0," https://www.bitcraze.io/crazyflie-2/, [Online; accessed 2017-06-25].

[3] "Qualisys motion capture," http://www.qualisys.com/, [Online; accessed 2017-06-25].

[4] F. Dellart *et al.*, "GTSAM," https://bitbucket.org/gtborg/gtsam/, 2017, [Online; accessed 2016-12-30].

[5] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The international journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[6] P. Moutarlier and R. Chatila, "An experimental system for incremental environment modelling by an autonomous mobile robot," in *Experimental Robotics I.* Springer, 1990, pp. 327–346.

[7] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 376–382, 1991.

[8] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous robots*, vol. 4, no. 4, pp. 333–349, 1997.

[9] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[10] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[11] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the bayes tree," *The International Journal of Robotics Research*, p. 0278364911430419, 2011.

[12] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age," *arXiv preprint arXiv:1606.05830*, 2016.

[13] S. Gezici, Z. Tian, G. B. Giannakis, H. Kobayashi, A. F. Molisch, H. V. Poor, and Z. Sahinoglu, "Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 70–84, July 2005.

[14] S. Sczyslo, J. Schroeder, S. Galler, and T. Kaiser, "Hybrid localization using UWB and inertial sensors," in *2008 IEEE International Conference on Ultra-Wideband*, vol. 3, Sept 2008, pp. 89–92.

[15] S. Pittet, V. Renaudin, B. Merminod, and M. Kasser, "UWB and MEMS based indoor navigation," *Journal of Navigation*, vol. 61, no. 3, p. 369–384, 2008.

[16] J. D. Hol, F. Dijkstra, H. Luinge, and T. B. Schon, "Tightly coupled UWB/IMU pose estimation," in *2009 IEEE International Conference on Ultra-Wideband*, Sept 2009, pp. 688–692.

[17] G. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. J. Tomlin, "The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC)," in *The 23rd Digital Avionics Systems Conference (IEEE Cat. No.04CH37576)*, vol. 2, Oct 2004, pp. 12.E.4–121–10 Vol.2.

[18] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based mav navigation in unknown and unstructured environments," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 21–28.

[19] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, "Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 3056–3063.

[20] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, April 2017.

[21] M. W. Mueller, M. Hamer, and R. D'Andrea, "Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1730–1736.

[22] A. Ledergerber, M. Hamer, and R. D'Andrea, "A robot self-localization system using one-way ultra-wideband communication," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 3131–3137.

[23] K. Hausman, S. Weiss, R. Brockers, L. Matthies, and G. S. Sukhatme, "Self-calibrating multi-sensor fusion with probabilistic measurement validation for seamless sensor switching on a UAV," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 4289–4296.

[24] K. Batstone, M. Oskarsson *et al.*, "Robust time-of-arrival self calibration and indoor localization using Wi-Fi round-trip time measurements," in *Communications Workshops (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 26–31.

[25] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 61–76, Feb 2012.

[26] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, Feb 2017.

[27] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.

[28] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[29] R. D. Maesschalck, D. Jouan-Rimbaud, and D. Massart, "The Mahalanobis distance," *Chemometrics and Intelligent Laboratory Systems*, vol. 50, no. 1, pp. 1 – 18, 2000.

[30] H. Strasdat, J. M. Montiel, and A. J. Davison, "Visual slam: why filter?" *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[31] T. Antonsson *et al.*, "Crazyflie 1.0/2.0 firmware," https://github.com/bitcraze/crazyflie-firmware, 2016, [Online; accessed 2016-12-19].

[32] M. D. Shuster, "A survey of attitude representations," *Navigation*, vol. 8, no. 9, pp. 439–517, 1993.

[33] P. Singla, D. Mortari, and J. L. Junkins, "How to avoid singularity when using euler angles," *Advances In The Astronautical Sciences*, vol. 119, no. II, pp. 1409–1426, 2005.

[34] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The flying machine arena," *Mechatronics*, vol. 24, no. 1, pp. 41–54, 2014.

