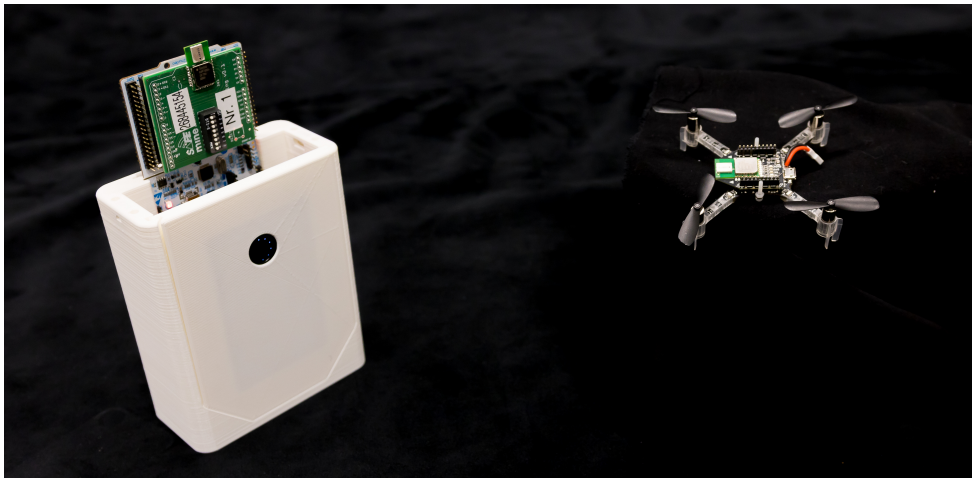
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Automatic Control Laboratory

Controlling Quadcopters using Ultra-wideband

Yvan Bosshard Michael Rogenmoser Tiago Salzmann

February 22, 2018



Abstract

The Automatic Control Laboratory (IfA) at ETH Zurich currently organizes a P&S course for Bachelor students on control of quadcopters. The system for this course currently uses Bitcraze's Crazyflie 2.0 quadcopters with three reflective balls attached, which allows for precise localization with a Vicon camera system. However, this system is rather expensive and a hastle to set up. The IfA has also been working on ultra-wideband (UWB) localization, developing a robust system to locate a tag using only UWB ranging. The aim of this group project was to combine the two other projects: locate and control the Crazyflie using only UWB and onboard sensors. While UWB is a lot less precise than the Vicon system, it is far more versatile and less expensive. Using an optimization-based approach, the location of the Crazyflie can be determined using only the distances between the Crazyflie and fixed anchors. As this information is quite flawed, the position has to be filtered and a controller optimized to work with this new method of localization.

Preface

Control theory research on quadcopters is becoming more and more popular due to their flexibility and complexity as a plant. Measurements and state estimation rely on fast and precise information processing and therefore need reliable and accurate sensor data. To provide localisation for this highly dynamical system a common approach is using vision based motion tracking by setting up a network of cameras to determine position and attitude inside a reference frame with millimetre accuracy. The distributed flying and localisation lab[1] uses a Vicon Motion Capture system with four *Vicon Vero v1.3*[2] cameras that provide the client with position and attitude information at 200 Hz. Although the system is able to provide highly accurate data, due to it's size, set-up time and price it is inconvenient for more diverse use scenarios. Ultra-wideband antennas on the other hand are small, lightweight and low-powered, which are key factors for the use on a quadcopter and the portability of the whole system.

Ultra-wideband (UWB) is a rather new technology in the consumer electronics market that was authorised by the Federal Communication Commission (FCC) for unlicensed use in 2002. Due to it's high bandwidth it can be used to send short-duration pulses which are suitable to do precise measurements of signal propagation time [7]. Using distance information to three or more known positions allows for the calculation of a unique position in 3D space. Fusing localisation information with on-board attitude sensors enables the system to work independent of Vicon.

[1] D-FaLL homepage (http://www.distfly.ethz.ch/index.php)

[2] see product page for more details (https://www.vicon.com/products/camera-systems/vero).

Contents

List of Figures

List of Tables

Chapter 1

The Idea

Previous group and semester projects at the Automatic Control Laboratory (Ifa) recently worked on and developed a P&S course on quadcopter control. A system (D-FaLL system) was developed to enable user-friendly access and programmability of the control algorithms. Figure 1.1 illustrates the basic setup of this system: Vicon cameras [5] allow for millimeter accuracy in both location and attitude estimation of a quadcopter in 3D space. The quadcopter used was the Crazyflie, described in Section 3.2.

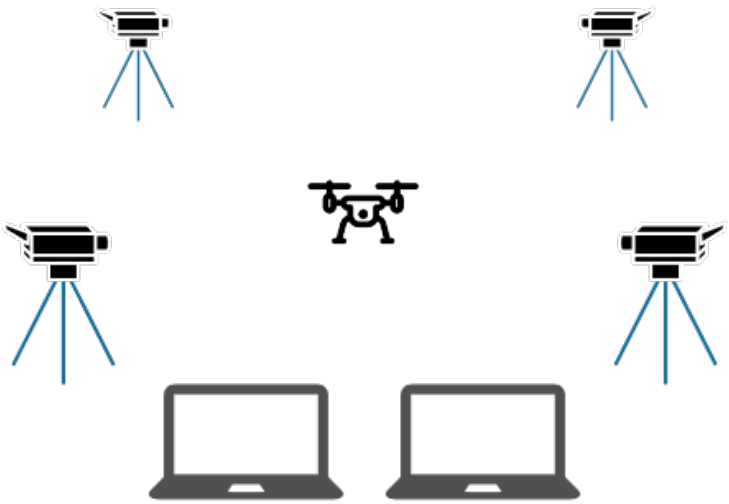


Figure 1.1: Schematic showing the original D-FaLL system

A different set of projects at Ifa concerned themselves with ultra-wideband (UWB) localization in 3D space. In particular, one group project developed an adaptable system for indoor localization, achieving position accuracy in the *dm* range [6]. The schematic shown in Figure 1.2 illustrates the basic principle for UWB localization: Using multiple fixed anchor positions, distances can be measured to an additional anchor in the enclosed space and by solving an optimization problem, the position can be estimated. This process is detailed in Chapter 2.

Figure 1.2: Schematic showing an UWB localization system

The main goal behind this group project was to combine these two systems: enable quadcopter control using UWB, eliminating the need for Vicon cameras. Figure 1.3 illustrates this goal of replacing the Vicon camera system, placing UWB Anchors in their position.

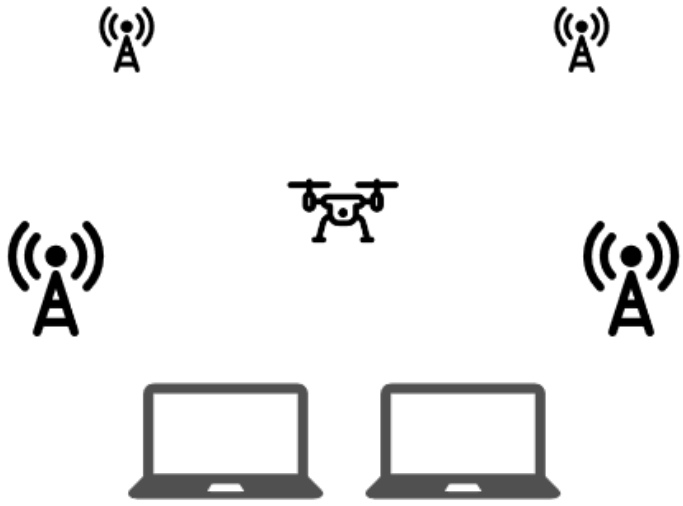


Figure 1.3: Schematic of the system developed

1.1 Advantages

There are multiple advantages to using such a system. Some are listed below.

Setup time The time required to set up the system is reduced: The Anchors have to be

placed in an appropriate arrangement and a quick self-calibration algorithm has to be run, eliminating the need for fancy dances with the Vicon wand.

Portability The UWB Anchors are more portable than the Vicon cameras as they are smaller in size and do not include a dedicated computer, which is needed to run the Vicon software.

Cable clutter While the Vicon system relies on tethered cameras and an addition computer to run, the UWB Anchors run on batteries and connect wirelessly. This not only reduces the cable clutter throughout the room, but also allows the system to be used independently of available power outlets.

Usage environments The entire system can be used in more diverse settings: While Vicon relies on infrared wavelengths, considering sunlight and reflections as difficult to handle interference, UWB considers most other signals as noise, only listening to other UWB signals (reflections of these signals can still be problematic). This means that theoretically, this new system could be used outside.

Cost The main advantage, however, is its price. While the Vicon system costs more than CHF 10'000.-, the price for the UWB system could be reduced to a few hundred CHF. This order of magnitude difference could enable widespread use of the localization technology.

1.2 Challanges

While there are quite a few advantages to this new system, there are considerable challenges that need to be overcome, listed below. Paramount among these is accurate, quick and reliable localization.

Accurate localization The position needs to be accurate to allow the controller to accurately control the quadcopter, such that the quadcopter ends at the intended position. While the Vicon system can attain millimeter accuracy, the UWB system at Ifa until now could only achieve 20cm accuracy in x- and y-direction, with an even worse 50cm accuracy in z-direction. This precision is insufficient for accurate control, however more precise estimates should be possible, especially as the sensors used claim 10cm accuracy in their distance measurements.

Fast localization The localization needs to be quick to enable feedback control. With a long delay in providing an accurate position, the given position will no longer be the true location, as the quadcopter will have moved since. Similarly, with a too low frequency providing accurate positions, the possible error resulting from incorrect control settings or other interference will not be corrected fast enough.

Reliable localization The location information has to be reliable, meaning errors in determining the position have to be filtered out, providing a best guess to the actual position.

Attitude estimation Attitude has to be estimated: The Vicon camera system provided the controller with accurate information on roll, pitch and yaw, which currently cannot be determined with UWB. For this, onboard sensors have to provide an estimate.

Compatability Ease-of-use and compatibility are another consideration, allowing for easy integration into the current P&S course the Ifa currently organizes.

Most of these challenges were tackled and a solution offered, all of which is detailed within this report.

Chapter 2

Theory

2.1 Localisation

The distance calculation between anchor and tag is based on a two-way communication where packets are sent between both devices and the travel time calculated. By using the known propagation speed of UWB-radio signals, one can calculate the distance. The method used in this project uses the concept of a (Double-sided) Two-Way-ranging, according to the Decawave User Manual [2].

2.1.1 Two-Way-Ranging (TWR)

As previously discussed, two-way-ranging is used to calculate the travel time of packets sent between the anchor and the tag. This method is based on a four packet communication, as seen in figure 2.1(a). In general, each packet stores three pieces of information:

Source Address Address of the device sending the packet. This is crucial for the next step, which is receiving an answer from the anchor/tag

Destination Address Address of the device receiving the packet. This variable is used to filter data flow, since all devices on receive mode are going to receive this packet, but only the one with corresponding address should answer.

