CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

F3

Faculty of Electrical Engineering
Department of Control Engineering

Master's Thesis

# Indoor UAV localization using Ultra-wideband system

**Jakub Csanda**
Master programme: Cybernetics and Robotics
Branch of Study: Cybernetics and Robotics

2020
Supervisor: Ing. Milan Rollo, Ph.D.

# MASTER‘S THESIS ASSIGNMENT

## I. Personal and study details

| | | | |
|---|---|---|---|
| Student's name: | **Csanda Jakub** | Personal ID number: | **459920** |
| Faculty / Institute: | **Faculty of Electrical Engineering** | | |
| Department / Institute: | **Department of Control Engineering** | | |
| Study program: | **Cybernetics and Robotics** | | |
| Branch of study: | **Cybernetics and Robotics** | | |

## II. Master’s thesis details

Master’s thesis title in English:

**Indoor UAV localization using Ultra-wideband system**

Master’s thesis title in Czech:

**Interiérová lokalizace bezpilotních prostředků s využitím systému UWB**

Guidelines:

Reliable localization is a crucial ability for UAVs to perform operations in indoor GPS denied environment. Nowadays, an Ultra-Wide Band (UWB) localization is becoming widely spread technology for such a purpose. Extend existing onboard UAV localization systems with UWB to allow operation in indoor environments.

1. Study the state-of-the-art methods of indoor UAV localization and provide their comparison in terms of accuracy and usability.
2. Study commonly used ROS (Robot Operation System) packages for UAV localization and sensor fusion.
3. Design and implement packages necessary to integrate 3rd party UWB localization system into ROS and provide a suitable mechanism to fuse its data with other onboard positioning and attitude sensors.
4. Develop a simulation platform to perform software-in-the-loop experiments and verify all the developed algorithms.
5. Carry out real-world experiments and compare results of such measurements against an independent reliable positioning system.
6. Provide system documentation and user manuals.

Bibliography / sources:

[1] Macoir, Nicola, et al. "Uwb localization with battery-powered wireless backbone for drone-based inventory management." Sensors 19.3 (2019): 467.
[2] Tiemann, Janis, and Christian Wietfeld. "Scalable and precise multi-UAV indoor navigation using TDOA-based UWB localization." 2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN). IEEE, 2017.
[3] Benini, Alessandro, Adriano Mancini, and Sauro Longhi. "An imu/uwb/vision-based extended kalman filter for mini-uav localization in indoor environment using 802.15. 4a wireless sensor network." Journal of Intelligent & Robotic Systems 70.1-4 (2013): 461-476.
[4] Tiemann, Janis, Andrew Ramsey, and Christian Wietfeld. "Enhanced UAV indoor navigation through SLAM-augmented UWB localization." 2018 IEEE International Conference on Communications Workshops (ICC Workshops). IEEE, 2018.
[5] Raja, Asad Khalid, and Zhibo Pang. "High accuracy indoor localization for robot-based fine-grain inspection of smart buildings." 2016 IEEE International Conference on Industrial Technology (ICIT). IEEE, 2016.

Name and workplace of master’s thesis supervisor:

**Ing. Milan Rollo, Ph.D., Artificial Intelligence Center, FEE**

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **29.01.2020** Deadline for master's thesis submission: **14.08.2020**

Assignment valid until:
**by the end of winter semester 2021/2022**

| | | |
|---|---|---|
| Ing. Milan Rollo, Ph.D. | prof. Ing. Michael Šebek, DrSc. | prof. Mgr. Petr Páta, Ph.D. |
| Supervisor’s signature | Head of department’s signature | Dean’s signature |

## III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.

| | |
|---|---|
| Date of assignment receipt | Student’s signature |

## Acknowledgement / Declaration

I would like to thank my supervisor, Ing. Milan Rollo, Ph.D., for a chance to participate in such an interesting project. I would also like to express my sincere gratitude to Ing. Tomáš Meiser and Michal Vatecha for their technical advice and patience.

I hereby declare that I have written this Master's thesis on my own and that I have used only the sources listed in references.

Prague 14. 08. 2020

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

# Abstrakt / Abstract

Počáteční kapitola této práce je věnována teoretickému úvodu do problematiky lokalizace a odhadování stavu bezpilotních prostředků spolu s porovnáním moderních lokalizačních přístupů. Dále je popsána architektura systému složeného z poskytnutého UWB lokalizačního systému a specifikovaného systému pro řízení bezpilotních prostředků. Pro zhodnocení UWB lokalizačního systému byla vyvinuta experimentální platforma. V práci je rovněž poskytnuta diskuze výsledků za použití této platformy. Poté je představena integrace UWB lokalizačního systému a systému pro řízení bezpilotních prostředků. Na konci práce jsou diskutovány experimenty provedené na dronu související s odhadem stavů za pomoci UWB lokalizačního systému.

**Klíčová slova:** dron, UWB, lokalizace, interiérová lokalizace, sensorová fúze, ROS

**Překlad titulu:** Interiérová lokalizace bezpilotních prostředků s využitím systému UWB

In this Master's thesis, the initial chapter is dedicated to the introduction to the localization and state estimation problems concerning the UAV application, along with the comparison of the state-of-the-art localization approaches. The system architecture consisting of the provided UWB localization system and a specified UAV framework is described next. For evaluation of the UWB localization system, an experimental platform was designed and implemented, and experiment results are discussed. Afterward, the integration of the UWB localization system with the UAV framework is presented. Finally, the experiments conducted on the UAV concerning the state estimator employing the UWB localization system are discussed.

**Keywords:** UAV, drone, UWB, localization, indoor, sensor fusion, ROS

# Contents /

# Tables / Figures

# Chapter 1
# Introduction

Many decades have passed since the development of the first aircraft. As technology progressed, more and more sophisticated aircraft were designed. An application of unmanned aerial vehicles (UAVs) in various areas is becoming very popular. UAVs are used for thermal imaging during search and rescue missions in a harsh environment, monitoring areas, product deliveries, or area surveying. However, UAVs can also be deployed in an indoor environment. The use of UAVs is particularly advantageous in a situation where it is inconvenient to send human workers. Examples can be the interior inspection of tanks filled with toxic gasses or building with high ceilings.

In recent years, the application of UAVs in industrial automation is a trendy topic as more and more potent technologies are available, and the cost and weight of necessary hardware decrease. Nowadays, a large one-time investment into the UAVs and essential infrastructure for their deployment can be cost-efficient compared to the workers' wages over a long time horizon. Humans are also prone to errors due to distractions, personal issues, current health situation, etc.

The industrial application introduces additional requirements on the UAV. Among the essential requirements is the accuracy of the UAV localization. The insufficient accuracy of the localization can lead not only to collisions but also to improper mission conducting. The necessary accuracy level varies among applications, but in general, the accuracy requirements are more strict compared to the outdoor applications due to a large number of potential collisions not only with the environment itself but also with human workers. The localization via the ultra-wideband (UWB) radio signals is considered as a promising method. This thesis's primary goal is to analyze the UWB localization system approach and integrate it with an existing UAV framework.

In Chapter 2, a theoretical background of the localization and state estimation problems is provided. Additionally, an introduction to the most commonly used localization approaches is included, and the comparison with the UWB localization approach is drawn. In Chapter 3, the UAV onboard control system architecture is proposed. Chapter 4 is dedicated to the development of the experimental platform used for the evaluation of the UWB localization system, along with the analysis of experiments themselves. Chapter 5 then describes the implementation necessary for the successful integration of the UWB localization system with chosen frameworks. Chapter 6 provides the experimental evaluation of the UWB localization system and its integration with the UAV frameworks in the application. Chapter 7 provides a summary of the thesis output and a few proposals for improving and extending the work.

# Chapter 2
# Theoretical Background

In this chapter, the theory necessary for the understanding of this thesis is discussed. As the goal of this thesis is the indoor UAV localization, the localization problem in robotics and representative localization approaches are discussed first, with an increased focus on the localization via the UWB technology. The rest of the chapter is dedicated to the UAV state estimation, emphasizing sensor fusion, and the Kalman filter approach.

## 2.1 Localization Approaches

In this section, a localization problem in robotics is discussed. A few primary criterions used to evaluate localization systems are described next. Afterward, the taxonomy of localization systems is briefly introduced in terms of the necessary components required for the proper functioning of systems. Then a few most commonly used localization techniques are described.

The rest of this section is dedicated to introducing the variant of the UWB localization system used later in this thesis, as well as a few other localization systems similar to the UWB localization system in terms of the system's architecture. For each localization system, a subsection was written with the following structure. A brief overview of the technology is followed by a more technical description of the type of signal used for localization. Afterward, localization techniques and methods most commonly used by each system are described. Each subsection is concluded with an overview of localization error sources.

Finally, the comparison of the localization systems described in this section is drawn.

### 2.1.1 Localization Problem

Robot localization is one of the fundamental problems in autonomous robotics. It is a process of determining a location of the robot in some coordinate system. If a robot is to determine a correct action while, e.g., moving from point A to point B, it must know its position and attitude.

To fully describe the robot's position and attitude, three or six parameters are necessary, considering a 2D and a 3D space. In a 2D case, the robot's position is described by two values and the robot's attitude by one value, as shown in Figure 2.1.

In a 3D case, three values are needed to describe the robot's position and another three to describe its attitude. Typically, these three parameters are so-called Euler angles roll, pitch, and yaw, representing rotation about axes $x, y$, and $z$, respectively, and can be seen in Figure 2.2. However, for the sake of this thesis, the term *localization* will be used for the process of determining only the position of the robot.

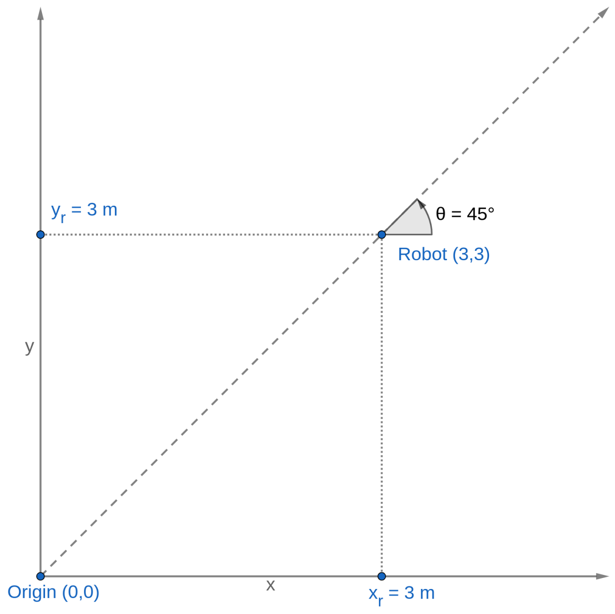


**Figure 2.1.** Example of robot position and attitude in a 2D counterclockwise Cartesian coordinate system.

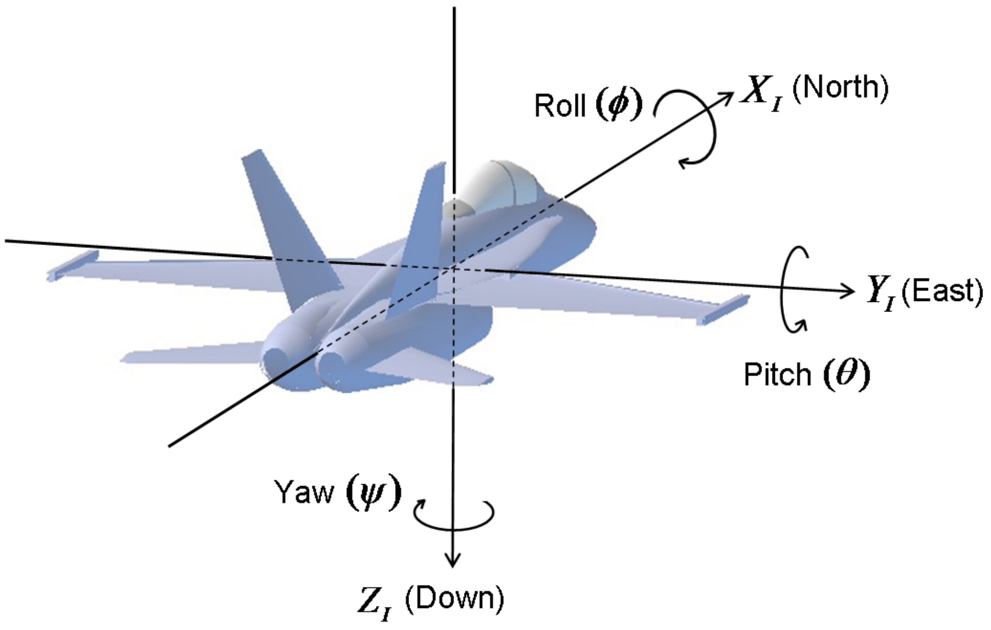


**Figure 2.2.** Euler angles in a fixed 3D coordinate system, adopted from [1].

### 2.1.2 Important Performance Factors In Localization

In localization, several factors are considered to evaluate the quality and usability of individual localization systems. Arguments in this subsection are based on [2].

An *accuracy* is perhaps the most important factor considered while evaluating a localization system. It is defined as how much the estimated position differs from the actual position. The accuracy of the system is often one of the first factors determining whether the system is applicable for a given task (e.g., in a complex indoor environment with narrow corridors and obstacles, the requirement for better accuracy is far more critical than in an open outdoor environment).

A *precision* is often expressed in combination with the accuracy factor. The precision informs about the credibility of the accuracy (or how often the deviation from the actual position is at least smaller than the given accuracy). E.g., a localization system may have a 20 cm accuracy over 95 % (precision) of the time.

A *cost* is, in fact, a group of factors. First, there is a cost of the hardware (transmitters, receivers, and other equipment) and software. Second, the installation cost of the hardware in the environment, provided that a suitable infrastructure does not already cover the operational area. Third, a cost related to the operation of the installed hardware (e.g., power consumption). The system's total cost needs to be correctly examined while evaluating which localization system is the best candidate for the desired application.

A *range* parameter is used to define the area around a static infrastructure element. If an object's true position is inside the area, the corresponding infrastructure element can be used for its localization. This factor is crucial in deciding where to place the static infrastructure elements and the number of these elements necessary for sufficient coverage of the environment.

A *responsiveness* is another crucial measure of a localization system. It is interpreted as the time needed for data processing and calculating the estimated location. It is a vital factor when the localization system is used in real-time applications such as movement control.

A *scalability* is the last factor discussed in this thesis. It indicates whether the system is suitable for the simultaneous localization of a high number of objects. With the rising demand for multi-robotic systems and the coexistence of humans and robots in the same environment, it is crucial to consider if a system can handle the desired number of objects to track while still performing reasonably well.

### 2.1.3 Taxonomy Of Localization Systems

Localization systems can be classified based on several attributes. One of the attributes that can be used to divide localization systems into two main categories is the necessary infrastructure.

The first category consists of systems that can localize the robot without external infrastructure, only with onboard sensors and their interaction with the environment. For a typical application, the localization in a fixed frame is of interest. This approach assumes that the initial position and attitude in this frame are known. The onboard sensors then estimate the evolution of system states from the previous state during a time interval. Some examples are the inertial measurement unit (IMU), camera, or revolution counter. The IMU typically measures linear accelerations and angular velocities and integrates them to obtain the robot's position and attitude. The camera can estimate the distance and angle derivation by comparing two consecutive images. The last example, the revolution counter, can be used by robots equipped with wheels to count the number of wheel revolutions. This approach is relatively cheap, fast, and can be used almost everywhere immediately without any need for deploying the infrastructure. On the other hand, as it depends on the previous estimate, it tends to drift in time, as every introduced error is integrated with each step.