[35] M. Field, D. Stirling, F. Naghdy, and Z. Pan, "Motion capture in robotics review," in *2009 IEEE International Conference on Control and Automation*. IEEE, 2009, pp. 1697–1702.

[36] "Datasheet: DWM1000 IEEE 802.15.4-2011 UWB Transceiver Module," Decawave, 2016.

[37] H. Soganci, S. Gezici, and H. V. Poor, "Accurate positioning in ultra-wideband systems," *IEEE Wireless Communications*, vol. 18, no. 2, pp. 19–27, April 2011.

[38] J.-Y. Lee, "Ultra-wideband ranging in dense multipath environments," Ph.D. dissertation, University of Southern California, 2002.

[39] R. Dalce, T. Val, and A. Bossche, "Comparison of indoor localization systems based on wireless communications," *Wireless Engineering and Technology*, vol. 2, no. 4, pp. 240–256, 2011.

[40] J. A. Preiss, W. Hönig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *IEEE Int. Conf. on Robotics and Automation (ICRA), Accepted, To Appear*, 2017.

[41] Arnaud Taffanel - Bitcraze Blog, "Flying many Crazyflie with Loco Positioning System," https://www.bitcraze.io/2016/09/flying-many-crazyflie-with-loco-positioning-system/, September 2016, [Online; accessed 2016-12-30].

[42] P. Martin and E. Salaun, "The true role of accelerometer feedback in quadrotor control," 2010.

[43] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[44] R. C. Leishman, J. C. Macdonald, R. W. Beard, and T. W. McLain, "Quadrotors and accelerometers: State estimation with an improved dynamic model," *IEEE Control Systems*, vol. 34, no. 1, pp. 28–41, Feb 2014.

[45] "Bitcraze AB," https://www.bitcraze.io/, [Online; accessed 2017-05-28].

[46] *MPU-9250 IMU Datasheet*, InvenSense, June 2016, rev. 1.1.

[47] *LPS25H Pressure Sensor Datahseet*, ST, April 2016, rev. 5.

[48] *STM32F405xx ARM Cortex-M4 MCU Datasheet*, ST, September 2016, rev. 8.

[49] Bitcraze Wiki, "Crazyflie 2.0 hardware specification," https://wiki.bitcraze.io/projects:crazyflie2:hardware:specification, July 2015, [Online; accessed 2016-12-30].

[50] ——, "Crazyradio PA," https://wiki.bitcraze.io/projects:crazyradiopa:index, July 2015, [Online; accessed 2016-12-30].

[51] "Loco positioning system," https://www.bitcraze.io/loco-pos-system/, [Online; accessed 2017-05-28].

[52] W. Hoenig *et al.*, "crazyflie_ros," https://github.com/whoenig/crazyflie_ros, 2016, [Online; accessed 2016-07-15].

[53] W. Hoenig, C. Milanes, L. Scaria, T. Phan, M. Bolas, and N. Ayanian, "Mixed reality for robotics," in *IEEE/RSJ Intl Conf. Intelligent Robots and Systems*, Hamburg, Germany, Sept 2015, pp. 5382 – 5387.

[54] A. Taffanel *et al.*, "Bitcraze UWB LPS position estimator ROS package," https://github.com/bitcraze/lps-ros, 2016, [Online; accessed 2016-07-15].

[55] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," 2012.

[56] L. Carlone, Z. Kira, C. Beall, V. Indelman, and F. Dellaert, "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 4290–4297.

[57] S. Liu and G. Trenkler, "Hadamard, khatri-rao, kronecker and other matrix products," *2008 International Journal of Information and Systems Sciences*, vol. 4, no. 1, pp. 160–177, 2008.

[58] "Smart mobility lab," https://www.kth.se/en/ees/omskolan/organisation/avdelningar/ac/research/control-of-transport/smart-mobility-lab/smart-mobility-lab-1.441539, [Online; accessed 2017-09-02].

TRITA EE 2017:143
ISSN 1653-5146