Type The four different stages of the complete ranging: PING, ANSWER, FINAL and REPORT.

When a device receives a packet and needs to send an answer, it creates a new packet and assigns the swapped received source and destination addresses. The type would be the next stage of the ranging:

$$\begin{aligned}
&\text{Source_Address}_{\text{new}} = \text{Destination_Address}_{\text{received}} \\
&\text{Destination_Address}_{\text{new}} = \text{Source_Address}_{\text{received}} \\
&\text{Type}_{\text{new}} = (\text{Type}_{\text{old}} + 1)
\end{aligned}$$

The complete communication procedure for one distance calculation can be seen in figure 2.2. Tag starts by sending a PING packet and, upon receiving it, the anchor responds with an ANSWER packet. The tag finishes the ranging by sending a FINAL packet. At each step (upon receiving or sending a packet) each device locally saves the current

UWB Packet
Source Address
Destination Address
Type

(a) PING, ANSWER and FINAL packet

REPORT Packet
Source Address
Destination Address
Type
T_reply1
T_round2

(b) REPORT packet

Figure 2.1: UWB packet, used for the two-way-ranging (PING, ANSWER and FINAL), and the REPORT packet, used for the final step of sending the timestamps to the tag. The REPORT packet has two additional entries where the timestamps are appended (unsigned integers with 64 bits each).

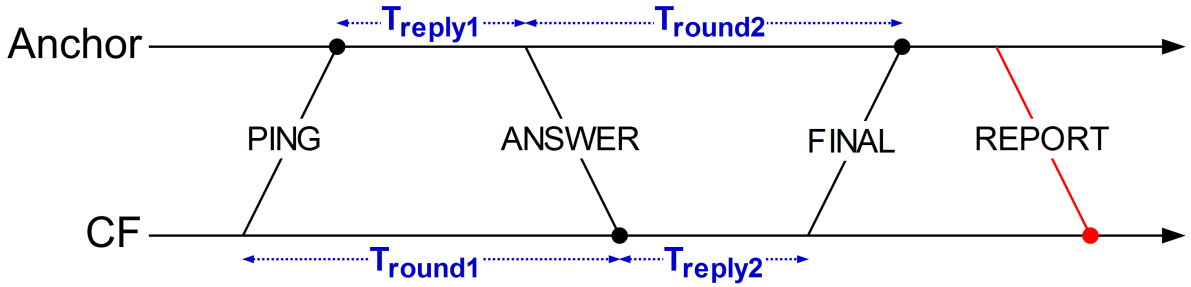


Figure 2.2: Complete two-way-ranging between Crazyflie and anchor. During the whole communication, in total two timestamps are saved on each device. the REPORT packet is used to send the two timestamps of the anchor to the Crazyflie.

timestamp. By calculating the time difference between two consecutive timestamps, it can get the absolute delta time of a single successful two-way communication, giving rise to the four delta timestamps seen in figure 2.2. These are the required parameters to calculate the traveling time.

Since all the timestamps are saved locally on different devices, the anchor has to send its two timestamps to the tag in order to use the equation for the propagation time. It does that by appending all the required data to a different packet, the REPORT packet, which has two additional entries (two 64 bit unsigned integers).
After having all the required information on a single device (in this case the Crazyflie), the final propagation time t_{prop} can be calculated using equation 2.1, taken from the DecaWave User Manual [2].

$$t_{prop} = \frac{t_{round1} \cdot t_{round2} - t_{reply1} \cdot t_{reply2}}{t_{round1} + t_{round2} + t_{reply1} + t_{reply2}} \tag{2.1}$$

2.1.2 Optimization-based triangulation

Given all the distances to all six used anchor points, the next challenge is to, using the known anchor positions, calculate the Crazyflies's position. Many different methods exist to do this, but for the sake of this paper an optimization solver will be implemented. The

problem to be solved can be formulated as a non-linear least-squares

$$\min \sum_{i}^{N} (d_i^2 - \|\mathbf{p}_i - \mathbf{p}_{CF}\|^2)^2 \tag{2.2}$$

where d_i is the distance between anchor i and the Crazyflie, $\mathbf{p}_i = [x_i, y_i, z_i]^T$ is the cartesian position of anchor i and $\mathbf{p}_{CF} = [x_{CF}, y_{CF}, z_{CF}]^T$ is the Crazyflies‘s position. Thus, d_i and $\mathbf{p}_i$ are fixed parameters in this least squares problem, and the $\mathbf{p}_{CF}$ are the variables to be determined which minimize (2.2).
Considering the generalized formal definition of a least-squares,

$$\begin{aligned} &\min \sum_{i}^{N} r_i^2, \quad r_i = y_i - f(\mathbf{x_i}, \boldsymbol{\beta}) \\ &m \text{ functions } \mathbf{r} = (r_1, ..., r_m) \in \mathbb{R}^m \\ &n \text{ variables } \boldsymbol{\beta} = (\beta_1, ..., \beta_n) \in \mathbb{R}^n \end{aligned} \tag{2.3}$$

where $\mathbf{x_i}$ is the independent variable and y_i is the dependent variable, a standard algorithm for solving these kind of problems is the Gauss-Newton algorithm, which has the iteration step shown in equation (2.4)

$$\boldsymbol{\beta}^{(s+1)} = \boldsymbol{\beta}^{(s)} + (\mathbf{J_r}^T \mathbf{J_r})^{-1} \mathbf{J_r}^T \mathbf{r}(\boldsymbol{\beta}^{(s)}) \tag{2.4}$$

where $\mathbf{J_r}$ is the Jacobi-matrix of the residual r with respect to the variables $\boldsymbol{\beta}$:

$$(\mathbf{J_r})_{ij} = \frac{\partial r_i(\beta^{(s)})}{\partial \beta_j} \tag{2.5}$$

Using the calculated distances and the known anchor positions, we can write the r_i function by defining

$$y_i = d_i^2, \quad \boldsymbol{\beta} = \begin{bmatrix} x_{CF} \\ y_{CF} \\ z_{CF} \end{bmatrix}, \quad \mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{2.6}$$

$$f(\mathbf{x}_i, \boldsymbol{\beta}) = (x_i - x_{CF})^2 + (y_i - y_{CF})^2 + (z_i - z_{CF})^2 \tag{2.7}$$

and finally find the Jacobi-matrix for our solver:

$$\mathbf{J_r} = \begin{bmatrix} 2(x_1 - x_{CF}) & 2(y_1 - y_{CF}) & 2(z_1 - z_{CF}) \\ 2(x_2 - x_{CF}) & 2(y_2 - y_{CF}) & 2(z_2 - z_{CF}) \\ \vdots & \vdots & \vdots \\ 2(x_N - x_{CF}) & 2(y_N - y_{CF}) & 2(z_N - z_{CF}) \end{bmatrix} \tag{2.8}$$

As for the initial condition, we may define any value that makes physical sense. Since the anchor points are normally placed within 10 meters of the center of the flying arena, setting an initial condition of $[5, 5, 5]^T$ for when there is no prior knowledge of the position may be a reasonable choice. After the problem has been solved before, the last recorded position can be used as the initial condition.

2.2 Kalman Filter

As suggested by early measurements of distances and position (see sec. 5), the data had to be filtered to acquire useful information from it. The parameters for the filtering were estimated using simulations (see sec. 5.3) and later fine-tuned on the actual system.

The use of a Kalman filter is justified by the assumption that a system with dynamics 2.9 evolves in time according to 2.10 with $\mathbf{w}$ being process noise covariance and $\mathbf{v}$ measurement noise covariance.

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k \end{aligned} \tag{2.9}$$

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{w} \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \mathbf{v} \end{aligned} \tag{2.10}$$

A prediction $\hat{\mathbf{x}}_{k+1}$ of the state variable is computed by applying the model as in equation 2.9 and the estimation covariance evolves as indicated in equation 2.11.

$$\hat{\mathbf{P}}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q} \tag{2.11}$$

After taking a measurement $\mathbf{z}_k$ of the state variable, it is updated by calculating the prediction covariance (2.12), the optimal Kalman gain (2.13) and finally updating the state variable and the estimation covariance (2.14 and 2.15) [9].

$$\mathbf{S}_k = \mathbf{R} + \mathbf{C}\hat{\mathbf{P}}_k\mathbf{C}^T \tag{2.12}$$

$$\mathbf{K}_k = \hat{\mathbf{P}}_k\mathbf{C}^T\mathbf{S}_k^{-1} \tag{2.13}$$

$$\mathbf{x}_k = \hat{\mathbf{x}}_k + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{C}\hat{\mathbf{x}}_k\right) \tag{2.14}$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\mathbf{C}\right)\hat{\mathbf{P}}_k\left(\mathbf{I} - \mathbf{K}_k\mathbf{C}\right)^T + \mathbf{K}_k\mathbf{R}\mathbf{K}_k^T \tag{2.15}$$

2.2.1 Distances

The filter for the distances assumes a static behaviour, expressed in equation 2.16, where $\mathbf{x} = \left[\hat{d}_i\right]^T$, $i \in [1, 6]$. This is justified by the fact, that the true distances only vary slightly between measurements, which can be corrected by the filter. Using the simulations, values for process and measurement noise covariances (w, v) could be estimated, leading to the matrices 2.17. Since the distances are needed to determine the 3D location, the individual states correspond also to the output, thus $\mathbf{C} = \mathbf{I}$. The measurements $\mathbf{z}_k = [d_i]^T$, $i \in [1, 6]$ are the values calculated by actually measuring the time of flight using the UWB sensors.

$$\mathbf{x}_{k+1} = \mathbf{x}_k \tag{2.16}$$

$$\begin{aligned} \mathbf{Q} &= w\mathbf{I} \\ \mathbf{R} &= v\mathbf{I} \end{aligned} \tag{2.17}$$

2.2.2 Position

Considering the fact, that the implementation of the controller is part of the P&S course, and due to the late implementation of the filter, again a very simple model was used. The dynamics are the same as in equation 2.16 with $\mathbf{x}_k = \mathbf{r}_k = (r_x, r_y, r_z)_k^T$. The values for process and measurement noise covariance (w', v') were also estimated using the simulations with similar matrices $\mathbf{Q}'$ and $\mathbf{R}'$ (2.18). Again the internal state represents the desired variables for further processing ($\Rightarrow \mathbf{C}' = \mathbf{I}$).

$$\begin{aligned} \mathbf{Q}' &= w'\mathbf{I} \\ \mathbf{R}' &= v'\mathbf{I} \end{aligned} \tag{2.18}$$

Chapter 3

Hardware

3.1 Anchors

The Anchors used for this project were developed for the previous projects on localization using UWB [6]. It consists of a microcontroller (ARM Cortex-M4 on a STM32 Nucleo board) and an UWB module, described in section 3.1.1. These were connected through a breakout board which incorporates switches for setting the device address. Together it is powered by a large USB battery and attached to a 3D-printed case. This setup, placed on a plastic pole, can be seen in Figure 3.1.