The second category of localization systems necessarily needs an external infrastructure for precise localization. This external infrastructure is usually used to estimate the robot position directly, in contrast with sensors from the first category that obtain the position estimate via propagation through the system model. Individual localization systems place different requirements on the onboard sensors. Typically, the robot needs to have a transmitter or a receiver that communicates with the infrastructure through signals. Some systems based on the processing of the optical signal can localize the robot without communication device, only by exploiting its unique features that are easily distinguishable in the image. Provided that a suitable infrastructure does not already cover the operational area, its development brings additional costs to the localization systems, and outside of their covered area, the localization systems are not usable. On the other hand, they are not subjects to the drift introduced by integrating the error of measurements.

In most UAV applications, using only sensors from one group is not sufficient, as each localization system has its advantages and disadvantages. Typically, a multi-sensor solution is used, using sensors that mutually compensate their disadvantages and combine their advantages. However, a question of how much to trust each sensor arises. Because of that, an approach called *sensor fusion* is used. This approach assumes that multiple sensors estimate or measure the UAV position and attitude, and sensors' outputs are weighted based on the accuracy of the sensor and fused into the estimate of the UAV states. That way, the drift can be corrected by measurements that are not subject to it.

### 2.1.4 Localization Techniques

In this subsection, a few localization techniques widely used for indoor localization via radio signals are introduced.

#### Received Signal Strength Indicator

The received signal strength indicator (RSSI) approach is based on the measurement of the received signal power strength. As the signal propagates through space, its power strength decreases. After a receiver receives the signal, the transmitter-receiver distance can be estimated from Formula (2.1) as

$$d = 10^{\frac{A - \mathrm{RSSI}}{10n}}, \tag{2.1}$$

where

- $d$ is the estimated distance between transmitter and receiver [m],
- $A$ is the RSSI value at a reference distance from the transmitter [-],
- RSSI is the signal RSSI measured at the receiver [-],
- $n$ is an environment-specific constant representing the signal attenuation that typically varies from 2 for outdoors to 4 for indoors [-].

As Formula (2.1) indicates, only a distance from the transmitter is calculated once the receiver receives the signal. The calculated distance is not sufficient for localization in space, as it would place the receiver on the sphere with radius given by $d$. A technique called *trilateration* is used to estimate the receiver's location in a 3D space. This technique requires that the receiver's distance from four transmitters can be estimated, creating four spheres, one around each transmitter. Their intersection point is chosen as the estimated location of the receiver. The 2D trilateration is illustrated in Figure 2.3. Sometimes, however, the spheres can have zero, or more than one intersections, due to inaccuracies of the measurements, and some algorithm must be employed to determine the most probable estimate. If possible, more transmitters can be used to increase the accuracy of this method.

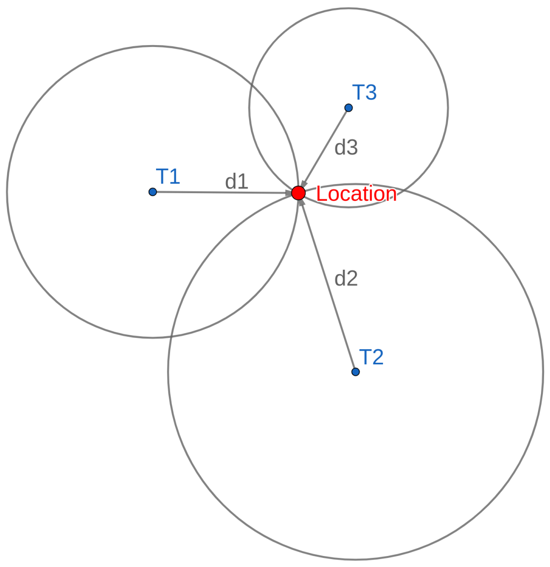


**Figure 2.3.** Illustration of the trilateration method with three transmitters labeled as T1, T2, and T3, and distances between receiver and each transmitter $d1, d2$, and $d3$, respectively.

While the RSSI-based localization systems are typically simple and cheap, they suffer from poor accuracy due to severe fluctuation of the signal strength due to transmission through walls and other obstacles, as well as due to the multipath phenomena.

### Angle Of Arrival

The angle of arrival (AoA) method uses multiple antennas as receivers to estimate the angle of the received signal. The angle is estimated by calculating the time difference at each antenna. The location is then obtained by employing the *triangulation* method. A line with direction defined by this angle relative to the antennas is drawn. The *triangulation* in the 3D space requires only three measurements (compared to *trilateration*'s four measurements), which is considered the main advantage of the AoA method. The disadvantage of this method is its complex hardware, as well as careful calibration. The accuracy is significantly reduced with an increase in the receiver's distance from the transmitter, as the error in angle is projected hugely into the location estimation. Because of that, this method is also susceptible to errors caused by multipath or when no line of sight is available and thus is rarely suitable for indoor applications.

### Time Of Flight

Time of flight (ToF), sometimes called the time of arrival (ToA), calculates the distance between the receiver and the transmitter by multiplying the ToF by the signal's propagation speed (typically the speed of light $c \approx 3 \times 10^8$ m/s). Very often, a timestamp is transmitted with the signal and is used to calculate the ToF.

A strict synchronization requirement is necessary between transmitters and receivers to determine a precise ToF. An alternative approach called two-way ranging (TWR) can be used to eliminate the time offset between two devices. As the name suggests, the devices exchange messages in both ways. There exist several variants of the TWR algorithms, but in general, the time offset is eliminated by comparing the timestamps of the request and response messages.

The accuracy of the ToF distance estimation also depends on the sampling rate and the signal bandwidth. If a sample rate is low compared to the signal velocity, the accuracy is lower as the signal may arrive between the consecutive samples, and the ToF is miscalculated. The signal bandwidth determines the robustness of the ToF accuracy in the multipath environment. The *trilateration* method is used for determining the precise location of the receiver.

#### Time Difference Of Arrival

Unlike the ToF method, the time difference of arrival (TDoA) method requires strict synchronization only on the transmitter side, which can be achieved more easily. For each unique pair of transmitters, a measured time difference between the signals' arrival is multiplied by the signal velocity to obtain a distance difference of the transmitter pair relative to the receiver. This distance difference is used to define a hyperboloid on which the receiver is located. In a 3D space, at least three TDoA measurements (corresponding to the system of at least three transmitters) are required to calculate the receiver's exact location as the intersection of defined hyperboloids.

Like the ToF method, the accuracy is influenced by the sampling rate and the signal bandwidth, as well as precise time synchronization mentioned above.

### 2.1.5 Global Navigation Satellite System

This subsection introduces a global navigation satellite system (GNSS), which is often used for outdoor localization of a robot. This chapter is based on [3]. Although it is not used for indoor localization, it is included in this thesis as a localization system that is arguably the most similar to the UWB localization system.

A satellite system is a system that uses satellites orbiting the Earth to provide global coordinate system localization for objects on and above Earth's surface. Each satellite transmits a radio signal along a line of sight. Each object that is to be located must have a receiver that can track this signal.

If a satellite system can provide global coverage, it is referred to as a GNSS. The first GNSS ever made is the United States' Global Positioning System (GPS). Nowadays, there exist several GNSS. Besides GPS, there is the Russian GLONASS, the Chinese BeiDou system, and the European Galileo system. There is also the Japanese QZSS and the Indian NavIC navigation satellite systems, but these provide only a regional coverage and are not considered a GNSS.

Each GNSS has a satellite constellation consisting of typically at least 24 satellites. These satellites are arranged in such a fashion that from most areas on the

Earth's surface, at least four of them are visible at any given time. Otherwise, a precise localization would not be possible due to the ToF localization technique used (see Section 2.1.4).

Nowadays, receivers are developed to be able to receive signals from multiple GNSS. This ability is beneficial in areas with limited sky visibility such as cities or forests, where some of the satellites responsible for coverage of such area is hidden behind an obstacle, but few satellites from different GNSS are visible. That way, each GNSS separately would fail to provide reliable localization, but the receiver can combine the information from both of them and localize itself based on this information.

### Signals

GNNS signals are radio signals that include ranging signals and navigation messages. Each GNSS has defined several frequencies that generally differ between individual GNSS, although some overlays are present as well. As each satellite in the same GNSS constellation transmits on the same frequencies as the others, to correctly identify the signal's source, the code-division multiple access (CDMA)[1] spread spectrum technology is used. The CDMA allows multiple transmitters to send information at the same time over the same communication channel. Each signal is modulated by a pseudorandom code unique to each satellite to distinguish between transmitters, expressing the need for a receiver to know each satellite's pseudorandom code to correlate with the CDMA channel to extract the desired signal.

GNSS signals transmit on an L-band frequency range. L-band is a range of frequencies in the radio spectrum from 1 to 2 GHz. This range is chosen mainly due to its resistance to unwanted natural effects. That means that these waves are not very influenced by rain, snow, clouds, fog, or vegetation. However, they cannot penetrate the dense environment, such as heavy forest canopies and concrete walls [4]. Also, higher frequencies would require more complex antennas, increasing the cost of application.

Since the signal travels on the path that goes through a non-vacuum environment, its speed is slowed. One of these delays happens in the ionosphere and is described by Equation (2.2).

$$v = \frac{40.3}{cf^2} T_{EC}, \qquad (2.2)[5]$$

where

- $v$ is the ionospheric delay [m/s],
- $c$ is the speed of light in vacuum [m/s],
- $f$ is the frequency of the signal [Hz],
- $T_{EC}$ is the number of free electrons [m$^{-2}$].

Equation (2.2) shows that if two signals with different frequencies travel along the same path simultaneously ($T_{EC}$ vary with time and position), the signal with lower frequency experiences higher ionospheric delay than the signal with higher frequency,

[1] The GLONASS also uses the frequency-division multiple access (FDMA) in combination with the CDMA.

making the transmission on more than one frequency advantageous.

### Localization

In order to provide accurate localization, there must be a line of sight between the GNSS receiver and at least four satellites. In an ideal case, three satellites would be sufficient. However, most receiver clocks' precision is around 5 ppm [6], meaning that, on average, they drift about one second every two days.

As discussed further in this subsection, even the slightest time difference is responsible for a significant error in the resulting computation of pseudorange[1]. This error can lead to a situation where localization spheres (spheres with a diameter of calculated pseudorange) do not intersect at a single point. Adding the fourth measurement and using a trilateration technique, receiver, which is aware of the fact that the source of the error is most likely its clock, is programmed to advance/delay its clock until pseudoranges converge to a single point. This way, the position can be estimated. Additionally, the receiver can synchronize its clock with satellites, eliminating the clock drift.

### Error Sources

In this subsection, errors that influence the accuracy of the GNSS are described. The term error is defined as the Euclidean distance between true and estimated positions. Equation (2.3) defines the Euclidean distance between points A and B in a 3D cartesian coordinate system.

$$d_{AB} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2}, \tag{2.3}$$

where

- $x_i$ is the $x$ coordinate of point $i$ [m],
- $y_i$ is the $y$ coordinate of point $i$ [m],
- $z_i$ is the $z$ coordinate of point $i$ [m],
- $d_{ij}$ is the Euclidean distance between points $i$ and $j$ [m].

These errors can be described by an ellipse (in determining a 2D position) or an ellipsoid (in a 3D case). The magnitude of the maximum error (maximum Euclidean distance between exact and estimated positions) is then represented by the length of the ellipsoid's axes. Very often, the ellipsoid can be approximated by the sphere, since the estimation's error is not far from being unbiased by one of the coordinate axes. That way, the maximum error bound is represented by the sphere's radius, and the actual error is always contained inside a sphere[2]. All introduced errors are also assumed to be independent, and the total error is calculated by summing all individual errors.

[1] Difference between pseudorange and range is that pseudorange is influenced by many physical effects.
[2] E.g., an error in accuracy up to 1 m means that the estimated position is somewhere inside a sphere with a radius of one meter centered at the actual position.

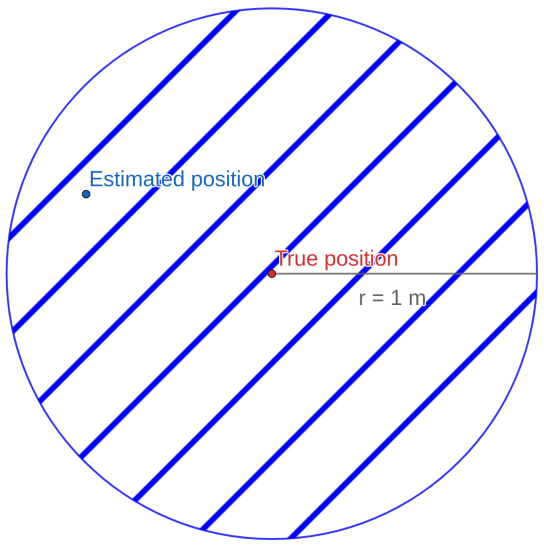


**Figure 2.4.** Example of bound representing an error up to 1 m and possible estimated position.

Pseudoranges are calculated from ranging codes to determine the distance between the receiver and transmitter. Since the pseudorange calculation is based on the ToF of the satellite signal, all inaccuracies and errors must be identified and corrected. The most critical error sources are ionospheric delay, tropospheric delay, multipath signal propagation (further referenced only as multipath), satellite clock drift, orbit error, and receiver noise.

The source of the ionospheric delay is the ionized part of Earth's atmosphere, ranging from 70 to 1,000 km. In the ionosphere, ultraviolet rays from the sun ionize gas molecules, consequently releasing free electrons. These free electrons influence electromagnetic wave propagation, and they are the source of the delay (causing an error in accuracy up to 5 m from real position). Fortunately, as can be seen in Equation (2.2), this delay is frequency-dependent. The receiver, if able, can virtually eliminate the delay by comparing both L1 and L2 signals, since both signals are delayed differently. If the receiver can not track two frequencies, the ionospheric model can be used to predict the delay, but its accuracy is not nearly as accurate as of the comparison of signals.

Another layer of Earth's atmosphere, troposphere, located up to 20 km above the surface, is responsible for another delay. This delay is a function of local temperature, pressure, and relative humidity. Furthermore, compared to the ionospheric delay, it does not depend on the frequency, making it impossible to eliminate the delay by using L1 and L2 signals. On the other hand, tropospheric models are much more accurate and stable than ionospheric models, allowing them to predict the delay somewhat correctly.

Each satellite has an atomic clock on board. These are very accurate, but nevertheless, drift a small amount, and since the signal travels at the speed of light, 1 ns drift of clock is responsible for the error of about 30 cm. Satellites can predict the offset from the ground-based master clock (that are more accurate than the clock onboard satellites), but even after this prediction and the fact that the satellite clock is periodically synchronized with the master clock, the error in accuracy can still occur.

Even though each satellite flies on a very accurately determined trajectory, they may vary a little, and similarly to the case with clock drift, every small change can result in a significant error. If this deviation from trajectory is detected, the GNSS ground control system sends correction to the satellite, but until then, the inaccuracy in distance measurement can be observed.

All of the error sources mentioned are very similar within a local area. Because of this fact, they can be highly compensated by differential (DGNSS) and real-time kinematics (RTK) systems briefly introduced at the end of this subsection.

The last significant source of error is a phenomenon called multipath. In short, multipath means that the signal can travel from satellite to receiver along multiple paths. Apart from the apparent direct path, a signal can be refracted or reflected by the environment. Since this phenomenon may vary significantly within a local area (e.g., urban area with many structures), RTK GNSS does not compensate for it. One of the most straightforward solutions is to consider only the first arriving signal.

#### DGNSS and RTK GNSS

Both DGNSS and RTK GNSS are significant enhancements to the classic GNSS in terms of accuracy. They can compensate for several critical errors that standard GNSS can have a hard time dealing with. The receiver that is to be localized is often referenced to as a rover.

The underlying idea is to place a base station on some fixed, precisely determined location, preferably in location minimizing undesired effects such as multipath. Next, pseudoranges from satellites' signals are used to calculate the location of the base station. The base station then compares the precisely determined position with a calculated position and calculates the correction data sent to the rover through a data link, typically through an ultra-high frequency (UHF) band.

Both DGNSS and RTK GNSS accuracy are highly dependent on the precision of the base station placement and the distance between the rover and base station (works very well up to tens of kilometers). The reason behind the distance dependency is that most of the compensated errors by DGNSS and RTK GNSS are similar within a local area, but they may vary significantly with increasing distance of rover from the base station.

The difference between DGNSS and RTK GNSS is that DGNSS uses a code-based positioning, while RTK GNSS uses carrier-based ranging that can provide ranges that are orders of magnitude more precise than code-based positioning (orders of centimeters).

### 2.1.6 Ultra-wideband Localization System

In this subsection, the ultra-wideband (UWB) technology is introduced. While the UWB technology originated some decades ago, the use of this technology was, for a long time, restricted for military purposes only. In 2002, the UWB was allowed for public use. However, some limitations were defined, such as the allowed frequency bandwidth and the maximum allowed power level, mainly due to a large number of existing narrowband technologies with which the UWB could interfere. Arguments in this subsection are based on [7], [8], and [9].

#### Signal

The UWB signal is defined as a radio signal that has an absolute bandwidth larger than 500 MHz or a fractional bandwidth larger than 20 %. The unlicensed use of UWB technology is authorized in the frequency range between 3.1 to 10.6 GHz.

The absolute bandwidth can be calculated as depicted in Formula (2.4), while the relative bandwidth calculation is defined in Formula (2.5).

$$B_{\text{abs}} = f_H - f_L, \tag{2.4}$$

where

- $B_{\text{abs}}$ is the absolute bandwidth of the signal [Hz],
- $f_H$ is the upper frequency of the $-10$ dB emission point [Hz],
- $f_L$ is the lower frequency of the $-10$ dB emission point [Hz].

$$B_{\text{rel}} = \frac{2(f_H - f_L)}{f_H + f_L}, \tag{2.5}$$

where

- $B_{\text{rel}}$ is the relative bandwidth of the signal [-].

The UWB signal waveform is characterized by pulses with a low duty cycle and a very short duration, typically no longer than a few nanoseconds. For each pulse, a time window is allocated, and the information is determined by the pulse position in the time window (or *time modulation*) and its orientation. These properties define some compelling advantages of the UWB system.

The fact that it has a low duty cycle results in relatively low power consumption, making the UWB system operation somewhat cheap.

As the length of each pulse is small, the possibility of overlapping the original pulse in case of signal reflections is reduced, making it more robust against the multipath problem, provided that a clear line-of-sight (LOS) exists between transmitter and receiver.

The wide bandwidth allows the UWB signal to penetrate through some[1] obstacles as it consists of both low and high frequencies.

The power spectral density (PSD) of the signal, which measures the signal's power compared to its frequency bandwidth, is very low for the UWB systems as the UWB signal has low power and wide bandwidth. This property is critical as the UWB signal, by its nature, shares a spectrum with some narrowband communication systems, such as WiFi. However, because its PSD is very low, the UWB can coexist with such systems, intervening basically as environmental noise. On the other hand, the low PSD makes the UWB communication more or less immune to interception from narrowband communication systems.

Although the average signal power of the UWB systems is considered very low, the UWB systems can transmit at high data rates without error. This fact can be seen from the Shannon-Hartley theorem, which can be seen in Equation (2.6).

$$C = B\log_2(1 + \frac{S}{N}), \tag{2.6}$$

where

- $C$ is the channel maximum capacity [bit/s],
- $B$ is the signal bandwidth [Hz],
- $S$ is the average signal power [W],
- $N$ is the average noise power [W].

Equation (2.6) defines the maximum number of bits that can be transmitted through a channel with defined bandwidth and a signal-to-noise ratio. It can be observed that the maximum capacity increases faster with increasing bandwidth rather than signal-to-noise ratio.

The UWB systems are typically able to achieve high *range resolution*. The *range resolution* can be defined as the ability of the system to distinguish two separate points in space based on their distance. The *range resolution* value can be approximated by Formula (2.7).

$$r \approx \frac{v}{2B}, \tag{2.7}$$

where

- $r$ is the achievable *range resolution* [m],
- $v$ is the velocity of the signal [m/s].

It can be seen that the high bandwidth of the UWB localization systems results in better *range resolution*.

[1] Metals and liquids are usually considered a problematic medium for UWB signals.

#### Localization

The UWB localization system is structurally very similar to the GNSS localization system.

The first similarity is in the infrastructure used. While the GNSS uses satellites with a known location, the UWB uses anchors with a precisely determined static location. The object that is localized via these anchors is called a tag. However, as the UWB localization system is short-range, it provides coverage of just orders of decades of meters.

The second similarity is in the localization technique. The UWB can use the ToF technique that is used by the GNSS. The UWB system, however, can also use the TDoA technique. Based on the technique used, the minimum number of anchors necessary is either three (for ToF) or four (TDoA).

#### Error Sources

Like the GNSS localization, the UWB localization calculates the tag's location based on the distance estimate from each anchor.

As the UWB is used only for local coverage of a small space, the errors caused by delays that influence the GNSS signals introduced by the atmosphere are not an issue in the UWB localization.

As for a multipath, the situation is a little more complicated. The UWB signal is typically more robust to multipath than the GNSS due to its short pulses. Unlike the GNSS, it can also penetrate many types of obstacles, meaning that it can localize, e.g., objects hidden behind the wall. Nevertheless, the penetration of material can slow down the propagation of the signal that is difficult to anticipate. In a non-line-of-sight (NLOS) scenario between anchors and tag, the delay caused by obstacles can severely reduce the accuracy of the distance estimation. This issue can be solved (at least partially) in the same fashion as in the GNSS localization, and that is adding more anchors into the system's infrastructure.

The UWB localization system also shares the requirement for the precise time synchronization necessary for correct distance estimation. However, equipping anchors with the same atomic clocks as satellites is typically impossible due to their high cost. Because of that, one anchor is usually selected and used as a *master* node. This node is used to transmit a synchronization message to other anchors. If the ToF technique is used, the synchronization message is sent to all tags as well. The NLOS scenario, however, can also influence this synchronization communication.

#### GNSS and UWB Precision Comparison

Based on Sections 2.1.5 and 2.1.6, it can be observed that the UWB localization system shares some error sources with the GNSS. However, the GNSS is subject to errors due to the atmospheric influence on the transmitted signal. Because of that, the accuracy of the UWB localization should be better. The accuracy of the GNSS localization system available for public use is usually considered around one meter, while the UWB localization is considered to deliver a sub-meter level of positioning.

However, the RTK GNSS can provide localization accuracy around one centimeter. This approach is not suitable for the UWB localization as it is already developed to provide local localization.

The GNSS localization is currently used widely and successfully in UAV navigation, even without the RTK improvements. However, it can be used only outdoors with a clear view of the sky. Besides, the requirements for indoor positioning accuracy are typically more strict than outdoors. Because of that, considering the similarities between both the GNSS and the UWB systems, the UWB localization system should be able to substitute the GNSS indoors. There are already several solutions (e.g., [10], [11]) using the UWB systems that can achieve orders of decades of centimeters accuracy.

### 2.1.7 WLAN Localization System

This subsection introduces the WLAN localization systems. Due to the increased coverage of most regions in the world with the WLAN signal, the approach using these signals for localization purposes is an appealing one, as the infrastructure needed is usually already available in the desired area. Therefore, these systems are often used as supplementary systems to a GNSS localization in places where GNSS localization is unreliable, such as an indoor environment. However, it should be used only where the localization accuracy requirements are not strict, as the accuracy of these systems is rarely better than one meter. The WLAN localization system reach is typically 50-100 m.

#### Signal

Similarly to the UWB localization system, the WLAN localization system uses electromagnetic waves of high frequencies. However, the WLAN systems typically use narrow bandwidth of the frequency spectrum. Due to this fact, the non-negligible interference with other communication channels using an overlapping frequency band can occur.

The most common WLAN signals can be separated, based on their frequency bands, in two categories. The first category uses the frequency band between 2.4 and 2.5 GHz, and the second category uses the 5 to 6 GHz frequency band. The 2.4 GHz frequency band is generally divided into 11 channels, each having a fixed frequency bandwidth of 22 MHz. However, only three of these channels are not overlapping. The second category is much more varied, offering up to 45 frequency channels with a 20 MHz frequency bandwidth, while 24 of these channels are not overlapping.

#### Localization

The WLAN localization typically uses an already installed infrastructure for wireless communication. Since the information about the received signal strength (RSS) is easily extractable from such communications, the localization approach using WLAN signals is usually based on the RSSI technique. As described in Section 2.1.4, this technique is cheap to use and can be used even in the NLOS scenarios. On the other hand, it is heavily influenced by the quality of the environment's signal propagation model described by Equation (2.1).

This model alone is usually insufficient for localization, as it does not take into account the thickness of the walls on the path. Due to that, a more complicated model that is necessary can include this information into distance estimation. Another approach called fingerprinting is often used instead. Fingerprinting is an empirical technique in which several calibration RSS measurements[1] are conducted at different locations throughout the area and are stored along with their ground truth, creating a so-called radio map. The localization is then done by comparing the object's RSS to the ones stored in the map and calculating the weighted distance from calibration measurements.

#### Error Sources

Unfortunately, the WLAN localization accuracy is heavily influenced by the incorrect modeling of the environment. On the other hand, if the fingerprinting is used instead, the RSS fluctuation due to changes in the environment[2] results in the calibration measurement inaccuracies.

### 2.1.8 Optical Localization System

The optical localization system is a localization system relying on the processing of light rays. The most typical optical sensor is a camera that is further specified based on the application. In this subsection, optical systems using multiple static cameras for the localization of a moving object are described. Used arguments are based on the [12].

#### Signal

The optical systems use very high-frequency electromagnetic waves. Typically, either visible light or infrared light signals are used for localization purposes. Visible light is defined as the light with a wavelength between 380 nm and 740 nm, which corresponds to the frequency between 790 THz and 405 THz, respectively. The infrared light spectrum lies between the visible light and the radio spectrum, ranging from 740 nm to a 1 mm, with the corresponding frequency range from 405 THz to 0.3 THz. It can be seen that the signal's wavelength is rather low, which makes it impossible to penetrate most of the obstacles.

#### Localization

The optical system utilizing several static cameras for real-time localization of a moving robot typically uses an illuminated object located on the top of the robot to increase the robustness of the localization algorithm. Additionally, the object can be designed in such a fashion that its attitude can also be determined.

[1] With respect to each access point.
[2] E.g., moving objects, closing doors, changing the device orientation.

The optical localization algorithms employ the AoA technique mentioned in Section 2.1.4. If the image captured by the camera contains the target, its pixel position in the image is determined. By including information about the distance between the image and the target, the 3D position can be calculated. The distance itself cannot be determined from a single image alone, and an additional measure is necessary for its determination. Typically, either an additional sensor that measures the distance is used or multiple images taken from different positions, including the target, are compared to estimate the scale factor.

#### Error Sources

As mentioned at the beginning of this subsection, the light rays can not penetrate most obstacles. Because of that, the NLOS issue makes the localization impossible. Therefore, in an environment with several obstacles, more cameras are necessary to provide sufficient coverage of the space. It is also necessary to consider that, unlike the UWB, WLAN, or ultrasound anchors, the static cameras are not omnidirectional, meaning that their field-of-view is limited.

Another issue with optical signals is environmental noise. This issue is mostly considered while working with the visible light under either very low or a very high illumination.

The localization algorithm itself is highly dependent on the camera parameters calibration and its pixel resolution. The higher the pixel resolution, the less error is introduced. On the other hand, the image processing computational cost scales with the increased number of pixels.

### 2.1.9 Ultrasound Localization System

In this subsection, the ultrasound localization system is briefly introduced.

#### Signal

In contrast with all the systems introduced so far, the ultrasound system does not use electromagnetic waves for localization. Instead, it uses mechanical waves, or more precisely, the sound waves. The main difference between electromagnetic and mechanical waves is that mechanical waves require medium to transport their energy from one point to another.

The ultrasound signal is defined as a sound signal with a frequency higher than 20 kHz, making the humans unable to hear the communication with their ears. The medium through which the ultrasound signal travels dramatically influences some of its attributes.

First, the signal power is attenuated by the medium. In an indoor localization, where the air is used as the medium, the maximum operational range of the ultrasound localization system is reportedly small, around 10 m [13].

Second, the signal velocity is also very dependent on the medium. In the airborne conditions, the velocity can be calculated by Formula (2.8).

$$v_{US} = (331.3 + 0.606T), \tag{2.8}$$

where

- $v_{US}$ is the ultrasound signal velocity [m/s],
- $T$ is the air temperature [°C].

Formula (2.8) indicates two properties of the ultrasound signal. First, the velocity of the ultrasound signal is drastically lower than the velocity of electromagnetic waves. This fact means that the available range resolution in Formula (2.7) can be great, as it is improved by reducing the signal speed. Second, unlike the electromagnetic signal, the ultrasound signal is considerably influenced by the air temperature.

#### Localization

Ultrasound localization is usually realized while using several static nodes. The object that is to be localized uses a mobile tag, similar to the case of the UWB localization.

The ultrasound localization typically relies on the TDoA technique. However, compared to the UWB localization, slightly different requirements are imposed due to the ultrasound signal's mechanical nature.

#### Error Sources

The ultrasound localization requirement differs from the UWB localization ones. As the ultrasound velocity is much lower, the synchronization requirement is not as strict. E.g., while the UWB time synchronization error of 1 ns can cause up to 30 cm of distance estimation error, the same distance estimation error corresponds to the time synchronization error of about 1 ms.

As can be observed from Formula (2.8), the ultrasound velocity is dependent on the air temperature. As the TDoA localization technique uses the velocity to estimate the distance, it is clear that the best knowledge of the temperature along the path is required. Luckily, the temperature gradient indoors is typically much lower than outside, reducing the error compared to the outdoors scenario. The ultrasound nodes are typically equipped with sensors able to measure the temperature to compensate for this kind of error.

As the ultrasound wave can not penetrate most of the materials in an indoor environment, the NLOS issue, along with multipath propagation, remains a challenge.