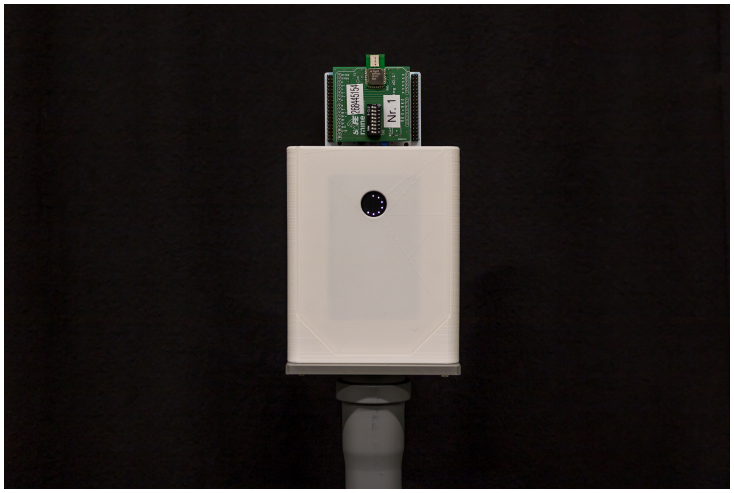


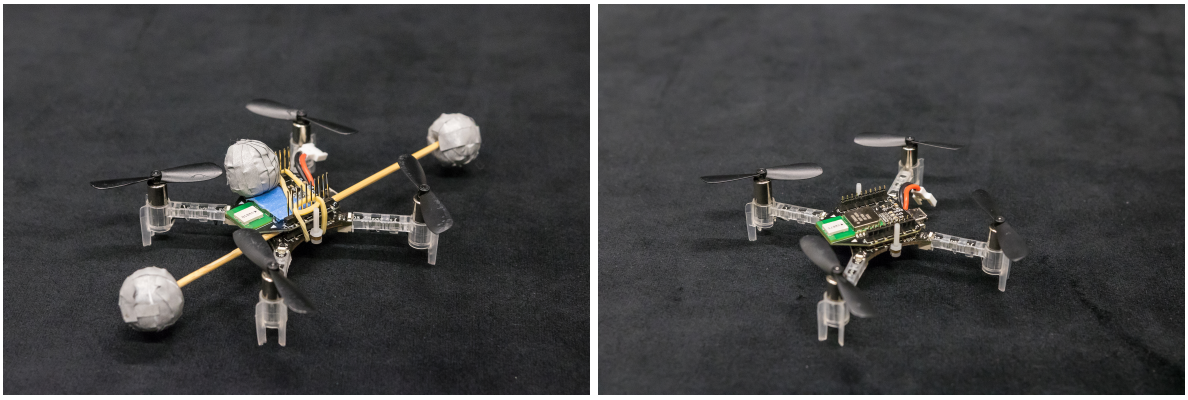
Figure 3.1: UWB anchor

3.1.1 DecaWave DWM1000

The DWM1000 module from DecaWave [1], the state of the art UWB module, was used for this project. It incorporates the DW1000 integrated circuit [2], also from DecaWave, and an UWB antenna. The DW1000 has many different functions, allowing data transmission of up to 6.8 Mbps and ranging (distance measurement) with up to 10cm precision. It communicates with the microcontroller through SPI.

3.2 Crazyflie 2.0

The Crazyflie 2.0 was developed by Bitcraze as a quadcopter development platform [3]. It is small and lightweight yet still powerful and precise, ideal for small environments. The entire code is open-source, ideal for development and educational purposes. The Crazyflie incorporates a powerful microcontroller (ARM Cortex M4) and an inertial meausrement unit (IMU). It has expansion pins on top, allowing for *expansion decks* to be attached, expanding its capabilities. One such expansion deck contains the same UWB module used for the anchors, allowing communication and ranging between them. In Figure 3.2 the Crazyflies can be seen. One of them has Vicon tracking markers attached, which were used for testing and comparing to a more precise position estimate.



(a) A Crazyflie with Vicon tracking markers (b) A Crazyflie without markers

Figure 3.2: Both have the UWB module attached

3.3 System

The D-FaLL system is based on a teacher student course environment, implemented with the Robot Operating System (ROS) [4]. A teacher computer receives location information from the Vicon system and passes this information along to all connected students. The teacher computer also sets areas in which each student's Crazyflie is allowed to fly. The student computer then uses the location information, implements a controller and sends basic motor commands to the Crazyflie using a dedicated Bluetooth radio. This information flow had to be adapted to use the UWB data, as described in chapter 4.

Chapter 4

Implementation

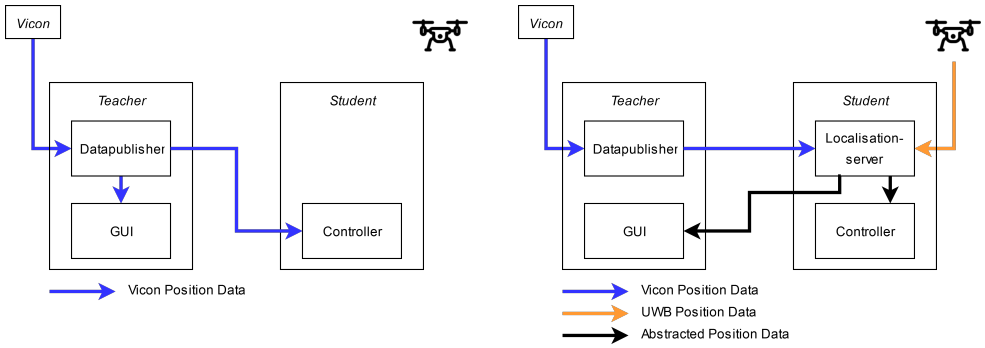
In order to use the system independent of Vicon and make the use of UWB convenient, a few major changes need to be mentioned. The usage of the Vicon camera system is deeply nested in the data flow of the whole system and many settings depend on its presence.

4.1 Data Flow

Without the UWB data in the system, the position data flow used to be hierarchical. A schematic of the information flow is illustrated in figure 4.1(a). The computer connected to the Vicon cameras calculates the position and angles of every detected Crazyflie and provides them over the network to the teacher computer. There a ROS-node converts this data into a data structure containing information about every Crazyflie and that again is provided over the network to the student computers. The controller running on every student computer has to then extract the dataset corresponding to the Crazyflie assigned to the student before using the data itself. Synchronously the data is also provided to the teacher GUI to present the teacher with a situation of all the quadcopters currently in use.

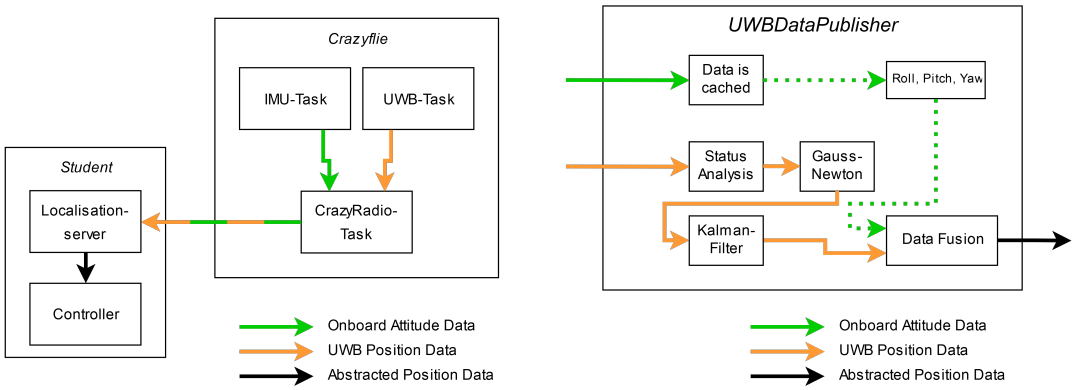
Without Vicon all the data needed to provide the controller with 3D coordinates and attitude information had to be attained by the Crazyflie and somehow fed to the controller. Since every quadcopter communicates via Bluetooth only with the student computer it was assigned to, the data has to be transmitted to the computer. A set of ROS-nodes (see sec. 4.1.1 for details) was added to the student program that collects all essential data, converts them into a data structure that is understood by, and finally provided to the controller.

In order to allow the students to chose whether they want to use Vicon or UWB, data from both localisation sources come together in the *Localisation-Server* (see fig. 4.1(b)) and converted into an abstract format that can be redirected to the controller. Additionally, an external ROS-node (not depicted in figure 4.1) gathers position data from the Localisation-Servers of every student and provides this information to the teacher GUI, such that situational awareness is provided even though students might use UWB data.



(a) Data flow before incorporating UWB data (b) Data flow after incorporating UWB data

Figure 4.1: Schematic of the controller data flow



(a) Data acquisition (b) UWBDataPublisher

Figure 4.2: Schematics of the flow of UWB position data

4.1.1 Position and Attitude

The fundamental data used to determine the position of the quadcopter in 3D space are the distances to the six UWB anchors. Onboard the Crazyflie a process is running that performs the TWR and saves the measured distances in a designated data structure, waiting for a request triggered by the student computer. Every 15 ms the distances to all the anchors are polled from the quadcopter and used to calculate the position.

Due to limited packet size of the Bluetooth link the attitude can not be polled simultaneously with the anchor distances. The angle data is polled at the same rate as the distances and later fused with the position data. The attitude is calculated onboard by a designated task involving the IMU measurements (see fig. 4.2(a)). The *CrazyRadio-Task* is the process handling the communication with the computer.

4.1.2 UWBDataPublisher

Once the data is loaded on the student computer it is provided to a ROS-node called UWBDataPublisher. Its main tasks are the calculation of the position and the fusion of attitude with position data. As the data is polled serially from the Crazyflie the dataset cannot be created at once and therefore to be able to fuse the data, attitude information

Teacher \ Student	ON	OFF
ON	UWB	Vicon
OFF	Vicon	Vicon

Table 4.1: Localisation sources depending on the selected state

is temporarily cached. Once an update of the anchor positions is available the position is calculated before providing the complete data to the controller.

Figure 4.2(b) visualises the data flow from the raw information to the final dataset. An additional stage before position calculation is illustrated as *Status Analysis*. This stage in the localisation process analyses the measured anchor distances and judges their viability (see also sec. 5.3.2). In case of errors or bad readings a message is printed to the console exposing the malfunctioning anchor(s).

4.1.3 LocalisationServer

The LocalisationServer is a ROS-node created to integrate the possibility to chose whether to use Vicon or UWB data. It acquires data from the *ViconDataPublisher* (see fig. 4.1(b)) and the *UWBDataPublisher* which allows both datasets to be forwarded to the controller. Inside the teacher GUI, a checkmark indicates if the use of UWB is permitted, whereas in the student GUI, a checkmark marks the selected localisation source (i.e. if UWB is actually used). If the use of UWB is not permitted, the student cannot change the state and the default localisation source, Vicon, is used. In the case that UWB can be used, the toggling of the checkmark switches between both data sources. Table 4.1 illustrates the states the system can be in and the used localisation source in every state.