### 2.1.10 Comparison Of Introduced Systems

In this subsection, all indoor localization systems introduced above are compared based on the accuracy, cost, and range. Of course, the attributes' values might differ from one implementation of the localization system to another. For this thesis's sake, the value assigned to each system is considered to be a common value among most realizations.

#### UWB Localization

The UWB localization *accuracy* is typically considered between 30 cm and 50 cm for a non-lab environment [13], [14]. The *cost* of the infrastructure setup is relatively high, but the operational *cost* of an already installed localization system is considered cheap due to the UWB signal power restrictions. The range of the UWB is typically comparable with the range of the WLAN localization in a full LOS environment.

#### WLAN Localization

As mentioned above, the WLAN localization system typically has an already installed infrastructure. Therefore, the *cost* of the system is low compared to the other localization systems. The WLAN technology signal typically offers the same reliability *range* as the UWB technology. However, due to the RSSI localization technique used, the *accuracy* rarely achieves a sub-meter level and is typically considered to be $3 - 5$ m [13], [15].

#### Optical Localization

The best optical localization can achieve *accuracy* in the order of millimeters. However, the infrastructure *cost* is very high, as for coverage of a larger space with multiple obstacles, the need for the number of cameras rises. To achieve the best results, the resolution of cameras is typically high, increasing their *cost*. Additionally, the processing of the images to allow a real-time localization requires much computational power, thus increasing the *cost* parameter even more. The *range* of the localization is highly dependent on the sensor size and camera resolution.

#### Ultrasound Localization

The ultrasound localization offers *accuracy* in the order of centimeters at a relatively low *cost*. The disadvantage of the ultrasound localization is the signal's *range*, which is smaller than the *range* of the UWB or WLAN localization.

**Summary**

| Technology | Accuracy | Cost | Range |
|---|---|---|---|
| UWB | $< 30$ cm | Medium | Medium |
| WLAN | $3-5$ m | Very low | Medium |
| Optical | $< 5$ mm | High | Camera dependent |
| Ultrasound | $< 5$ cm | Low | Low |

**Table 2.1.** Comparison of the indoor localization systems.

To conclude, the best possible *accuracy* seems to be possible with the use of high-cost cameras, along with a high-cost server that can process the images sufficiently fast. On the other hand, a WLAN localization system is very cheap but does not provide an *accuracy* level suitable for the UAV localization problem. The ultrasound localization system is also a relatively cheap technology with excellent performance in terms of *accuracy*, but its small range limits its use for small environments. The UWB localization seems to provide a sufficient level of *accuracy* on an acceptable *range* level. The cost of the infrastructure is slightly higher, but the performance/cost ratio favors the UWB localization compared to the expensive camera solution.

## 2.2 State Estimation

The output of the localization systems is often not sufficient on its own, whether it is because of the large noise[1] projected onto the localization accuracy, or because additional information about the localized object is necessary that is not available from the localization systems. For this reason, the control theory is applied, mainly the concept of state-space modeling. The system states can be loosely defined as the variables that provide complete information about the model at any given time instance. As mentioned above, the localization systems often do not bring the necessary amount of information, which is projected into the fact that some states can not be measured. Moreover, the states that are measured are subjects to a noise. Due to that, a software solution is necessary to estimate the actual system state from direct measurements and a system model.

In this section, the state estimation process is introduced. At the beginning of the section, a state-space representation of the actual physical system's mathematical model is defined, emphasizing the discrete-time, linear, and time-invariant (LTI) systems. Next, the need for sensor fusion is discussed. Afterward, the linear Kalman filter algorithm for sensor fusion and state estimation is introduced, along with its non-linear variants. Finally, the particle filter is briefly introduced as an alternative to the Kalman filter. Arguments in this chapter are based on [16].

[1] Noise can be defined as any unpredictable modification of a signal.

### 2.2.1 State-space Representation

A physical system is in control engineering often modeled using a state-space representation of the mathematical model. This description is defined as the set of state, input, and output variables that are related either by the first-order differential (in *continuous-time* domain) or difference equations (in *discrete-time* domain). In this subsection, a discrete-time state-space representation of linear and time-invariant systems is defined.

#### Discrete-time deterministic LTI system

The following equations describe a deterministic discrete linear system with $n$ states, $p$ inputs, and $q$ outputs:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k), \\ \boldsymbol{x}(0) &= \boldsymbol{x}_0, \end{aligned} \tag{2.9}$$

where

- $\boldsymbol{x}$ is the state vector of dimension $\mathbb{R}^{n\times 1}$,
- $\boldsymbol{u}$ is the input vector of dimension $\mathbb{R}^{p\times 1}$,
- $\boldsymbol{y}$ is the output vector of dimension $\mathbb{R}^{q\times 1}$,
- $\boldsymbol{A}$ is the state transition matrix of dimension $\mathbb{R}^{n\times n}$,
- $\boldsymbol{B}$ is the input matrix of dimension $\mathbb{R}^{n\times p}$,
- $\boldsymbol{C}$ is the output matrix of dimension $\mathbb{R}^{q\times n}$,
- $\boldsymbol{D}$ is the feedthrough matrix of dimension $\mathbb{R}^{q\times p}$,
- $k$ is the time sample,
- $\boldsymbol{x}_0$ is the initial state vector of dimension $\mathbb{R}^{n\times 1}$.

A defining characteristic of a deterministic system is the possibility of precise reconstruction of the state development in time based on the initial state $\boldsymbol{x_0}$, known input vector $\boldsymbol{u}(k)$, and observed output vector $\boldsymbol{y}(k)$ at every time sample $k$, provided that the system is *observable*. In general, the output of the system does not necessarily provide enough information for this reconstruction as some states are not projected on the output. This issue is solved by employing a linear state observer, a parallel *observable* system that is designed in such a way that the divergence between its output and the system output converges to zero as time goes to infinity.

#### Discrete-time stochastic LTI system

In a real-world application, however, the system's state development, as well as its observation, are subjects of noise, and thus they can not be predicted precisely. Because of that, it is necessary to describe these signals as random processes, and the resulting LTI system is described by the following equations:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{Ax}(k) + \boldsymbol{Bu}(k) + \boldsymbol{w}(k), \\ \boldsymbol{y}(k) &= \boldsymbol{Cx}(k) + \boldsymbol{Du}(k) + \boldsymbol{e}(k), \\ \boldsymbol{x}(0) &= \boldsymbol{x}_0, \end{aligned} \tag{2.10}$$

where

- $\boldsymbol{w}$ is the process noise vector of dimension $\mathbb{R}^{n\times 1}$,
- $\boldsymbol{e}$ is the measurement noise vector of dimension $\mathbb{R}^{q\times 1}$.

For convenience, it is assumed that both sequences are white noises independent on the state and input vectors and that they are from the normal probabilistic distribution. The state time development, in this case, can not be reconstructed even if the initial state vector $\boldsymbol{x_0}$, the input vector $\boldsymbol{u}(k)$, and the output vector $\boldsymbol{y}(k)$ at each time sample $k$ are known. That is because of two reasons. First, the state development is subject to the process noise $\boldsymbol{w}(k)$, introducing uncertainty to the state development. Second, the output vector is subject to the measurement noise $\boldsymbol{e}(k)$, which introduces additional uncertainty to the output vector. The linear state observer approach, in this case, is not recommended, as both the process and the measurement noises can not be measured. Therefore, they can not be chosen as inputs to the parallel observer model, and the system and its observer will receive different information.

Due to this fact, another approach must be employed in order to estimate the state development of the system. If the system is indeed LTI, and the process and the measurement noises are white, the optimal linear estimator can be derived. This estimator is called the Kalman filter, and it is described in more detail in the following subsection.

### 2.2.2 Kalman Filter

Stochastic LTI systems are, as mentioned in the previous subsection, subject to process and measurement noises. If these noises are white, then they are random variables that can not be measured, making them impossible to estimate. Typically, these noises are also Gaussian, meaning that they are defined by only one parameter, the covariance matrix. In this subsection, a Kalman filter algorithm [17] is introduced as an optimal linear state estimator. As can be seen further in this subsection, the Kalman filter uses only the information from the current time sample $k$ and the previous time sample $k-1$. Therefore, the Kalman filter memory requirements are low, whereas the computational speed of the algorithm itself is fast. Both these attributes together make the Kalman filter suitable for the real-time application.

### Assumptions

A Kalman filter algorithm can be applied to stochastics LTI systems that are modeled perfectly. Moreover, it assumes that process noise $\boldsymbol{w}$ and measurement noise $\boldsymbol{e}$ are uncorrelated white noises from a normal probabilistic distribution:

$$\begin{aligned} \boldsymbol{w} &\sim \mathcal{N}(0, \boldsymbol{Q}), \\ \boldsymbol{e} &\sim \mathcal{N}(0, \boldsymbol{R}), \end{aligned} \tag{2.11}$$

where

- $\boldsymbol{Q}$ is a process covariance matrix of dimension $\mathbb{R}^{n \times n}$ that is exactly known,
- $\boldsymbol{R}$ is a measurement covariance matrix of dimension $\mathbb{R}^{q \times q}$ that is exactly known.

If all assumptions are satisfied, the Kalman filter algorithm is an optimal filter.

### Data-update Step

The Kalman algorithm has two stages, the *data-update* and the *time-update* step. The data-update step, sometimes called the *correction* phase, is done each time a new measurement is received, and it is described by the following equations:

$$\begin{aligned} \boldsymbol{K}(k) &= \boldsymbol{P}(k|k-1)\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{P}(k|k-1)\boldsymbol{C}^T + \boldsymbol{R})^{-1}, \\ \hat{\boldsymbol{x}}(k|k) &= \hat{\boldsymbol{x}}(k|k-1) + \boldsymbol{K}(k)(\boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k|k-1) - \boldsymbol{D}\boldsymbol{u}(k)), \\ \boldsymbol{P}(k|k) &= (\boldsymbol{I}_n - \boldsymbol{K}(k)\boldsymbol{C})\boldsymbol{P}(k|k-1), \end{aligned} \tag{2.12}$$

where

- $\boldsymbol{I}_n$ is the identity matrix of dimension $\mathbb{R}^{n \times n}$
- $\boldsymbol{K}$ is the Kalman gain matrix of dimension $\mathbb{R}^{n \times q}$,
- $\boldsymbol{P}$ is the state covariance matrix of dimension $\mathbb{R}^{n \times n}$,
- $\hat{\boldsymbol{x}}$ is the state estimate vector of dimension $\mathbb{R}^{n \times 1}$.

The data-update step can, in general, use several different measurements to update the state estimate, as long as all measurements satisfy the assumption conditions specified in Equations (2.11). This process is called *sensor fusion*, and each measurement has its own matrices $\boldsymbol{C}$, $\boldsymbol{R}$ defined, which are used when the corresponding measurement is available and should be fused into the estimate. Since each new measurement (unless its error is infinite) increases the total information on the states, the estimation's uncertainty, described by the covariance matrix $\boldsymbol{P}$, decreases, and the estimation's accuracy rises.

As mentioned in Section 2.1.3, this approach is very useful, as one category of the sensors is typically subject to drift, and the measurements from the other category are often received at a lower frequency than necessary. If the Kalman filter algorithm is employed, the measurements from both sensor categories can be fused together, and their disadvantages can be compensated.

### Time-update step

The second stage of the Kalman algorithm is the time-update step, sometimes called the *prediction* phase. Unlike the data-update step, the time-update step runs in a cycle with the frequency defined by the desired frequency of the estimation rather than every time a new measurement is received. This essentially means that, typically, there are multiple time-update steps between two data-update steps, because many sensors run on frequency much lower than the one desired by the estimation. The following equations define the prediction phase:

$$\begin{aligned}\hat{\boldsymbol{x}}(k+1|k) &= \boldsymbol{A}\hat{\boldsymbol{x}}(k|k) + \boldsymbol{B}\boldsymbol{u}(k),\\ \boldsymbol{P}(k+1|k) &= \boldsymbol{A}\boldsymbol{P}(k|k)\boldsymbol{A}^T + \boldsymbol{Q}.\end{aligned} \tag{2.13}$$

In Equation (2.13), it can be seen that unless the process noise is zero, the uncertainty on the states increases.

### Variants of Kalman Filter

The basic variant of the Kalman Filter, as mentioned in Formula (2.11), has a number of assumptions. If one of these assumptions is not satisfied, the Kalman filter algorithm described above might not perform correctly. There, however, exists approaches and extensions that deal with situations like this that are briefly introduced, and are described in more detail in [16].

If the process and measurement noises are correlated, then the data-update and time-update steps can be performed as one combined step. If this is not desirable (e.g., due to slow measurements), there exists an approach that uses a system transformation to recover separated phases.

If one or both noises are colored, then the system's augmentation is done to transform to behave as white noises. Afterward, a standard algorithm can be used while considering this augmented system.

The major limitation of the Kalman filter is that it can be employed only by linear systems. There are, however, two variants of the filter that address this issue and show promising results.

The first one is the extended Kalman filter (EKF) that uses linearization of the non-linear model, and on this linearized model, the standard Kalman algorithm is run. However, the linearization brings a few disadvantages. First, the filter, in general, loses its optimality. Additionally, the filter might quickly diverge if the system is not modeled precisely, or the initial estimate is wrong. Even so, the EKF is nowadays widely used and considered as a standard in many applications.

The second one is the unscented Kalman filter (UKF), proposed in [18]. The major limitation of the extended Kalman filter is its application on highly non-linear systems, where linearization is not sufficient approximation of the system. Instead of linearization, the UKF uses the unscented transformation. A sufficient amount of sample points around the mean are chosen and then propagated through non-linear functions while estimating new mean and covariance. This also removes the need for calculating the Jacobian necessary for the linearization, which sometimes is a difficult task by itself.

Both non-linear filters mentioned above are nowadays used, and since each of them performs differently in terms of running speed and quality of results in different scenarios, the question of which one is better to use is specific for each individual system.

### 2.2.3 Particle Filter

As discussed in Section 2.2.2, the KF's optimality is assured if applied to linear systems that are subject to white noise. If any of these is not held, the KF remains the optimal linear filter, but there is no guarantee that a non-linear filter would not perform better. In fact, it was proven that for a uniformly distributed measurement noise, a non-Kalman filter is optimal [19]. Non-linear variants of the KF are discussed in Section 2.2.2, but they are not proven to be optimal. On the other hand, if the system is strongly non-linear, the performance of these KFs is considered insufficient as they only approximate the non-linear systems. In this subsection, an alternative to the KF called the particle filter (or sequential Monte Carlo filtering) is briefly introduced.

A particle filter is a probability-based non-linear state estimator. Compared to the EKF and UKF algorithms, it typically performs better while applied to non-linear systems considering the estimation error, as it does not use an approximation of the non-linear systems, whereas the EKF uses a first-order approximation and UKF uses a higher-order approximation. On the other hand, the computational cost of the estimation is increased [20].

#### Initialization

The initialization of the particle filter assumes that a non-linear system is defined by Equations (2.14). As a time-invariant assumption was used in Section 2.2.2, the same assumption is made for Equations (2.14). This assumption is made only for better orientation in equations, as both the KF and particle filter can be applied on time-variant systems in a similar manner.

$$\begin{aligned} \boldsymbol{x}(k+1) &= \mathrm{f}(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{w}(k)), \\ \boldsymbol{y}(k) &= \mathrm{g}(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{e}(k)), \end{aligned} \tag{2.14}$$

where

- $\mathrm{f}(\cdot)$ is a non-linear state transition function,
- $\mathrm{g}(\cdot)$ is a non-linear measurement function.