4.1.4 TeacherService

When Vicon used to be the only localisation data source, the data of every Crazyflie was provided synchronously by the *ViconDataPublisher*, that also delivered data to the teacher GUI to provide an areal situation to the teacher. Since using UWB forces the localisation to be distributed and asynchronous, the previous approach of how data is handled does not work. The TeacherService, a ROS-node running on the teacher computer, solves the problem by acquiring position data from every students *LocalisationServer* and combining the asynchronous data into a data structure containing data of every student. The data is then forwarded to the teacher GUI to provide the same information even with students flying with UWB data.

4.2 Working with UWB Data

In order to provide a convenient use and fast setup, the workflow for the self-localisation of the UWB anchors was integrated into the teacher GUI. A tab containing all control elements of the UWB localisation was added, together with status information. In figure 4.3 the key elements of working with UWB can be seen:

Calibration To calibrate the anchors (i.e. run the self-localisation, described in section 4.3) a *Run*-button triggers the process. A checkmark next to it indicates a successful calibration.

Manual Positioning Should the calibration fail, the possibility exists to determine the positions of the anchors manually and enter the data into a YAML-file that can be loaded using the *Load Anchor File*-button.

Anchor Positions In either case, the positions of the anchors are displayed in the table beneath the calibration section.

Customisation To customise the coordinate system an offset can be entered for every coordinate and applied using a button next to the specified offset. If all reference anchors are placed on the same plane, the z-component of the position is ambiguous (i.e. it can be above or below the reference plane). To correct for falsely determined height every anchor position can be mirrored through the xy-plane.

Enable UWB The checkmark indicates if the use of UWB is permitted. It determines which localisation sources can be used by the students (consider table 4.1).

Upon launch of the teacher GUI only the buttons for self-localisation and loading the YAML-file are enabled. Only after anchor positions are acquired either way, UWB can be enabled. In case of manual entry, no offset can be set and inverting a z-component is not possible.

Any change in UWB settings is announced to the student computers to allow them to fetch any changed information (e.g. new anchor locations, the permission to use UWB).

4.3 Self-Localisation

The teacher computer runs a ROS-node called *UWBManagerService* that handles all UWB related settings. Every change made in the teacher GUI is communicated to this node where the UWB status, stored internally, is updated. Student nodes and the teacher GUI can request the current status containing the position of the six UWB anchors as well as two binary flags, one indicating the permission to use UWB, the other providing feedback regarding a successful self-localisation.

The idea behind the anchor self-localisation is to create a more portable, fast and efficient way for setting up the flying arena. By using three fixed anchor points as reference

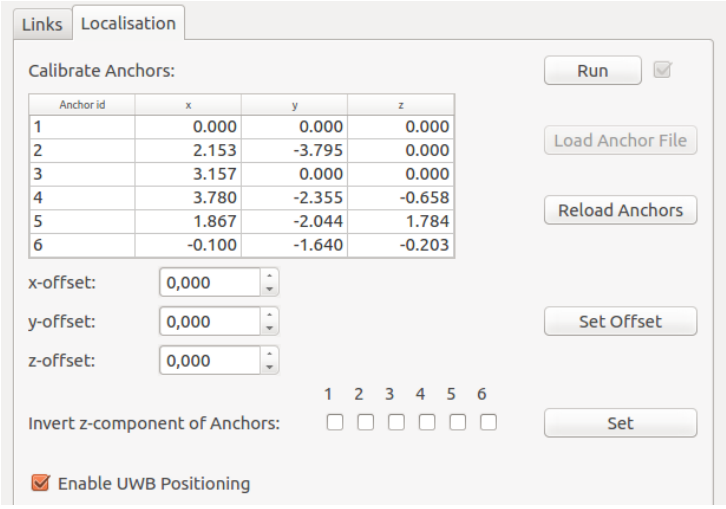


Figure 4.3: Localisation tab in the teacher GUI

(which are going to span the cartesian x, y and z reference directions), the objective is to find the remaining 3 randomly placed anchor positions. Similar to the Crazyflie, an optimisation-based solver was implemented.

4.3.1 Reference Anchors

As mentioned above, the system consists of three fixed and three randomly positioned anchor points, which are usually placed using the configuration depicted in figure 4.4(a). The term "fixed" in this sense means that these anchor points present a condition on their arrangement, since this arrangement builds the cartesian coordinate system used by the localisation service. These anchors may also be placed in random positions, but the order and orientation should be taken into account in order to take note of the resulting reference coordinate system.

In general, the three first anchors (IDs 1, 2 and 3) are set to be the fixed reference anchor points. By measuring the distances d_{12}, d_{13} and d_{23} and defining their positions as $(x_1, y_1, z_1) = (0, 0, 0)$, $(x_2, y_2, z_2) = (x_2, y_2, 0)$, as well as $(x_3, y_3, z_3) = (d_{13}, 0, 0)$, it is possible to define a unique cartesian coordinate system with origin at anchor 1, as it can be seen in figure 4.4(b). Note that the position of anchor 2 does not take directly the measured distances, but with simple trigonometry it is possible to find its location.

4.3.2 Anchor-Anchor Distances d_{ij}

In order to use the solver, all required distances d_{ij} need to be measured. This process is started by a special packet sent by the Base Station connected to the Teacher computer. This "localisation" packet is sent to only anchor 1, which is going to take the Crazyflie's role of Tag and start a ranging with each single anchor point. Upon measuring all distances d_{1j}, $\forall j$, anchor 1 forwards this localisation packet to the next one (in this case anchor 2) and all the measuring starts over. This process repeats for all anchor points. After each complete ranging, the corresponding anchor point sends the measured distance to the Base Station, which averages and filters the data.

4.3.3 The Self-Localisation Solver

Considering all 9 position components that need to be found, the solver would have in total 9 variables (all the three components of the three anchor points) with 12 functions (all needed distances between anchor points d_{ij}, $i \neq j$).
One problem that can arise by trying to solve this least-squares is the initial condition. By using 0 as the initial condition for all variables, and taking into consideration that the reference anchor 1 is at position $(x_1, y_1, z_1) = (0, 0, 0)$, the solver may return some errors regarding the computer's precision. The following $\mathbf{J_r}$ matrix is an example for this case:

Here we consider all initial conditions to be zero and the following reference anchor positions (values in meters)

$$\begin{aligned} \text{Anchor 1} &\rightarrow (x_1, y_1, z_1) = (0, 0, 0) \\ \text{Anchor 2} &\rightarrow (x_2, y_2, z_2) = (5, 10, 0) \\ \text{Anchor 3} &\rightarrow (x_3, y_3, z_3) = (7.5, 0, 0) \end{aligned}$$

$$\mathbf{J_r} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 & 20 & 0 \\ 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Upon calculating $(\mathbf{J_r}^T \mathbf{J_r})^{-1}$ using MATLAB, one gets the following error:

```
1 inv(J'*J)
2 Warning:  Matrix is singular to working precision.
```

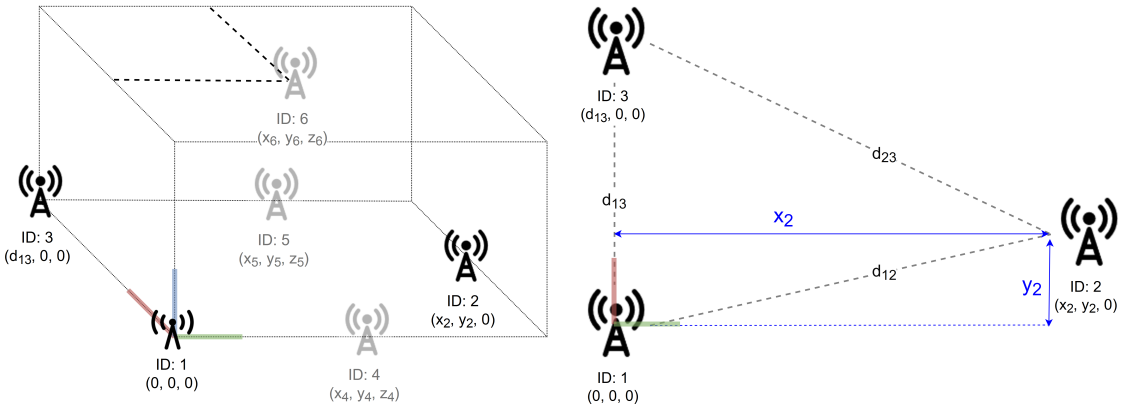
basically meaning that the entries of the inverse are either too small or too big to be represented using the standard variable types (in this case a double[1]).

The rather extreme behavior of the entries can be explained by interpreting the physical aspect of this problem. By setting all initial conditions to 0 we are basically saying that all three anchor points are all on the same position, although all distances between anchors are never zero ($d_{ij} \neq 0$).

4.3.4 The Solution - A simpler problem

Many possible solutions were explored by directly manipulating the $\mathbf{J_r}$ matrix to try to regularize it, some of them being inserting random or fixed non-zero initial conditions or

[1] double-precision floating-point representation (64 bits)



(a) Three dimensional representation of the positioning of all six anchor points

(b) Reference anchor points

Figure 4.4: Reference (fixed) and the remaining three anchor points. With simple trigonometry it is possible to find the position of anchor 2 using only the distances to anchor 1 and 3. By using the position of these three fixed anchor points one can use an optimization-based triangulation to locate the remaining three.

shifting the reference anchor points. But these methods turned out not to be as reliable, as the inserted values did not always lead to an invertible matrix $(\mathbf{J_r}^T\mathbf{J_r})^{-1}$ in the sense of not getting an almost singularity warning.

A more reliable solution would be feeding the initial condition of the solver with the position estimation of a less accurate method. In this case, the solver would be improving this position estimation towards a more precise value. The less accurate method explored in this project was also a least-squares solver, but involving only one non-fixed anchor. The idea is to use the three fixed reference anchor points to separately find the positions of the non-fixed anchors. This simpler solver has therefore only three variables, three functions and needs to be solved three times in total (once for each anchor). Regarding the almost singularity warning, the inverse $(\mathbf{J_r}^T\mathbf{J_r})^{-1}$ was (according to all the simulations and tests done), always successfully calculated for all used initial conditions, even with $(0, 0, 0)$.

It turns out that this simpler solver gave initial values that basically did not change after improving it with the full solver, meaning that the values were already precise enough. Because of that matter, the actual implemented anchor self-localisation service does not include the full solver anymore, getting position estimations using only the simpler version by individually finding the anchor positions.