Moreover, the assumption of the known initial state of each particle is necessary, along with the $N$ specifying the number of particles[1] used for the state estimation. The particle filter then runs iteratively, taking into account only the information from the

[1] Increasing $N$ increases the estimation accuracy, as well as the computational cost.

current time sample $k$ and the previous time sample $k-1$.

#### Time-update step

Similarly to the KF time-update step, the time-update step of the particle filter uses the state transition matrix to predict each particle state vector, as described in Equation (2.15).

$$\hat{\boldsymbol{x}}_i(k|k-1) = \mathrm{f}(\hat{\boldsymbol{x}}_i(k-1|k-1), \boldsymbol{u}(k-1), \boldsymbol{w}_i^g(k-1)), \tag{2.15}$$

where

- $i = 1, 2, ..., N$ is the index of the particle,
- $\boldsymbol{w}^g$ is a randomly generated noise vector based on the process noise PDF[1].

#### Data-update step

When a measurement is obtained, a relative likelihood of each particle is calculated, as can be seen in Equation (2.16).

$$q_i(k) = \mathrm{p}(\boldsymbol{y}(k), \hat{\boldsymbol{x}}_i(k|k-1)), \tag{2.16}$$

where

- $q$ is the relative likelihood of a particle,
- $\mathrm{p}(\cdot)$ is the PDF.

The relative likelihood is then normalized by Formula (2.17) so that the sum of likelihoods across all particles is equal to one.

$$q_i(k) = \frac{q_i(k)}{\sum_{j=1}^{N} q_j(k)}. \tag{2.17}$$

The normalized likelihood forms the probability density function $\mathrm{p}(\hat{\boldsymbol{x}}(k|k)|\boldsymbol{y}(k))$. Finally, a new set of particles is generated from this probability density function in a resampling step, which is shown in Equation (2.18).

$$\hat{\boldsymbol{x}}_i(k|k) \sim \mathrm{p}(\hat{\boldsymbol{x}}(k|k)|\boldsymbol{y}(k)). \tag{2.18}$$

[1] PDF stands for probability density function.

# Chapter 3 System Architecture

Although the main topic of the thesis is UAV localization, additional problems arise for a proper experimental evaluation such as trajectory planning, UAV control, and failsafe mechanisms, as can be seen in Figure 3.1.

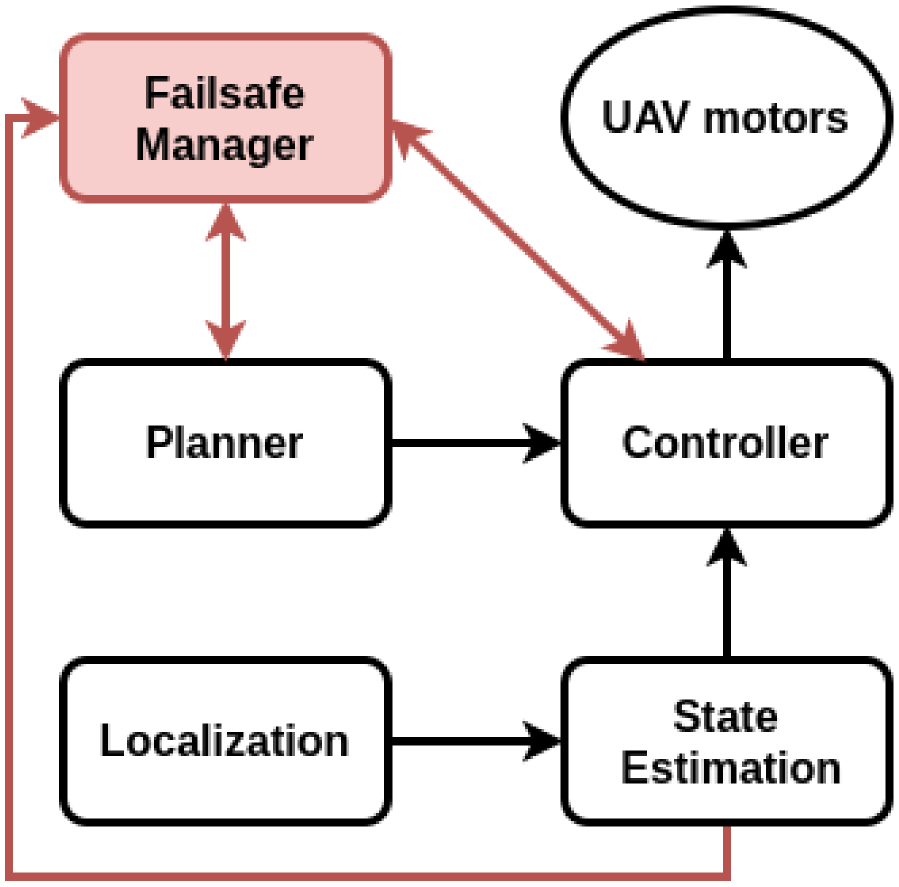


**Figure 3.1.** High-level architecture design.

In this chapter, the design of the architecture seen in Figure 3.1 is described in more detail along with used tools for its solution[1].

The first part of the chapter is dedicated to a brief introduction of the ROS[2], Gazebo, and MRS[3] frameworks. In the second part, a few existing ROS packages used

[1] Note that the failsafe mechanism solution is described in Chapter 4.
[2] Robot Operating System.
[3] http://mrs.felk.cvut.cz/

for UAV localization and sensor fusion are discussed. In the third and final part, the proposed architecture of the whole system and its configuration are introduced, and based on the UWB localization system requirements, one of the sensor fusion packages is chosen for the state estimation.

## 3.1 Frameworks Introduction

In this section, the essential frameworks into which the external UWB localization system is going to be integrated are introduced. These frameworks are ROS, Gazebo, and MRS, which are discussed in more detail in the following subsections.

### 3.1.1 ROS Framework

ROS [21] is an open-source robotic framework that significantly reduces the implementation complexity necessary for a given application by offering driver-level software, hardware abstraction, etc. ROS is designed to support multiple programming languages. That way, the implementation of additional functionality is done by implementing a ROS package. ROS package is a directory that roofs an application, allowing its easy reusability by integrating it into another system. The ROS architecture is intended to be a system consisting of many processes running in a peer-to-peer topology.

These processes are called *nodes*. As ROS is designed as a modular system, each *node* can be implemented to fulfill a single task, and the whole application consists of many *nodes* communicating among themselves through *messages*. These *messages* are strictly defined, and they can be composed by standard primitive data types, as well as other already defined ROS *messages*.

The communication itself is done either via *topics*, *services*, or *actions*[1].

The *topic* communication is based on a broadcast approach, meaning that the transmitting and receiving nodes are typically not aware of each other existence, and there might be either a single or multiple nodes on both the transmitting and receiving end of the communication. In the ROS framework, nodes on the transmitting end of the communication are called *publishers*, and nodes on the receiving end of the communication are called *subscribers*. Each node can publish and subscribe to multiple topics. Each topic is specified with its name in the form of a string data type. In a multi-robot system, the need for communication via the same topics might be necessary. Nodes can be launched with a predefined namespace, adding a prefix before the topic name to avoid the topic name conflicts. An example of a simple communication graph can be seen in Figure 3.2. In this example, node A is a sole publisher to the topic 1, which in turn has a single subscriber, node D. Nodes C and D are both publishers to the topic 2, that has a single subscriber, node E. Node D is

[1] The *action*-based communication is not native to the ROS framework. It is implemented as ROS package, available at `http://wiki.ros.org/actionlib`.

also publishing on topic 3, which is subscribed by two nodes, F and G. Node B is not communicating with other nodes via topics.

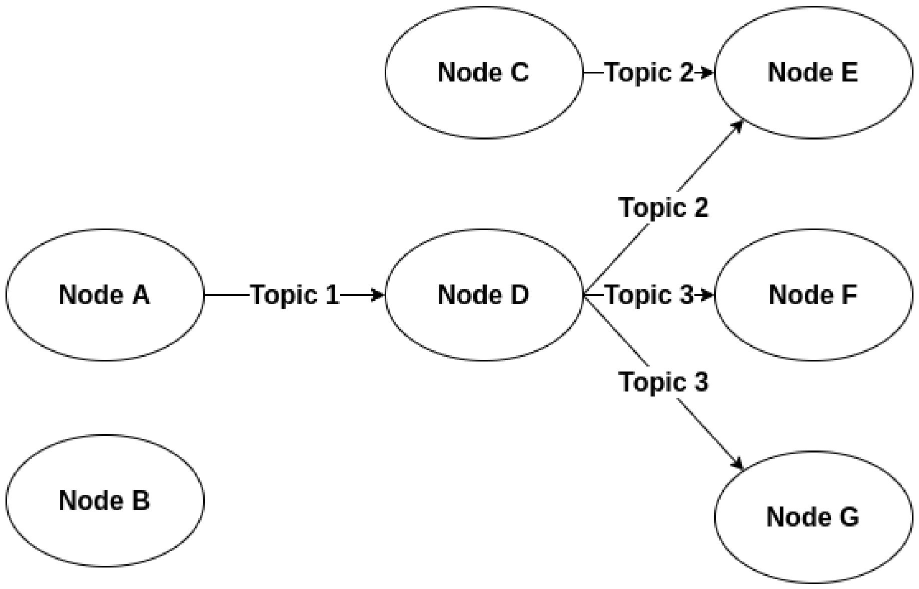


**Figure 3.2.** Example of ROS communication via topics.

The broadcast nature of the topic communications is not appropriate in case of a request-reply communication need. For this purpose, the service communication is offered. The communication via services uses similar messages as the communication via topics, but each service must have both the request and reply messages defined. Unlike the topic communication, only a single node might be advertising a service. This node is called a *server*. Nodes that are sending the request message to the service server are called clients. After a client sends a request message, it awaits the reply. If the service offered by the server takes a long time to execute, this waiting might be undesirable, as the client node is blocked (assuming that is it not implemented as a multithreaded node).

The blocking issue mentioned in the previous paragraph can be resolved by implementing the process as an *action*. The action communication architecture offers the same functionality as the service communication. However, the user can also receive the feedback while the action defined the server is executed, as well as cancel the executing of the action if a specific event happens. Each action has three messages defined: the goal message, the feedback message, and the result message.

### 3.1.2 Gazebo Simulator

The Gazebo [22] simulator was chosen for the development since it can be easily used in combination with the ROS framework. Among its perks is its user-friendly interface, both graphical and programmatic, realistic and robust physics engine, and high-quality graphics. Finally, the Gazebo is distributed as freeware.

The environment in which the Gazebo simulation is running is called *world*. It is represented by the .world extension file, a specific file defined by SDF[1]. In this file, the user can specify how the world is illuminated, how the ground interacts with other items, the physics engine used in the simulation, the gravitational acceleration affecting the world, etc. This file is also responsible for populating the world with models representing, e.g., robots, sensors, and buildings.

As mentioned in the paragraph above, an object can be represented by *model* directly defined in the .world file, but a model defined that way can not be used in different Gazebo world. Therefore, it is better to specify the model separately and afterward include it into the .world file. Similarly to the world environment, the model is defined by the SDF file. The SDF file is a set of *links* that are connected via *joints*. The link can also be extended with the *collision*, *visual*, and *inertial* elements. It is not uncommon for individual elements to differ from one to another. Typically, the visual element is modeled much more precisely via triangle mesh, while the collision and inertial elements are, if possible, approximated by some simple shape to reduce the simulation's consumption of resources.

The Gazebo also introduces the so-called *plugins*. The plugin is essentially a C++ code compiled as a shared library, allowing it to be included in the simulation. Plugins can be used to control either the simulation world or the particular models or be used as sensors.

However, due to the integration with ROS, which uses a slightly different XML-based structure of files called URDF[2], it is necessary to use the URDF files which are then converted into the SDF files. Models created in such a way can be used both in ROS (visualization in *rviz*, communication via topics), as well as in Gazebo (physical interaction with the world).

The URDF files are often developed using the XML Macros (or xacro), allowing higher readability and easier maintenance of models employing expandable macros.

### 3.1.3 MRS Framework

MRS framework [23], [24] is a group of ROS packages developed for the deployment on UAV systems that are equipped with a Pixhawk PX4 [25] flight controller. The most important parts (considering the integration part of the implementation of this thesis, described in Section 5.2.1, along with the simulation Section5.1) of this framework are introduced in this section. However, as Section 3.2 is discussing several ROS packages used for the UAV localization and sensor fusion, the part of the MRS system solving the state estimation problem, the *mrs_uav_odometry*, is described in more detail in Section 3.2.3.

The MRS UAV system defines a vector of four independently controllable degrees of freedom. These are the position parameters describing the UAV position in the world frame, accompanied by a parameter describing the heading of the UAV. These

[1] Simulation Description Format.
[2] Unified Robot Description Format

parameters, along with their derivations[1], are used for a feedforward trajectory tracking that allows proper state-feedback control. This tracking is implemented as the linear model predictive control (MPC) algorithm. For the feedback control, several controllers are employed for different parts of the UAV mission, such as takeoff, landing, mission conducting, emergency, etc. For the controllers' automatic switching (when finishing the takeoff, if an emergency has been detected), a manager node is implemented.

Additionally, the MRS UAV system offers several already implemented services used for high-level control of the UAV, such as a flight to the desired position with a given heading angle. The MRS UAV system also comes with a simulation package designed for the ROS simulator with three available UAV models and countless models for worlds, sensors, etc.

## 3.2 ROS Localization Packages

In this section, the already implemented ROS packages solving the localization problem and sensor fusion are discussed. The KF approach was chosen over the particle filter, particularly because many KF, EKF, and UKF models perform reasonably well on non-acrobatic UAVs, because of the increased computational cost of the particle filters, and because of the work necessary for integrating the UWB localization system with already existing ROS packages. The following three packages were considered: the *robot_localization*[2][26], the *ethzasl_msf*[3] [27], and the already mentioned *mrs_uav_odometry*.

### 3.2.1 Robot Localization

The robot localization package is consists of two state estimation nodes implementing two non-linear Kalman filter variants, namely the EKF and UKF. Besides the estimating algorithm itself, the configuration of both nodes is pretty much the same. In the prediction step, the omnidirectional 6DoF motion model is used. The model consists of 15 states, three parameters describing the robot position, their first and second derivations, and three parameters for the robot orientation and their first derivations. Several sensors measuring any of the states can be fused into the state estimation. The package allows arbitrary configuration of the process and measurement noise covariance matrices.

[1] I.e., position, velocity, acceleration, and jerk for each axis in the world frame, along with the heading and the heading rate.
[2] Available at `https://github.com/cra-ros-pkg/robot_localization`.
[3] Available at `https://github.com/ethz-asl/ethzasl_msf`.

### 3.2.2 ETHZ ASL Multiple Sensor Fusion

The ETHZ ASL multiple sensor fusion package is designed specifically for the micro UAV application. It uses an EKF algorithm with a 6DoF motion model. However, the package is also able to estimate the IMU acceleration and gyroscope biases. It can also estimate the roll and pitch drift, sensor transformation between a sensor and the IMU, and pose scaling of a measurement sensor. The sensor fusion algorithm is even able to compensate for the time delay in measurements. On the other hand, the EKF is fixed to use the IMU readings to predict the state at the IMU rate.

### 3.2.3 MRS UAV Odometry

The MRS UAV odometry package uses a different approach than the two packages mentioned above. Instead of using a single KF, it uses three linear KFs, each estimating a different set of states. The first KF estimates the UAV 2D position, velocity, and acceleration. The second KF is used to estimate the UAV heading, heading rate, and heading acceleration. The third KF estimates the UAV altitude, altitude velocity, and altitude acceleration. By combining these 3 KFs, a 4DoF model is obtained. This modularity assumes that the states from different sets are not correlated. Furthermore, the measurement covariance matrix is assumed to be diagonal with the same-valued diagonal elements. This assumes that, e.g., the 2D position measurements are not correlated. On the other hand, the modularity allows easy substitution of, e.g., one heading estimator with another during the flight without modifying the rest of the system.

The system is already integrated with the rest of the MRS framework, i.e., controllers and trackers, and it offers a possibility of using the desired acceleration and heading rates[1] in the prediction step to propagate through the system matrix. Because of that, the IMU readings can be used in the correction step if deemed advantageous.

## 3.3 Architecture Design

This section is dedicated to system architecture design. The first part briefly introduces the sensors used, along with some necessary consideration for their deployment. The second part discusses the properties of the external UWB localization system. The third and final part introduces the proposed architecture.

### 3.3.1 Sensors Used

As the UWB localization integration done in this thesis is intended for real-world experiments in an indoor environment, the choice of additional sensors for the state

[1] Outputs of the controllers.

estimation is limited. Assuming that the typical UAV mission is not acrobatic, the state model has 4DoF: three linear parameters and one angular corresponding to the heading. The IMU readings (or the desired linear acceleration and heading rate for the MRS UAV odometry case) can be used for a prediction step of the KF algorithm. However, the correction step assumes that some other sensors are employed, and to eliminate the drift of the prediction, the chosen set of sensors should cover all 4DoF parameters.

The external UWB localization system, described in more detail in Section 3.3.2, provides 2D position measurements.

For the altitude estimation, a laser rangefinder can be used, with several requirements kept in mind. Even though the roll and pitch states are stabilized throughout the flight via the Pixhawk autopilot, nevertheless, the change in UAV speed typically results in a deviation from the zero angles. The Equation (3.1) is used as a correction to the rangefinder measurement is necessary to obtain a reliable estimate of the altitude:

$$z = (z_m - x_O\mathrm{s}(\theta) + y_O\mathrm{c}(\theta)\mathrm{s}(\phi) + z_O\mathrm{c}(\theta)\mathrm{c}(\phi))\mathrm{c}(\theta)\mathrm{c}(\phi), \tag{3.1}$$

where

- $z$ is the altitude estimate [m],
- $z_m$ is the measured altitude in the rangefinder frame [m],
- $x_O, y_O, z_O$ are the $x, y, z$ position offset of the rangefinder from the UAV mass body center [m],
- $\theta$ is the UAV pitch angle [rad],
- $\phi$ is the UAV roll angle [rad],
- $\mathrm{s}, \mathrm{c}$ are the $\sin, \cos$ functions.

Another thing to consider is that in an indoor environment, there is typically less available space for safe maneuvering than outdoors due to a higher number of obstacles. Therefore, during the flight, obstacles are usually much closer to the UAV. Depending on the height of the obstacle and flight altitude, a rangefinder fixed on the UAV can suffer from the roll or pitch angle change by detecting, e.g., the wall behind itself. This issue can be solved by incorporating the information about the static obstacles in the environment, adjusting the plan of the mission, developing a correction algorithm based on the current state estimate and the environment model, or, if a gimbal is integrated to the UAV base, fixing a rangefinder sensor onto it rather than onto the UAV base[1].

The typical heading measurements sensor, a magnetic compass, will not be used in this thesis. The underlying reason is the fact that in most human-made indoor environments, a large concentration of the ferromagnetic metal material is usually present, affecting the compass measurements. It was proposed that the use of two UWB sensors, the heading can be estimated from their relative position, as described in Equation (3.2), assuming that one tag is specified as the front tag (FT) and the other as the back tag (BT).

[1] In that case, the correction due to the roll and pitch angle change should not be used.

$$h(k) = \text{atan2}(y_{FT}(k) - y_{BT}(k), x_{FT}(k) - x_{BT}(k)) - h_O, \tag{3.2}$$

where

- $h$ is the estimated heading [rad],
- $x$ is the measured $x$ position [m],
- $y$ is the measured $y$ position [m],
- $h_O$ is the offset from the UAV's yaw axis [rad].

In general, the measurement times of each tag differ, and so the Equation (3.2) should contain some kind of correction for this time difference. However, the tags are assumed to be close to each other (depending on the size of the UAV), they are both receiving the same UWB signal from antennas, and the UAV's speed is relatively low, the resulting error of the time difference is negligible.

Unfortunately, the heading calculated this way heavily correlates with the 2D position measurements. Furthermore, it is evident that since the distance between FT and BT is relatively small, each position estimate error will be significantly projected to the heading. Thus, a specific KF has to be designed to include the information about the relative distance between FT and BT, which is fixed, and that correlates the heading parameter with the FT and BT measurements. Sadly, neither of the packages discussed in Section 3.2 is capable of such a thing. The design of such a model is out of the scope of this thesis. Therefore, some compromises that are introduced later in this thesis were necessary.

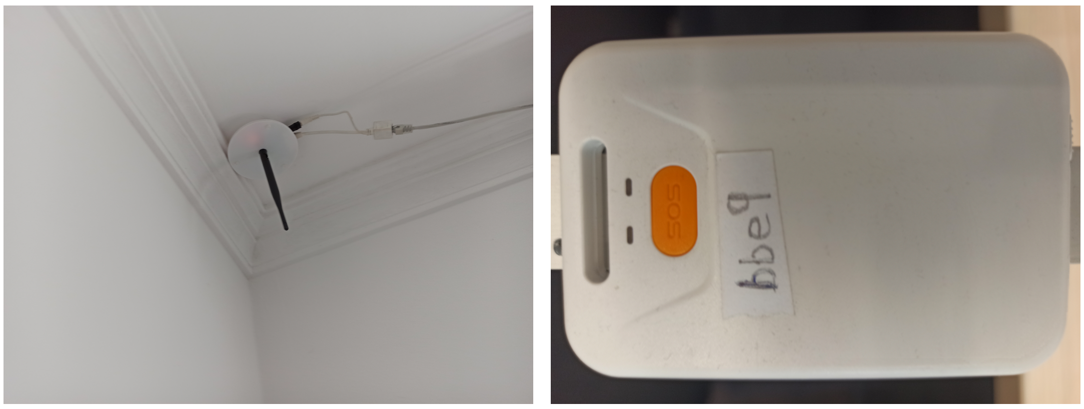
### 3.3.2 The External UWB Localization System

The external UWB localization system was obtained from ALIS Tech[1]. ALIS Tech is a Czech company specializing in localization and anti-collision systems in warehouse environments with human operators. The company also played a role in providing support with installing the demonstrative installation.

Based on the information from the system manufacturer, the system can provide real-time localization with an accuracy of up to 50 cm. However, this information was not accompanied by a precision specification, and no datasheet was received to support this claim. It was decided to conduct several experiments, described in more detail in Section 4.2, to verify the accuracy and obtain additional information about the localization system behavior.

[1] `https://alis-tech.com`.

The experimental system architecture follows the description in Section 2.1.6. It consists of four anchors that are interconnected through ethernet. These anchors have synchronized time among themselves. Two tags able to communicate with these anchors are also part of the system. These tags have an integrated IMU sensor. However, the tags' time is not synchronized with the anchors' time. Both the anchor and the tag can be seen in Figure 3.3. The illustration of the system deployment can be seen in Figure 3.5, where the UWB tags integrated with the UAV base, BT and FT, are localized by four anchors, A1-4. The localization of tags occurs in the fixed frame with origin given by the anchor A1 as depicted in the illustration.



The UWB anchor. The UWB tag.

**Figure 3.3.** The UWB anchor and the UWB tag.

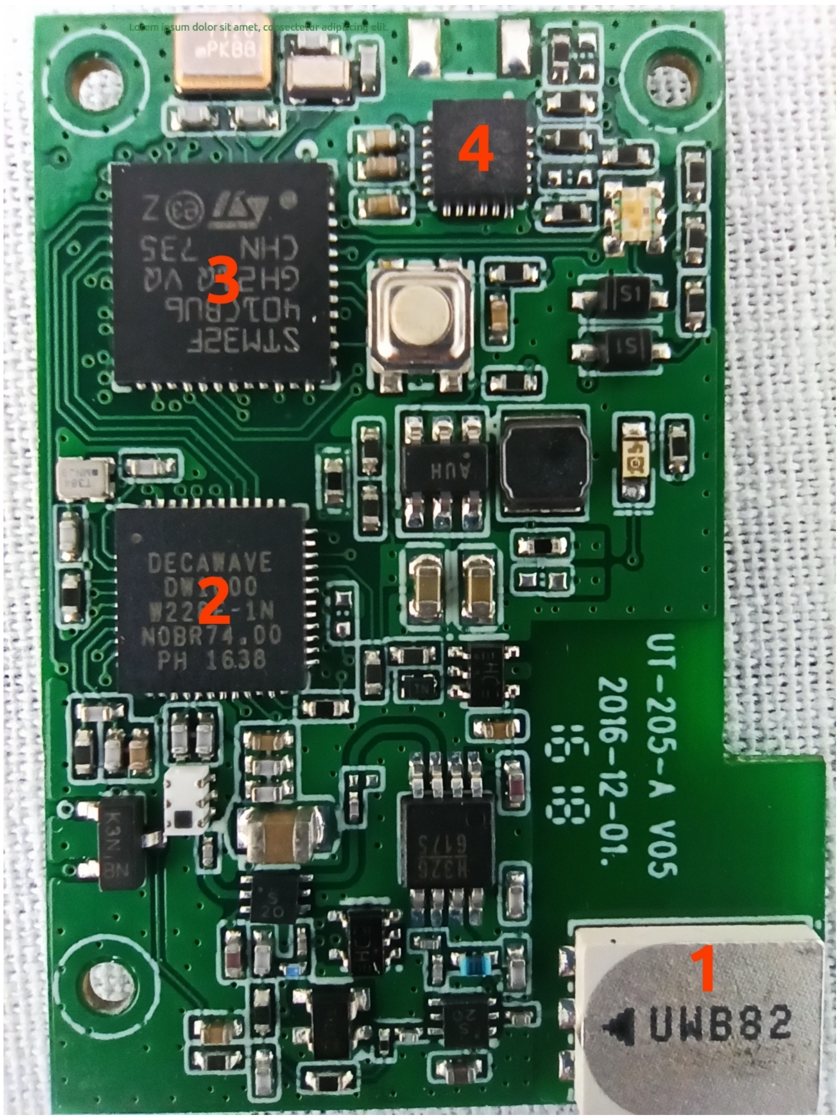


**Figure 3.4.** The UWB tag electric board. The highlighted components are (1) the antenna, (2) the UWB compliant wireless single-chip receiver DW1000, (3) is the STM32F 32-bit microcontroller, and (4) is the tag's IMU.

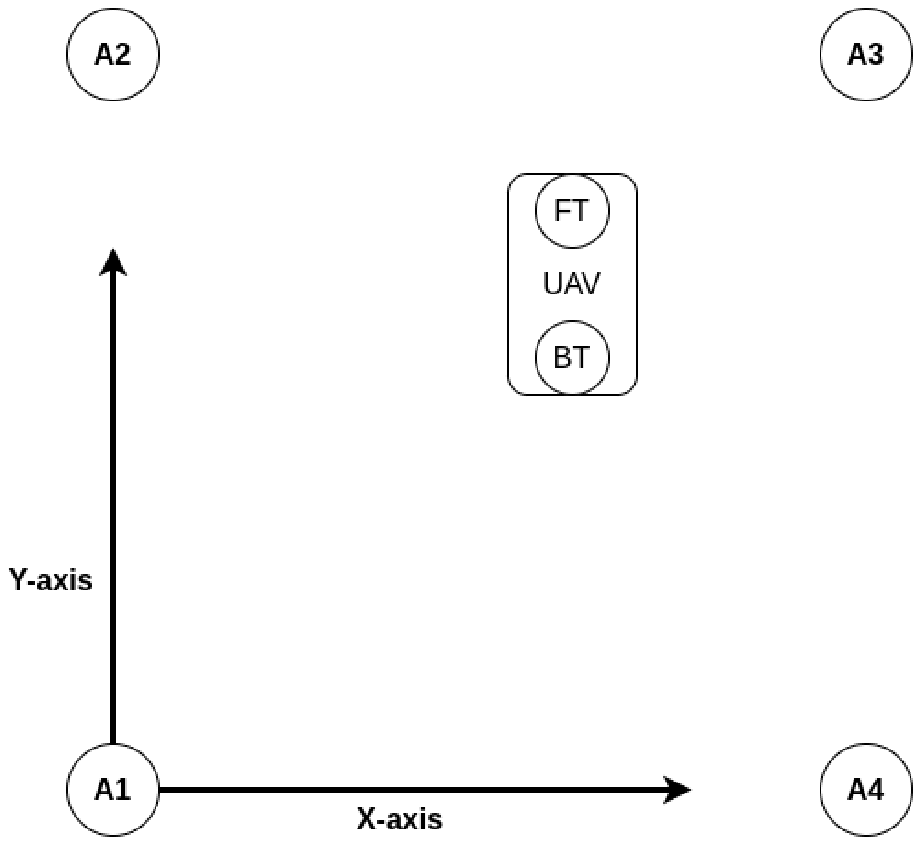


**Figure 3.5.** Illustration of the UWB localization systems deployment.

The last part of the system is a computational server, implemented as a proprietary back-end software. The server is provided by the manufacturer for the configuration and operation of the localization system. The server also processes the anchor-tag distance measurements based.

It is essential to specify the IP address of all anchors and their position in the UWB localization coordinate system, which is assumed to be a 3D Cartesian coordinate system. That means that each anchor's position is defined by x, y, and z coordinates. The relative positions (in cm) among anchors define the coordinate system. Thus, the real-world relative position between anchors has to be measured correctly, as any error can affect the quality of the localization. Unfortunately, these errors are likely to occur. Because of that, the system has a built-in function used to correct the distances among anchors.

The ID of the tags, whose position is estimated, must be specified. Additionally, the maximum speed of individual tags can be specified to provide a more reliable estimation.

It is necessary to choose which localization technique should be used for the localization. The system allows choosing between TDoA and ToF techniques. As described in Section 2.1.4, the ToF places greater emphasis on time synchronization, as the time has to be synchronized between both sides of the communication. As mentioned at the beginning of this subsection, only anchors have their time synchronized.

Because of that, the ToF technique uses the TWR method to eliminate the time offset, and as is discussed in Section 2.1.4, this method introduces additional time delay due to an increased number of messages necessary to obtain a distance measurement.

A location calculation can be modified. The system offers processing of measurements from three or four anchors, resulting in a 2D or a 3D localization. If raw measurement data are not desirable, the system also offers a KF with nine modes. Mode 0 corresponds to the raw data measurements, and each subsequential mode increases the aggressivity of the KF. However, the precise configuration of each mode is unknown.