4.4 Crazyflie‘s Z-Component

While the measured distances between the anchors and the Crazyflie are pretty accurate, the calculated z-component given by the solver always showed some discrepancies compared to reality. This may be caused by the anchor arrangement used, in which practically

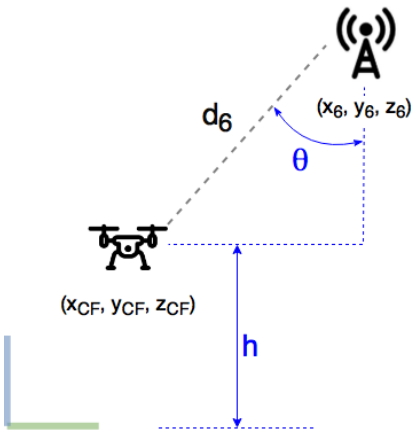


Figure 4.5: z-component of the position calculation. By using the measured distance d_6 and the (x, y) position of the Crazyflie (x_{CF}, y_{CF}) and anchor 6 (x_6, y_6), it is possible to estimate the z-component of the Crazyflie's position $z_{CF} = h$

all anchors are on the same height, making it unfavorable for the solver to get precise values for the normal component to the plane. But changing the arrangement (by placing the anchors in different heights) did not show any noticeable difference.

By exploring the fact that the measured distances between all devices are accurate enough, the solution was to use only one anchor and its (x, y, z) position as well as the (x, y) position of the Crazyflie, (x_{CF}, y_{CF}), to caluclate the z-component, as shown in figure 4.5. The z-component of the solver is still calculated, but totally ignored.

4.5 Remaining Constraints

Flying only using only UWB data is possible considering the previously described implementations. However running the system without the Vicon computer connected is still not possible. The reason for that is how the assignment of devices to the students works. Every quadcopter used in the P&S-system is registered with its unique alignment of reflective markers on the Vicon computer together with a unique character string (the 'name' of the Crazyflie). This name is used to assign a Crazyflie to a student and lets the controller determine which position data needs to be considered.

In order to communicate with the quadcopter, the communicating ROS-node has to know the unique Bluetooth address of its assigned device. This second relation again is formed by pairing the name with the address. Finally all quadcopters detected by Vicon are listed and one chosen to be assigned.

Since only quadcopters in the Vicon database can be assigned, the relation of a physical device to a student had to be changed in order to fly without the Vicon computer connected. The Bluetooth address now is the primary identifier of every quadcopter, which is related to a still unique name on the teacher computer. Every Crazyflie contained in

this database can be assigned, independent of Vicon containing an entry for that particular name. If Vicon should still be used, the name has to match the one on the teacher computer.

Chapter 5

Measurements

To evaluate the accuracy and performance of UWB localisation a mean for comparison had to be found. Since the Vicon-system offers sub-millimetre accuracy on position and very precise attitude its data was used as a reference. A ROS-node was incorporated into the system to be able to log data from Vicon and the estimated position and on-board attitude using UWB, as well as distances to all the anchors. Since Vicon data is being published at a rate of 200 Hz its data was cached until an UWB data update (i.e. distances to all the anchors) was received. Both datasets were then logged into separate files together with a timestamp in a format that could later be read by a MATLAB script. This allows to relate the estimated position to the underlying data (the anchor distances) and to the true distance, by comparing position of the anchors to the Vicon position.

5.1 Trajectory Plotting

A first analysis of the performance can be done using a plot of the recorded trajectory. Using a script that loads files containing Vicon and UWB data as well as one containing the anchor locations, an animated plot can be drawn. The temporal component of the 3D plot allows to determine where and when major differences in the estimation versus the true location happen. One of the first recordings can be seen in figure 5.1, where the animation revealed that the position estimation started being chaotic when flying too close to one of the anchors.

It is important to understand the difference between flying with Vicon data and plotting UWB data, and the opposite, flying with UWB data and plotting Vicon data. The former illustrates how UWB position is affected by the position of the Crazyflie, the latter can be used to analyse how position is affected by flawed UWB data. Figure 5.2 shows a recording using UWB data where the temporal component of the plot indicates which part of the estimations during the run were responsible for abrupt manoeuvres.

Final recordings show marked improvement of tracking and flying performance using UWB data, such that staying at a fixed setpoint is possible in most scenarios. The plot in figure 5.3 indicates, that the trajectory of UWB data matches the Vicon trajectory closely.

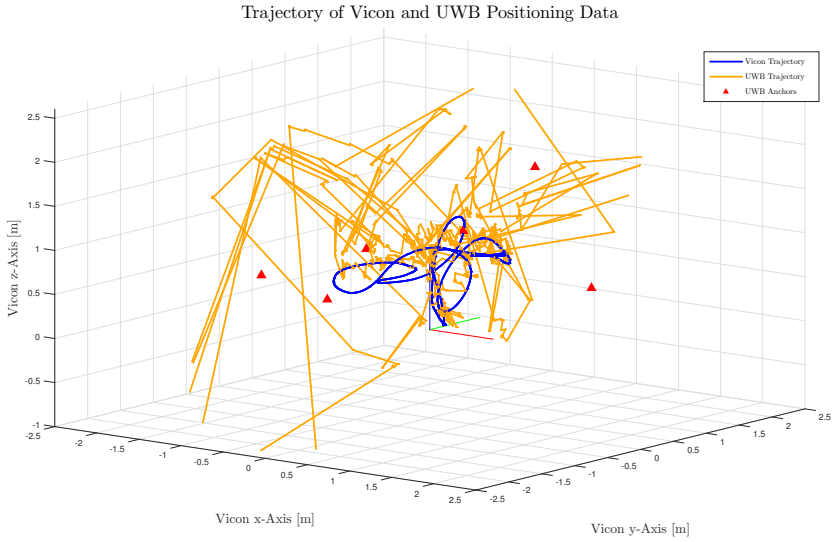


Figure 5.1: Early position recording, flown using Vicon data

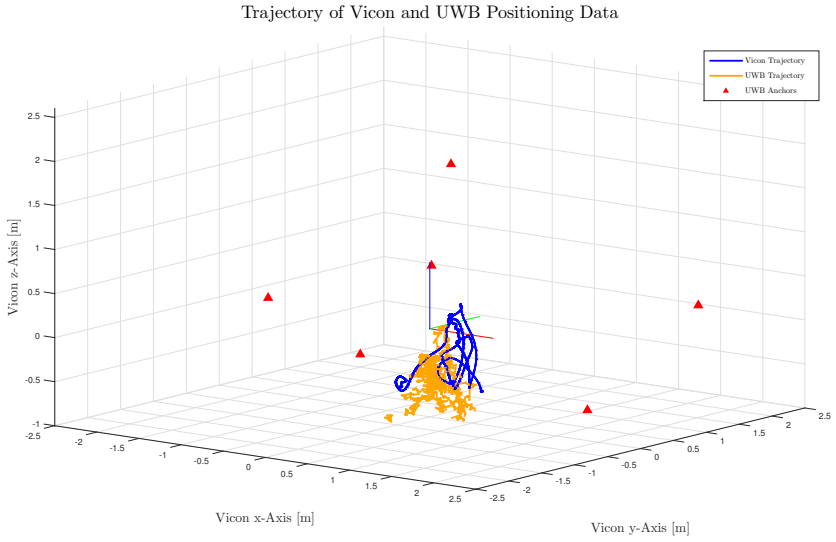


Figure 5.2: Position recording, flown using UWB data

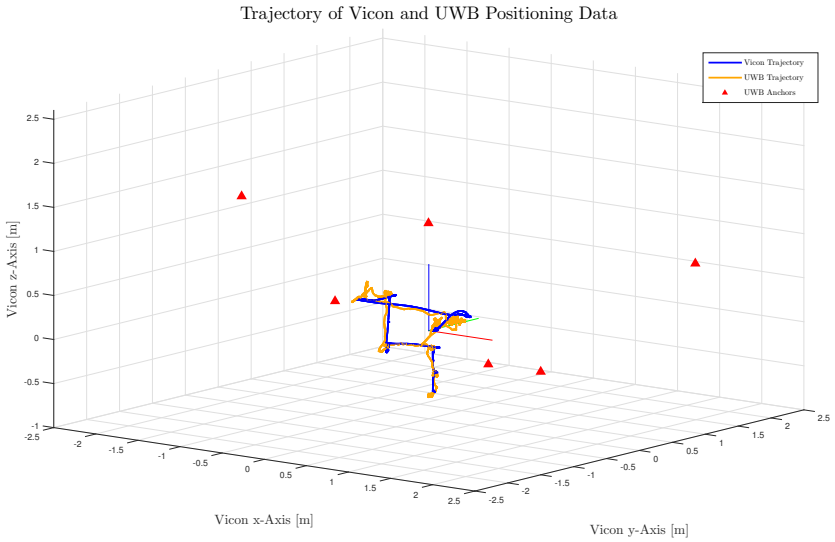


Figure 5.3: Latest position recording, flown using Vicon data

5.2 Error Analysis

Since the controller responsible for the tracking of a given setpoint uses 3D coordinates and three angles around the three axis of the inertial frame of the Crazyflie, there are six variables that have to be accurate. Even though the individual variables are not independent of each other using a more advanced model of the dynamics of the Crazyflie, an approach with separated variables was chosen to filter them. To analyse the error in each variable the 3D coordinates and attitude angles were separately considered.

5.2.1 Position

An initial comparison for the behaviour of the position error can be gained by considering a plot of all error vectors during a recording. The error vector at time t is defined as

$$\mathbf{r}_{error,t} = \mathbf{r}_{Vicon,t} - \mathbf{r}_{UWB,t}, \tag{5.1}$$

such that the set of all error vectors can be visualised in a scatter plot (see fig. 5.4). A uniformly distributed error would result in a sphere around the origin of the plot whereas a shift of the centre of the sphere would indicate a constant offset.

To analyse the individual coordinates and to estimate filter parameters a histogram of the error vector was plotted. A shift of the mean value again indicates a constant offset. In figure 5.6 the histograms for every spatial coordinate of an early recording is plotted. The larger offset in z-direction is present because of the fact that the z-component of UWB data offset was not set correctly. The standard deviation of every component can be used as an initial guess for the covariance of the measurement noise.

After implementing a Kalman-filter on the three coordinates, the difference in performance can be seen by comparing the scale of the axes in figure 5.5 and also the standard deviation in figure 5.7 compared to earlier recordings.