The full location estimation process can be seen in Figure 3.6.

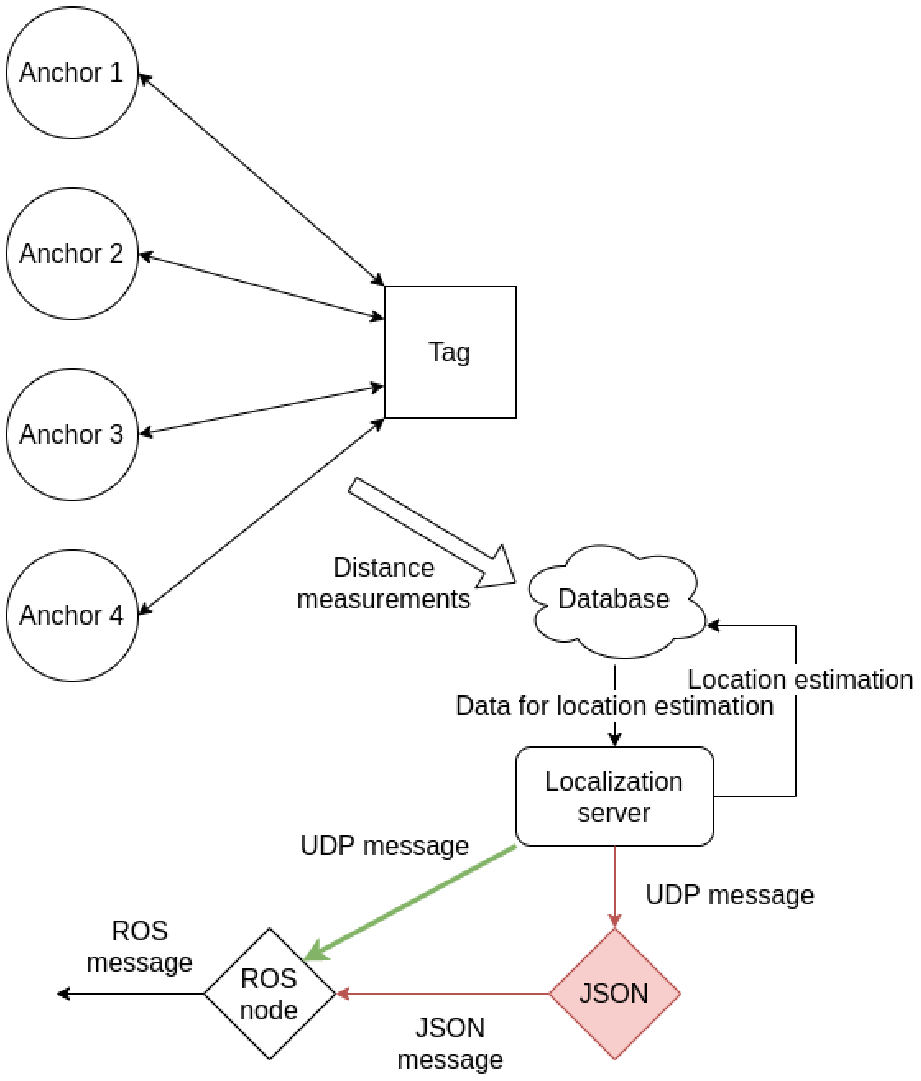


**Figure 3.6.** The UWB localization system process diagram.

The communication between anchors and tag provides a distance estimate that is stored in the database. However, this is only true if the tag is not steady. If it stops moving, the communication is interrupted as the tag enters a power-saving mode. Once the tag starts moving again, the communication starts again as well. Based on the specification of the localization algorithm, the database provides necessary data to the localization server. The localization algorithm then estimates the tag's position and stores this estimate in the database. The estimate is also converted from the UDP to the JSON message and then transmitted to the specified destination. This step seems unnecessary, and because of that, the JSON conversion has been cut off, and the UDP message is directly transmitted to the ROS node responsible for the conversion to the ROS message.

### 3.3.3 Proposed Architecture

To propose an architecture of the UAV onboard control system, it is first important to choose the state estimation package. Based on the results of the experiments in Section 4.2, it can be seen that for some configurations, the UWB tag noise model can be approximated with an uncorrelated 2D gaussian distribution. As discussed earlier in the chapter, neither state estimation ROS packages available are sufficient to include the heading estimation from the relative position of both UWB tags. Because of those two facts, the disadvantage of the modular MRS state estimator regarding the correlation of parameters was not considered. Since the MRS state estimator is already integrated with the MRS UAV system, it was chosen as the state estimator package for this thesis's sake.

Two UAV system architectures are proposed. The first one is an ideal case architecture that would correlate the heading estimation with both tags' position estimations. The second one is an actual architecture implemented, as the development of an ideal case KF is out of the scope of this thesis.

The ideal case can be seen in Figure 3.7. It consists of four sensors, namely the LiDAR rangefinder, the IMU, and two UWB tags. The rangefinder provides a distance measurement between the UAV and the ground that can be transformed into the $z$ estimate and used in the KF correction step. The IMU typically provides at least linear acceleration and angular velocity information, which can be used either in the KF prediction or correction step. Each UWB tag brings a new 2D position measurement. These are used to obtain both the 2D position estimate of the UAV and its heading estimate. The controller commands (desired linear acceleration and desired angular velocities) can be used in the KF prediction step. As the goal is to deploy the system on the real UAV, a failsafe and mission manager must be present. The failsafe manager receives information from the state estimator, and if a fault/failure of a defined kind is detected, the information is propagated to the mission manager. The mission manager is responsible for the autonomous transition between different phases of the mission and handling failsafe if necessary. The mission plan is then fulfilled with the help of a tracker, a controller, and an autopilot.

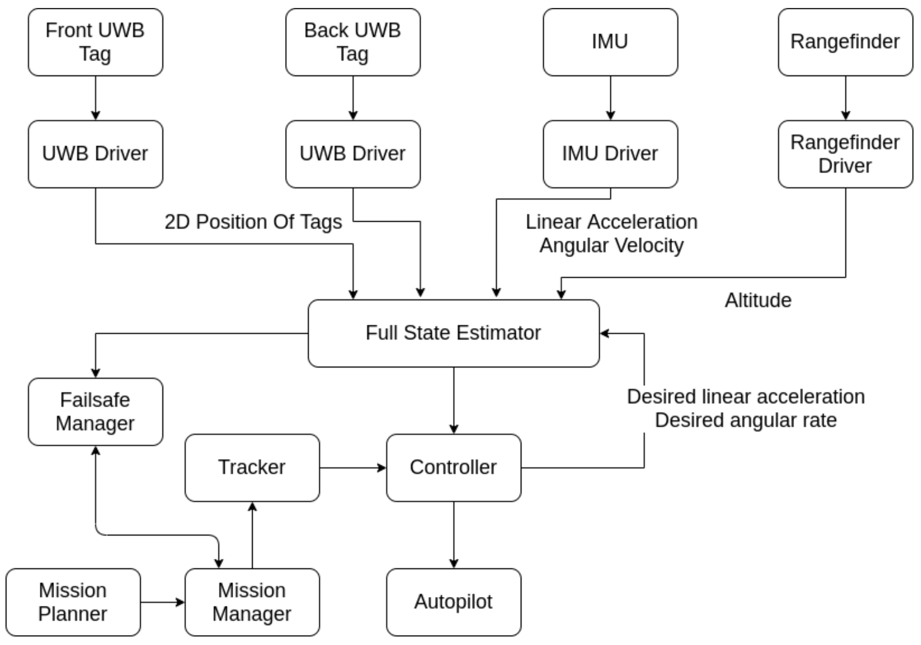


**Figure 3.7.** The ideal UAV system architecture.

The architecture used in this thesis differs mainly in the state estimation part, as discussed earlier in this chapter. States are divided into three sets, and each set is estimated by a different KF, as can be seen in Figure 3.8. Due to the issue discussed in Section 3.3.1 concerning the heading estimation from the relative position of both UWB tags, it was decided that for this thesis, another heading estimation must be considered. It was decided that the gyroscope can be used as a sole sensor providing heading information, as its drift is negligible for short demonstrations.

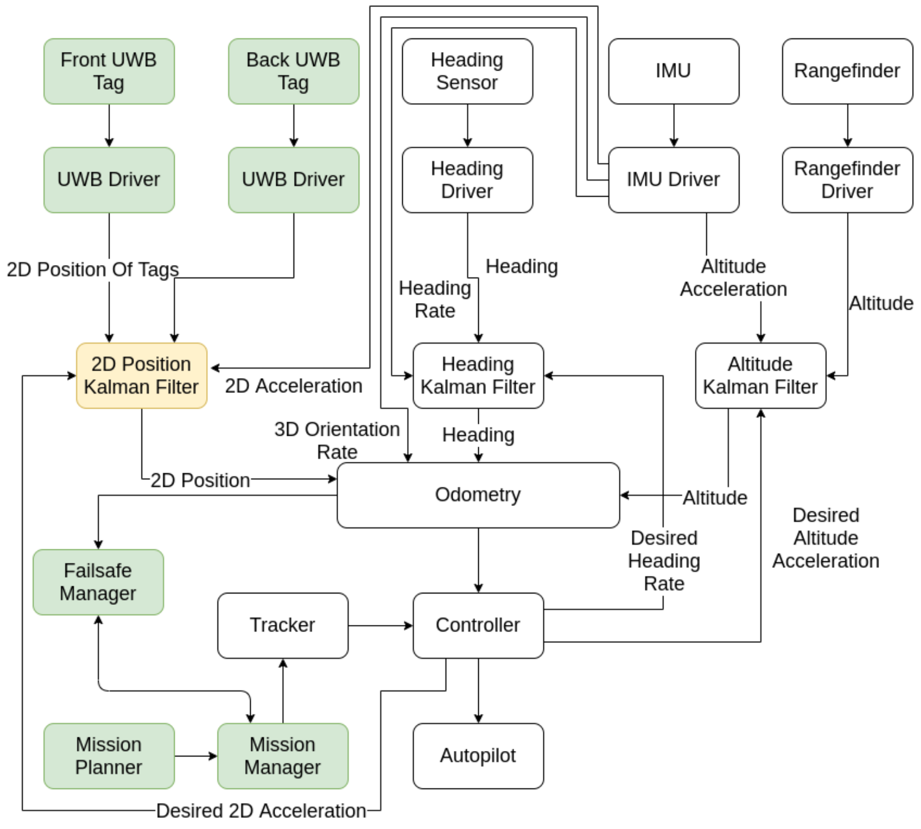


**Figure 3.8.** The real UAV system architecture. Green blocks are newly designed blocks, while yellow blocks represent components that were modified.

# Chapter 4
# UWB Tag Identification

For the proper functioning of the whole UAV onboard system, the identification of the UWB tag accuracy is necessary. In this chapter, the software and hardware implementation of the experimental platform is described. Afterward, a set of conducted experiments is discussed and analyzed. Based on the analysis, the UWB tag model parameters are determined.

## 4.1 UWB Tag Experiment Implementation

For the evaluation of the UWB localization system accuracy, a measurement analysis is necessary. In an environment already equipped with UWB anchors, the tag can be placed on the known position for a sufficiently long time, collecting the measurements. Afterward, by subtracting the known position from the measurements, the noise signal is obtained and can be further analyzed. However, as mentioned in Section 3.3.2, the UWB tags have a power-saving property that cannot be easily turned off (as it is a part of the firmware), preventing this static measurement.

Due to the impossibility of conducting the static measurement experiments, another approach was chosen. The UWB tag is placed on a platform on a fixed position. Afterward, the platform moves with a predefined trajectory. That way, the UWB tag will not enter the sleep mode, yet the known trajectory can still be subtracted from the measurements, exposing the noise signal.

### 4.1.1 Experiment Platform

The platform for obtaining dynamic measurements consists of several components. A bipolar stepper motor is used, along with a stepper motor driver carrier that supports the microstepping technique that allows a smooth run of the motor. The Raspberry Pi 3 computer controls the driver by sending a pulse width modulation signal to its STEP pin. A rod is mounted on the stepper motor, and each time a stepper motor moves, the rod rotates by a constant angle. Two UWB tags are placed 45 cm from the rod's center on the opposite sides. This setup allows the UWB tags to rotate in a fixed plane along a circular trajectory. The last used component is a transmissive optical sensor with phototransistor (light gate) output to track the full circle rotation. Each time a rod passes completes a full circle, the infrared light signal between emitter and detector is interrupted, increasing a voltage and detecting the passage.

The system schematic is shown in Figure 4.1.

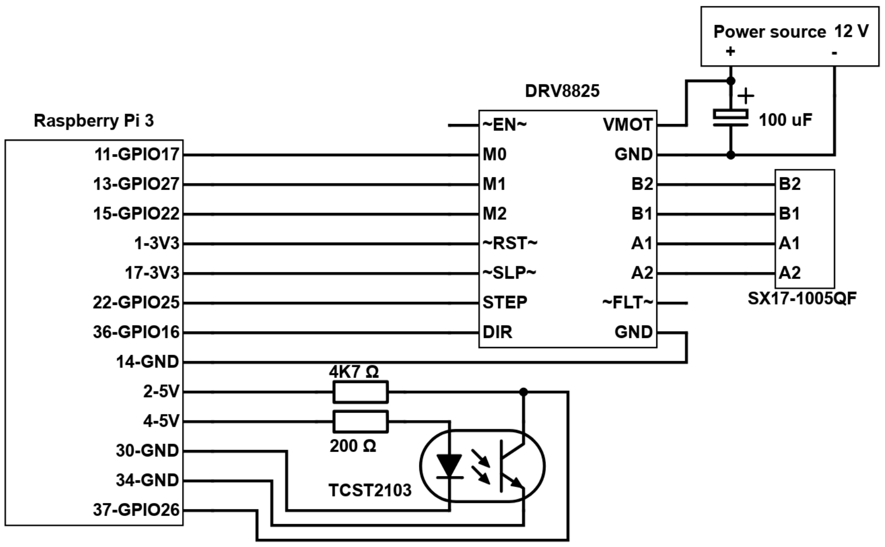


**Figure 4.1.** System schematic.[1]

A few considerations are necessary while setting the experiment up. The motor's current peak limit should limit the driver's maximum current output. The SX17-1005QF stepper motor used in this thesis has the current peak limit of 1 A [28]. To ensure that this value is not surpassed, the driver's active current limiting can be determined based on its reference voltage, as shown in Equation (4.1).

$$I_{OUT} = 2V_{REF}, \tag{4.1}$$

where

- $I_{OUT}$ is the driver's maximum current output [A],
- $V_{REF}$ is the driver's voltage reference [V].

Next, to prevent any damage due to the voltage spikes, a capacitor is placed as close to the driver as possible. Finally, the light gate is wired to the Raspberry computer. Based on the model, i.e., TCST2103 [29], the test conditions are voltage $V_{CC} = 5$ V and forward current $I_F = 20$ mA. Since Raspberry Pi can be used as a 5 V power source, it was decided that the light gate will operate under these conditions. The forward voltage corresponding to this forward current is $V_F \approx 1.1$ V. Since the source voltage is way higher than the voltage drops of the diode, it is necessary to include a resistor before the diode to limit the current through it and prevent its destruction. The value of the resistor needed is calculated by Equation (4.2).