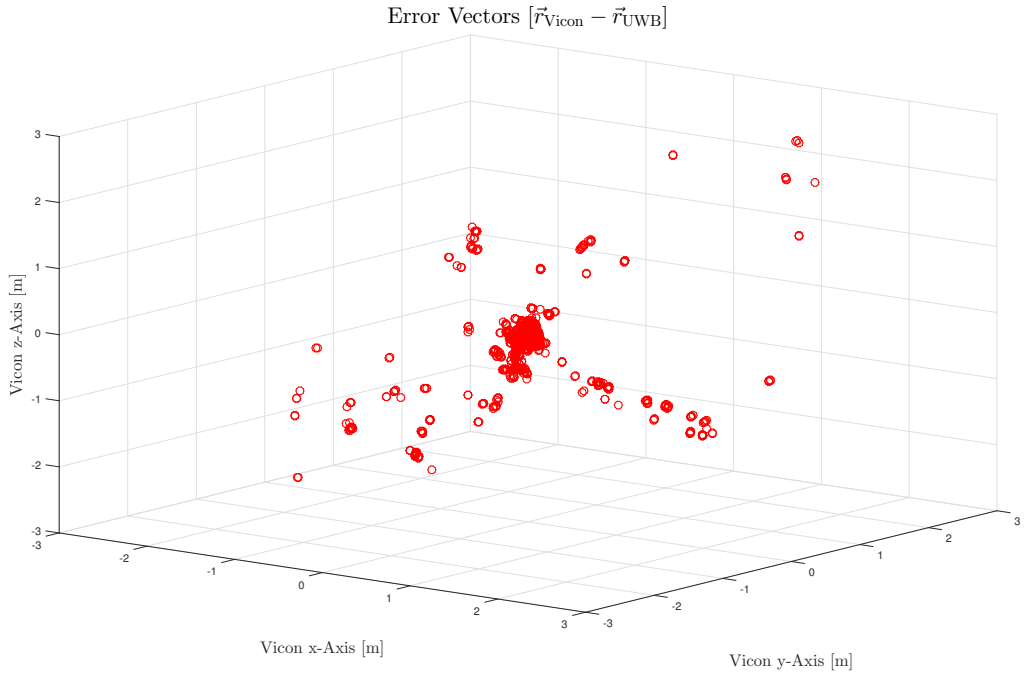


Figure 5.4: Point cloud of the error vectors of an early recording

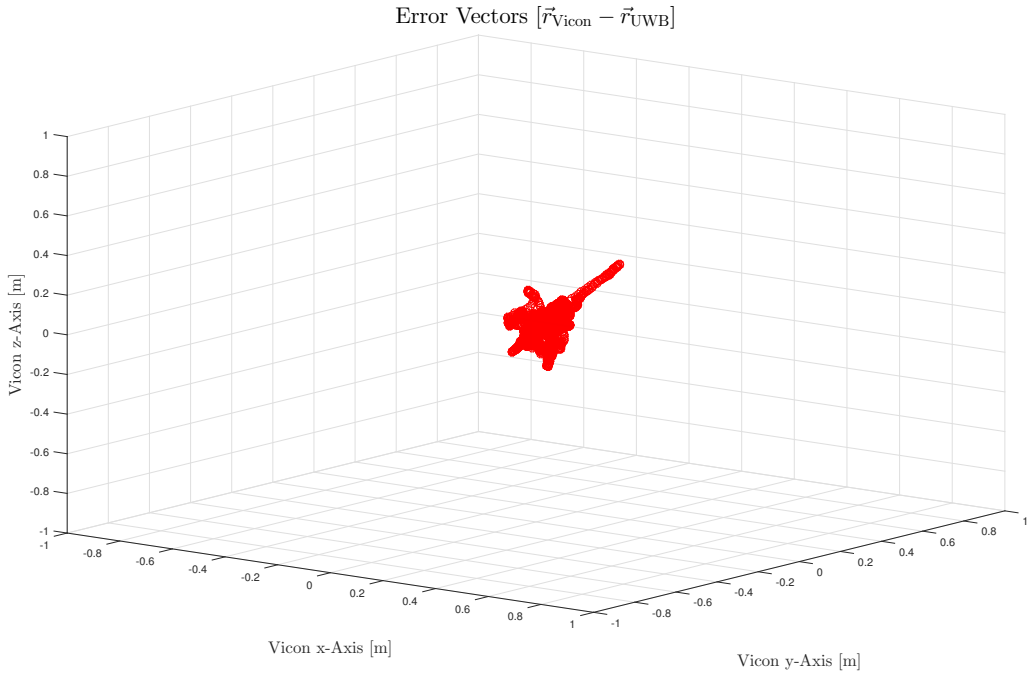


Figure 5.5: Point cloud of the error vectors of a late recording

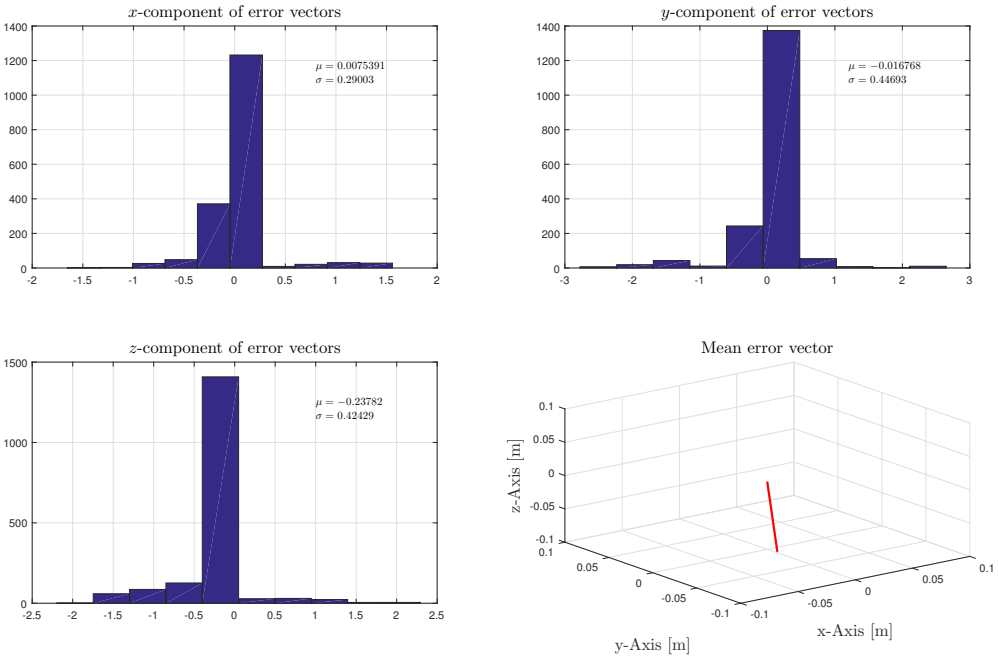


Figure 5.6: Error histogram of an early recording

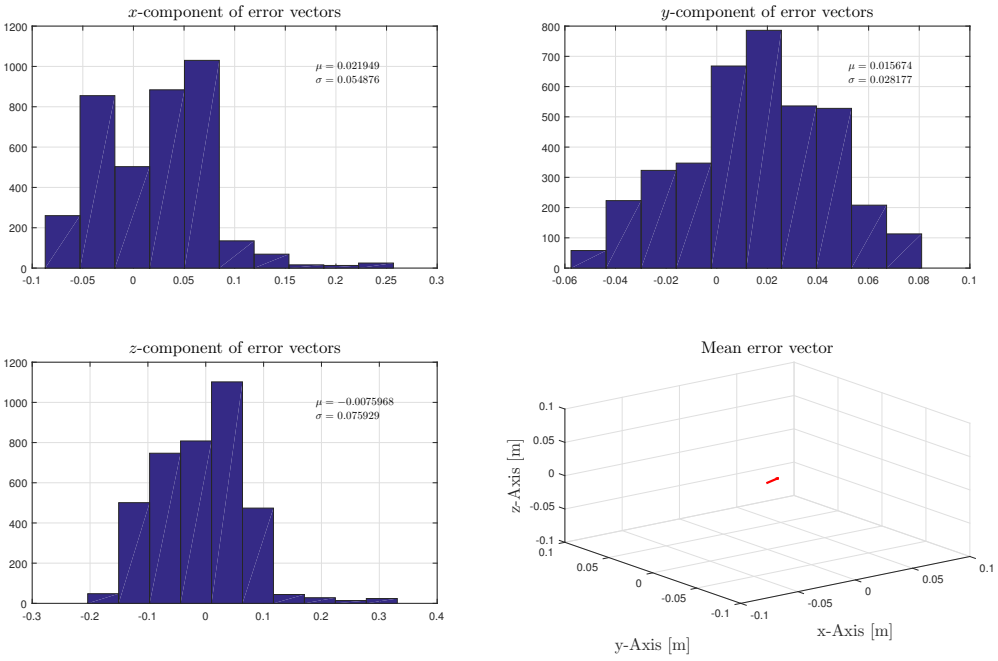


Figure 5.7: Error histogram of a late recording

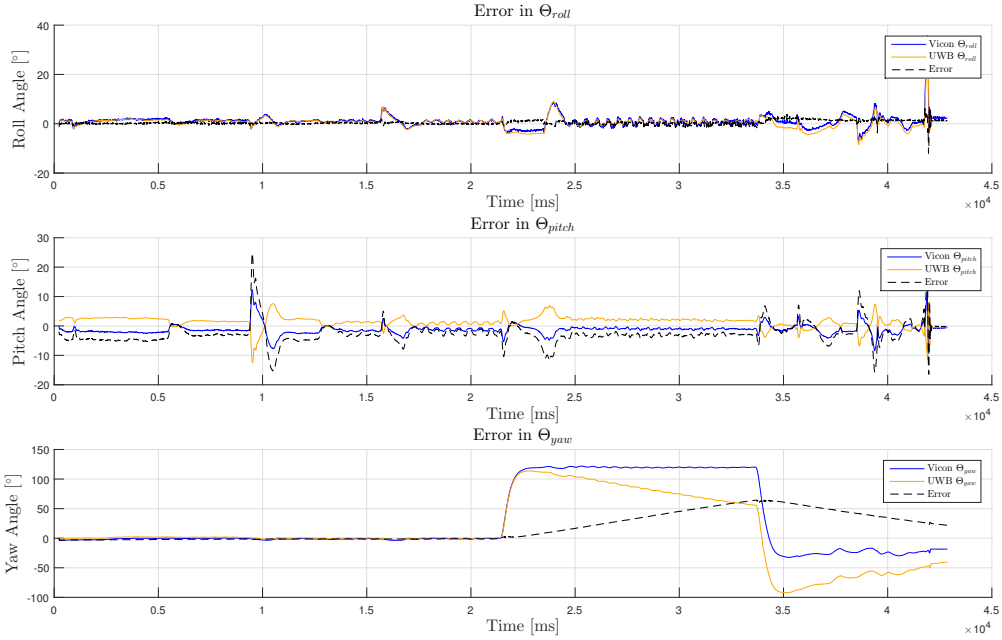


Figure 5.8: Uncorrected attitude measurements

5.2.2 Attitude

The Crazyflie firmware contains its own routines to calculate roll, pitch and yaw angles using the onboard IMU. Initial attempts replacing Vicon measured attitude included only one angle to be used from the IMU. The performance of roll and yaw was not well enough to fly with this data and pitch was impossible, since the quadcopter crash as soon as onboard pitch was used. Initially an *attitude heading reference system* was considered to improve the performance implementing a Madgwick-filter [8]. The expected improvement in the stability of the quadcopter was not achieved as could be seen by plotting the measured angles versus Vicon angles. This might come from the fact that this filter was not developed for the application in a reference frame moving laterally. However comparing stock angle measurements to Vicon data highlights some of the problems faced before implementing the filter. Figure 5.8 reveals three important aspects of using onboard angle data: The first graph shows that the roll angle is very accurate, even after very fast oscillations. The second highlights that flying with pitch was impossible since the onboard data used an inverted definition of this angle. Finally the yaw angle is accurate as long as there is no fast change around this axis. If the yaw angle changes to fast the measurement starts drifting and diverging from the true value.

The pitch angle can be corrected easily by multiplying the angle with a factor of -1 before using it in the controller and by limiting the turn rate around the yaw angle the chance of it starting to diverge can be reduced. Despite the value not drifting by controlled acceleration this does not prevent the same problem to arise if the Crazyflie is brought into rotation by an external force which it has to compensate for. Comparing the measurements in figure 5.9 to the initial data it can be seen, that the error is relatively small.

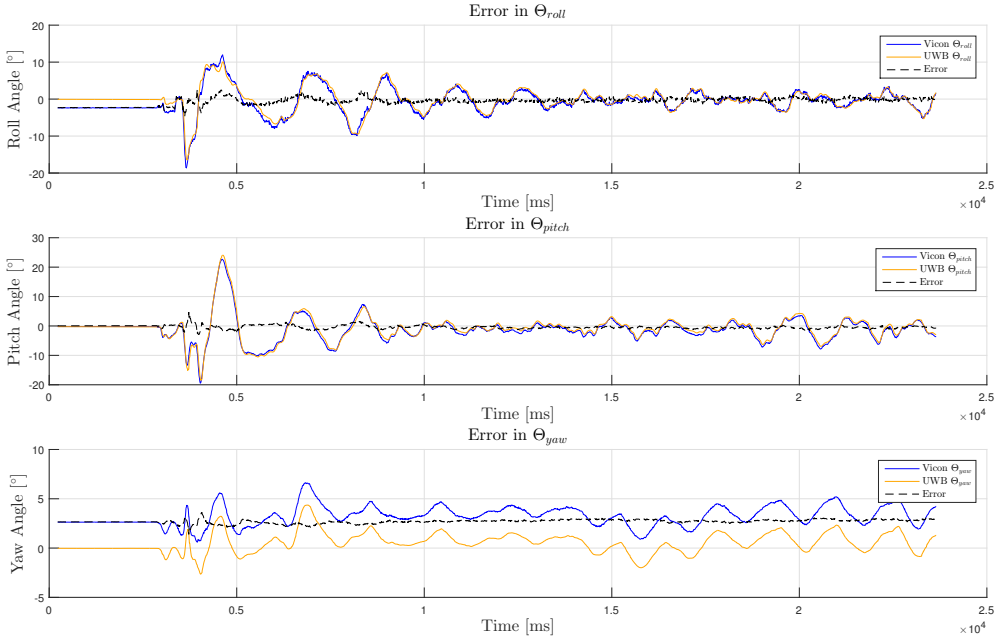


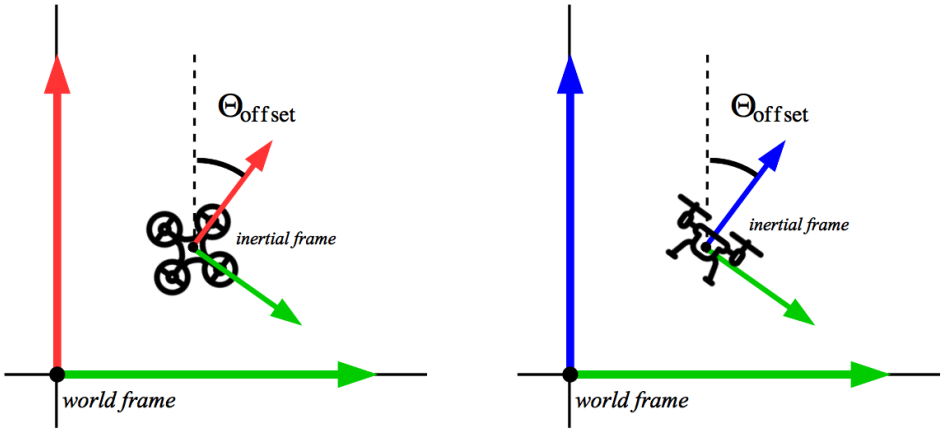
Figure 5.9: Corrected attitude measurements

In the graph of the yaw angle in figure 5.9 the error has a constant offset of around 3°. The reason for that offset is the alignment of the quadcopter when it is turned on. The onboard yaw angle is calculated using the inertial x-axis during initialisation as a reference and during flight, the deviation of the x-axis describes the current yaw angle. The controller on the other hand uses yaw angles relative to the positive x-axis of the world frame such that misalignment during initialisation leads to this offset. A schematic explanation can be seen in figure 5.10(a).

The problem of an angle offset arises for all three angles (see also fig. 5.10(b)) but since the surfaces the Crazyflie takes off from are nearly parallel to the xy-plane of the world coordinate system these effects are rather small. Still a level positioning of the Crazyflie during the initialisation greatly improves stability in all three angles.

5.3 Filtering Simulations

The trajectory analysis described in section 5.1 suggested during early testing (see fig. 5.1) that position had to be filtered. A Kalman-filter applied to each of the three coordinates separately improves the performance considerably. Due to the implementation of the filter inside a compiled program without the possibility to quickly adapt filter parameters it was convenient to test different parameters in a simulation first, before fine tuning them on the real system.



(a) Yaw offset (b) Roll offset

Figure 5.10: Schematic of angle offsets during initialisation

5.3.1 Trajectory

To test the filters a MATLAB script similar to the one explained in section 5.1 was used, with the addition to set filter parameters for each individual coordinate and plot the filtered trajectory. Figure 5.11 shows filter applied to all three coordinates with the same parameters. Using the data associated with the analysis illustrated in section 5.2.1 the parameters could be chosen to improve the resulting position data.

The implementation of the filters in the running system still were not able to track the true position closely enough to fly stable, which drew the need to analyse another part of the localisation process.

5.3.2 Distances

The fundamental data the localisation relies on is the information about the distance to each of the six anchors. By plotting the distance to every anchor over time it is evident that the data is compromised (see fig. 5.12(a)).

The TWR process is very prone to bad readings which is reflected by a distance equal to zero in the plot. The large amount of bad readings is depicted in figure 5.12(b) together with the percentage of bad readings per anchor. The zeros in the data are replaced onboard with the last measured value but some outliers remain. These measurements are corrected by assuming the distance might have changed but only by a small amount. If distance changed by more than Δd, the measurement $d_{i,t}$ to anchor i at time t is updated by

$$d_{i,t} = d_{i,t-1} + \rho \cdot (d_{i,t} - d_{i,t-1}), \tag{5.2}$$

where Δd is the threshold which defines when a measurement is corrected, and ρ a proportionality factor that can be tuned.

The corrected distances together with a Kalman-filter on each individual distance are plotted in figure 5.13.

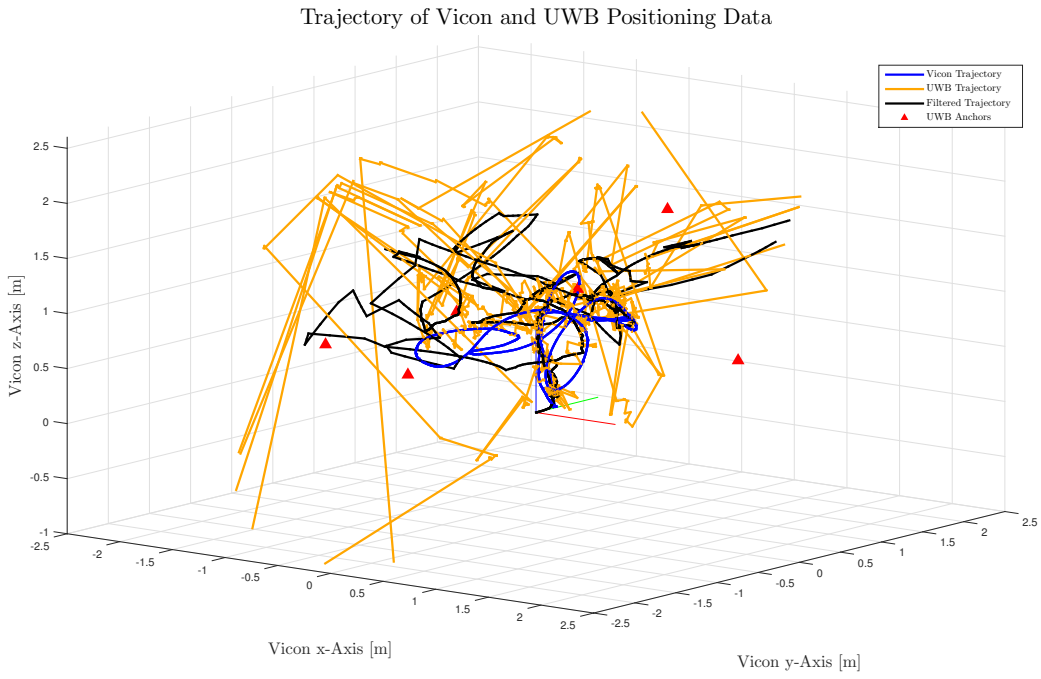
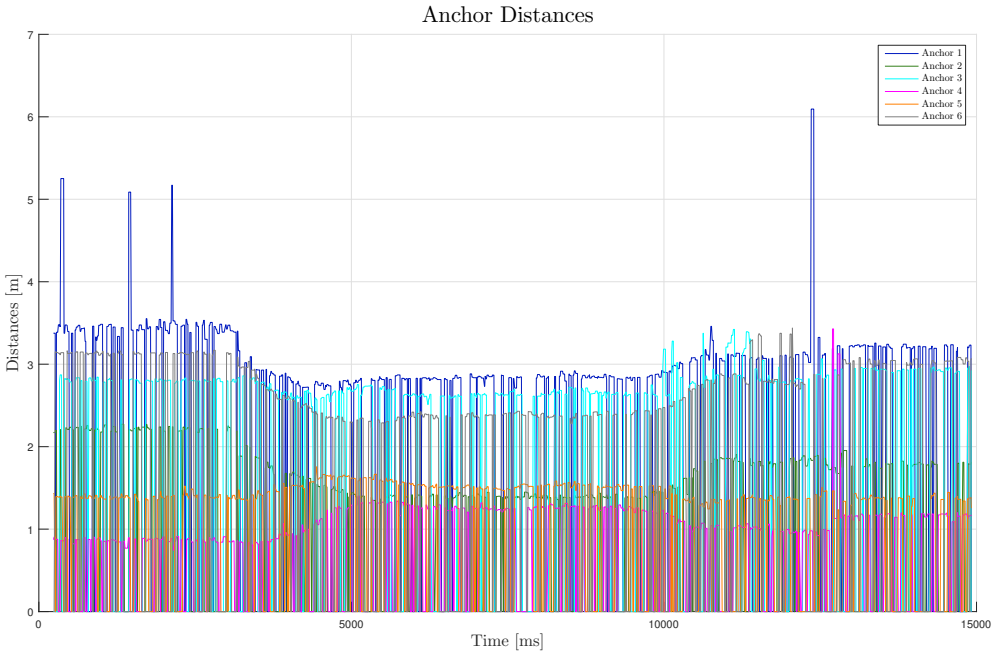
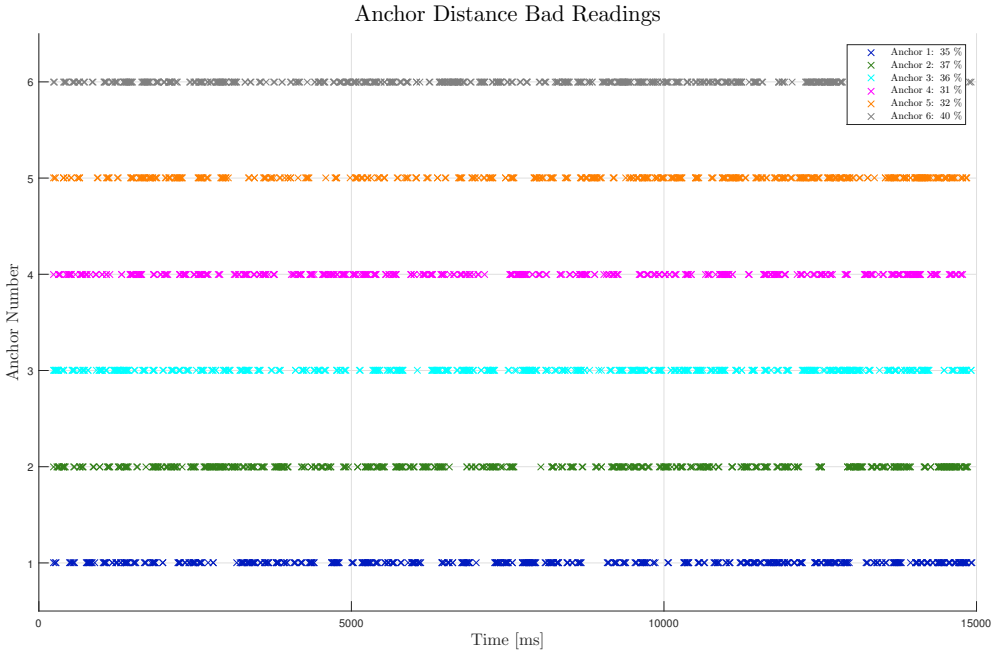


Figure 5.11: Early position recording, filtered

To tune the Kalman-filter on the distances another graph (see fig. 5.14) was considered, depicting the measured and true (Vicon measured) distances. Despite a constant offset on some of the anchors, the overall tracking of the distances is good enough to be used for localisation.



(a) Distances measured using TWR



(b) Bad readings per anchor

Figure 5.12: Uncorrected distance data

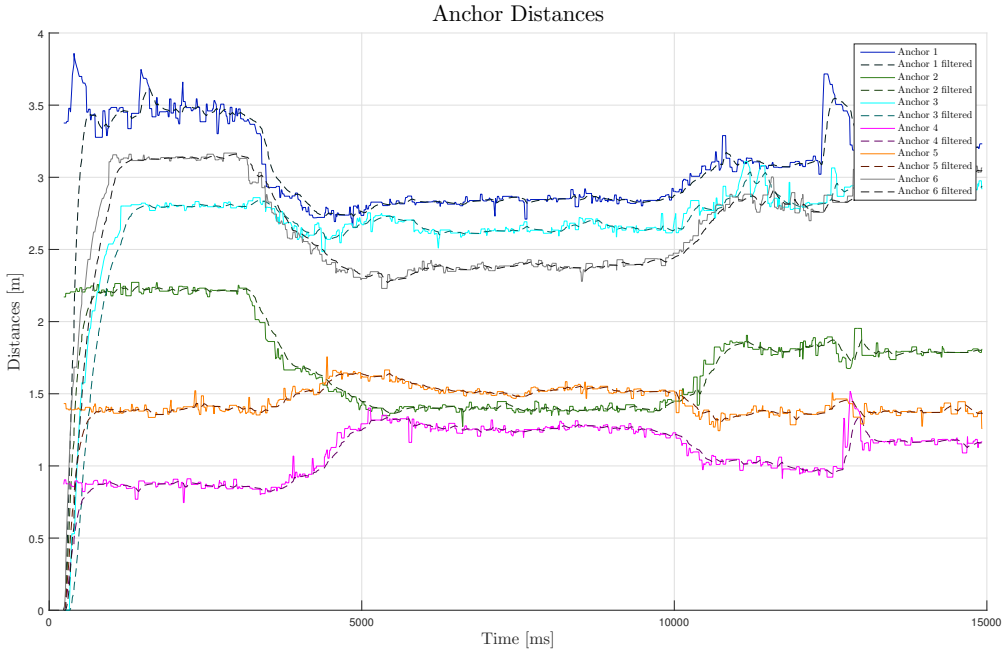


Figure 5.13: Corrected and filtered distances

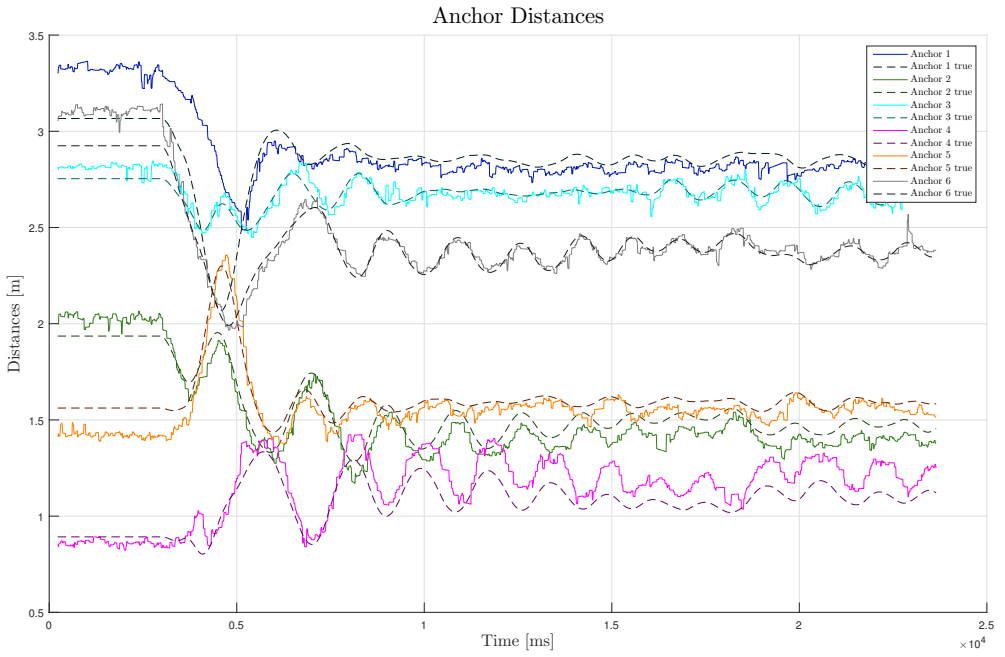


Figure 5.14: Measured versus *true* distances

Chapter 6

Conclusion

The main goal of this group project was to have the quadcopter flying in a controlled manner without the assistance of the Vicon system, which was achieved. Albeit requiring specific setup and environment, the Crazyflie was observed to fly within 10cm of its setpoint while enough power remained in its battery. Reiterating the key challenges posed at the beginning of this paper:

Localisation The Crazyflie was accurately, reliably and quickly located within a 3D space for most scenarios, illustrating a clear improvement in both speed and accuracy over previous work at Ifa [6]. The important z-component was accurately determined with the help of a few additional tricks and filtering.

Attitude While the attitude information ultimately used was already in the source code of the Crazyflie, deeper insight was gained by testing an alternate estimator.

Ease-of-use The use of UWB data now seamlessly works at the click of a button. The setup process of course environment has shortened tremendously, requiring only placement of anchors and a quick execution of the self-localisation algorithm.

6.1 Next Steps

While a lot has been achieved, there are still quite a few improvements to be conducted. The system currently solves the optimization algorithm, determining position from the distances to the anchors, on the student computer. A decided improvement would be to solve this problem onboard the Crazyflie, allowing for more frequent and more accurate position updates. It would also allow this data to be sensibly fused with IMU values, allowing for more precise and accurate filtering. The challenge which remains is reducing the problem's complexity to reduce the runtime onboard the flying microcontroller. Currently, if the optimization problem were solved onboard, the distances would be updated less frequently, resulting in far fewer accurate positions, which results in loss of control of the Crazyflie.
Another problem is that, for a P&S course environment, more than one Crazyflie is expected to fly at the same time. Having more than one Crazyflie asking the anchor nodes for distances results would result in far fewer accurate distances per Crazyflie. To

solve this, a new localization algorithm would have to be implemented, not relying on two-way ranging (TWR), but time of arrival (TOA) or time-difference of arrival (TDOA). These protocols are based on the Crazyflie only listening to anchors sending packets and saving their timestamps, while for TWR, each Crazyflie is permanently asking each anchor for their distance.
Another point of improvement would be the attitude measurements. While the onboard sensors provide a sufficient estimate, a yet unreleased UWB sensor could determine the angle of arrival of the signal, allowing for more precise localization and attitude estimation.
Throughout the work on this project, the position of the anchors was continually changed, but it was observed that for different anchor placements and for different setpoints within these anchors, the position was determined more or less accurately. An ideal anchor placement was, however, not found, only a best so far. It could be possible, however, to numerically or analytically determine the best possible position for these anchors using a computer simulation.
Ultimately, as for most other systems, the filters implemented as well as the controllers used could be further optimized, allowing for more stable flight.

Bibliography

[1] Dwm1000 datasheet. https://www.decawave.com/sites/default/files/dwm1000-datasheet-v1.6.pdf, 2016. Accessed: 2018-10-13.

[2] Dw1000 user manual. https://www.decawave.com/sites/default/files/dw1000_user_manual_2.12.pdf, 2017. Accessed: 2018-10-13.

[3] Bitcraze webpage. https://www.bitcraze.io, 2018. Accessed: 2018-10-13.

[4] Robot operating system (ros). http://www.ros.org/, 2018. Accessed: 2018-10-13.

[5] Vicon. https://www.vicon.com/, 2018. Accessed: 2018-10-13.

[6] M. Binder, L. Bieri, M. Meier, and N. Melchior. Indoor localization using ultra-wide-band.

[7] Rakhesh Singh Kshetrimayum. An introduction to uwb communication systems. *IEEE Potentials*, 28(2):9–13, March-April 2009.

[8] S. Madgwick, A. Harrison, and R. Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *Rehabilitation Robotics. IEEE International Conference. 2011.* IEEE, June-July 2011.

[9] Mark W Mueller, Markus Hehn, and Raffaello D'Andrea. Covariance correction step for kalman filtering with an attitude. *Journal of Guidance, Control, and Dynamics*, pages 1–7, 2016.