[1] Block representing Raspberry Pi was reduced, and only pins that are used are shown (in format *pin number-function*).

$$R_1 = \frac{V_{CC} - V_F}{I_F}, \tag{4.2}$$

where

- $V_{CC}$ is the source voltage [V],
- $V_F$ is the diode's forward voltage [V],
- $I_F$ is the diode's forward current [A],
- $R_1$ is the value of the resistor before the diode [Ω].

After substituting into Equation (4.2), it can be determined that the value of the protective resistor should be $R_1 \approx 195\ \Omega$. A 200 Ω resistor was therefore actually used due to the availability. The platform can be seen in Figure 4.2.

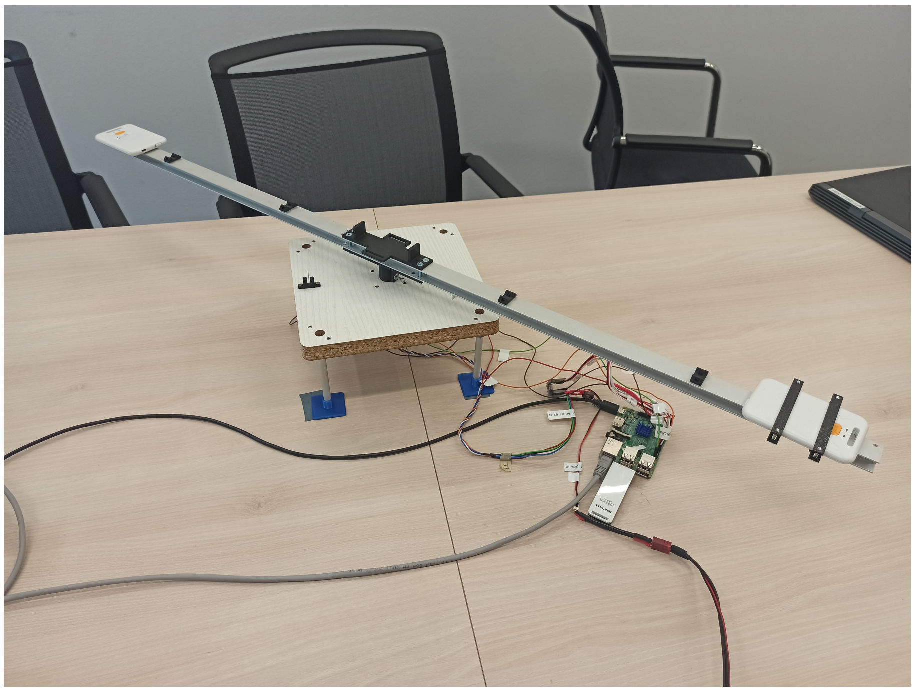


**Figure 4.2.** The platform used for experiments.

### 4.1.2 Control Software

Once the experimental platform is wired correctly, the next important step is implementing the control software that can run on the Raspberry computer connected to the stepper motor driver. Additionally, the ROS framework must be installed on the Raspberry computer, as another script is developed that can convert measurements into the ROS messages.

The script controlling the motor driver was implemented employing Python's *pigpio*[1] library that allows easy control of the Raspberry's GPIO pins. These pins can

[1] Documenation available at `http://abyz.me.uk/rpi/pigpio/`.

be specified inside the script to behave either as inputs or outputs. The script is used either as an input signal evaluator or an output signal generator. The output pins are the pins responsible for the control of the motor's direction, speed, and microstepping mode, while the input pin is connected to the light gate and evaluates the rising edge (RE) in the signal caused by an interruption due to obstacle passage.

While the motor control's microstepping mode and direction control are relatively straightforward, the motor speed control requires a little insight into the motor's behavior. For a motor to make a step, it must receive a pulse on the driver's STEP pin. Therefore, a pulse width modulation (PWM) is used, and the script allows to set a desired frequency and duty cycle of the signal. Given a target motor's RPM, the PWM signal's frequency can be calculated by Equation (4.3).

$$f = \frac{360nv}{60\theta}, \tag{4.3}$$

where

- $f$ is the PWM signal's frequency [Hz],
- $v$ is the target motor's frequency [rpm],
- $n$ is the number of microsteps per step [-],
- $\theta$ is the number of degrees per step [°].

As mentioned before, the developed script is also used to evaluate the signal transmitted from the light gate. As a rising edge on the signal appears when the obstacle passes through the light gate, the timestamp of the passage is saved for future data processing. A fast reaction to this rising edge is necessary to reduce the delay between the time of the actual passage and the saved timestamp. Luckily, the *pigpio* library offers callback function on the input signal's rising edge, interrupting the execution of the rest of the script, processes the function defined for the callback, and only afterward continues with the script's execution. The callback function simply determines the current timestamp and publishes it in the form of the ROS message on a defined topic. During the testing, it was observed that the callback is invoked multiple times during a single obstacle passing. Due to this, a simple debouncing algorithm was implemented that compares the current time with the last time the callback function was invoked. If the time difference is smaller than a defined threshold, the rising edge is considered an imperfection of the light gate sensor, and the timestamp is not published further.

Finally, a ROS publisher node is implemented that can receive the UDP communication from the UWB localization server, containing the calculated x, y position data of the tag specified by its ID. The node (further identified as the UWB tag driver) converts all this information, along with the current time, to a ROS message. Additionally, if the time between the Raspberry computer and the localization server is synchronized, the time delay of the measurement can be calculated by subtracting the timestamp of measurement from the current time. The whole setup can be seen in Figure 4.3.

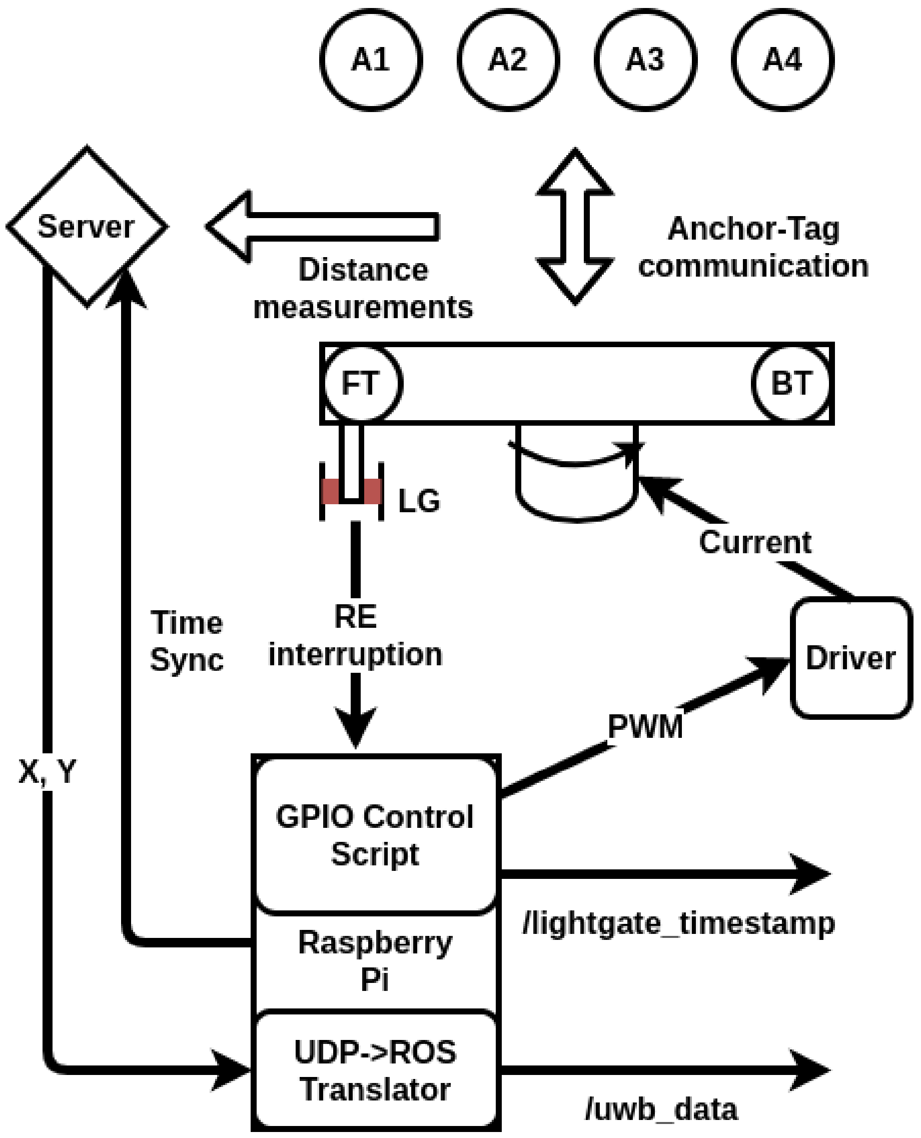


**Figure 4.3.** Platform software setup.

## 4.2 Experiments

In this section, the experiments conducted on the platform with UWB tags designed in Section 4.1 are described. The resulting measurements are then statistically evaluated to obtain the estimate of the UWB localization system accuracy. It is worth noting that the noise parameters are evaluated while assuming that they follow a 2D Gaussian distribution. The approximation with a 2D Gaussian is supported by Figures 4.4, 4.5, 4.6, and Table 4.1.

During the experiments, the UWB anchors were installed on the ceiling of the room's corner. The 2D dimensions of the room itself are about $6 \times 2.7$ m. The coordinate system was chosen so that the x-axis was aligned with the longer wall, y-axis with the shorter wall, and the origin was chosen as the position of a room's corner. Each experiment was also conducted so that the clear LOS among the tag and all four anchors existed.

The 2D Gaussian approximation claim is supported by a set of measurements in Table 4.1. During these measurements, the platform's position is fixed, and the localization system's built-in KF setting is varying. The KF offers up to 9 modes of IMU and 2D measurement sensor fusion. Each mode defines different weights, and with increasing mode, the IMU is more favored than the measurement. The approximation is tested based on three confidence levels (CL), commonly known as the 68-95-99.7 rule. Three confidence ellipses were drawn, and the number of inliers among the data was calculated. The mean of ellipses corresponds to the mean of 2D Gaussian approximation, while the semiaxes are defined in Equation (4.4).

$$a_i = v_i \sqrt{\chi_2^2(\alpha)\lambda_i}, \tag{4.4}$$

where

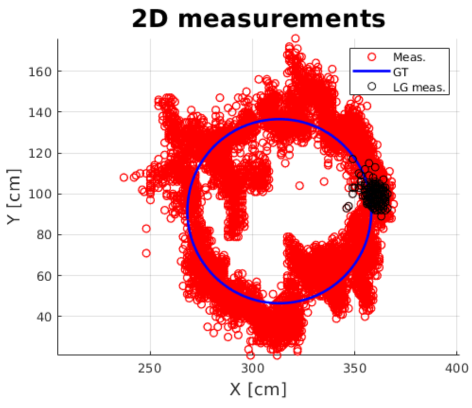
- $a_i$ are the semiaxes defining the ellipse,
- $\chi_2^2$ is the $\chi^2$-distribution with two degrees of freedom,
- $\alpha$ is the probability that the random vector will be outside the ellipse,
- $\lambda_i$ are the eigenvalues of covariance matrix of 2D Gaussian distribution,
- $v_i$ are the unit eigenvectors corresponding to $\lambda_i$.

Three representative measurements were chosen (KF mode 0, KF mode 3, and KF mode 7) to show the actual noise distribution compared to the approximation with the 2D Gaussian, and can be seen in Figures 4.4, 4.5, and 4.6[1]. Additionally, a histogram of the absolute measurement error is shown. Based on the Figures 4.4, 4.5, 4.6, and Table 4.1, the approximation by the 2D Gaussian PDF was concluded to be sufficient.

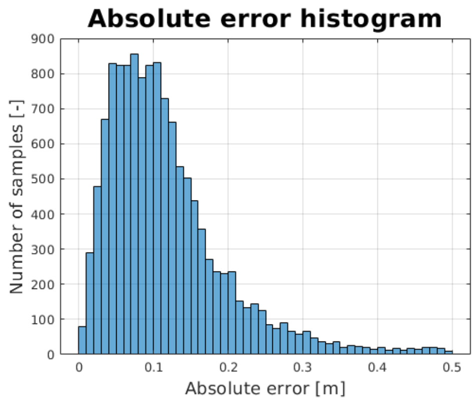
[1] `Meas.` stands for measurements, `GT` stands for ground truth, and `LG` stands for light gate.

| Measurements | CL1 samples [%] | CL2 samples [%] | CL3 samples [%] |
|---|---|---|---|
| Normal distribution baseline | 68.269 | 95.450 | 99.730 |
| $KF_0$ | 70.899 | 92.784 | 99.395 |
| $KF_1$ | 68.994 | 94.658 | 99.715 |
| $KF_2$ | 69.490 | 94.235 | 99.763 |
| $KF_3$ | 72.075 | 93.662 | 99.695 |
| $KF_4$ | 70.667 | 94.031 | 99.798 |
| $KF_5$ | 68.621 | 94.883 | 99.810 |
| $KF_6$ | 69.052 | 95.053 | 99.901 |
| $KF_7$ | 71.633 | 93.933 | 99.795 |
| $KF_8$ | 69.330 | 94.762 | 99.846 |

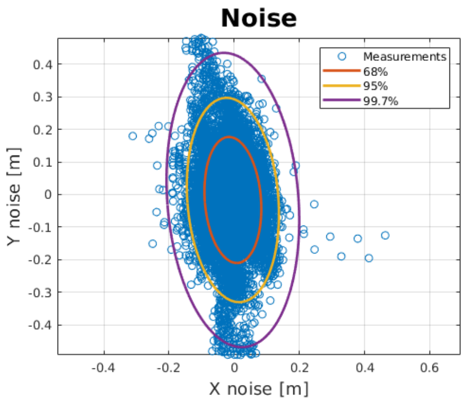
**Table 4.1.** First set of UWB measurements, examining the 2D Gaussian approximation.



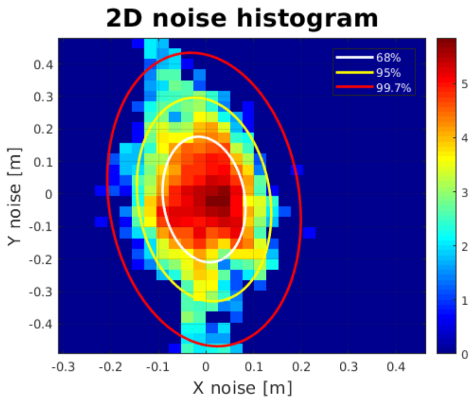
Measurement data against ground truth.



The absolute error histogram.

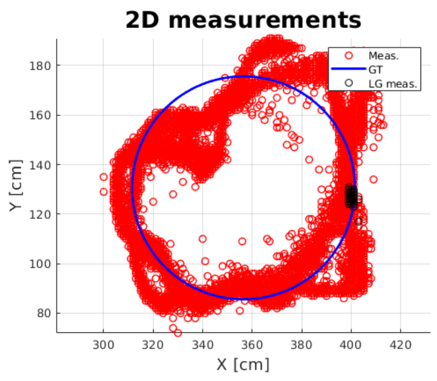


Noise against PDF approximation.

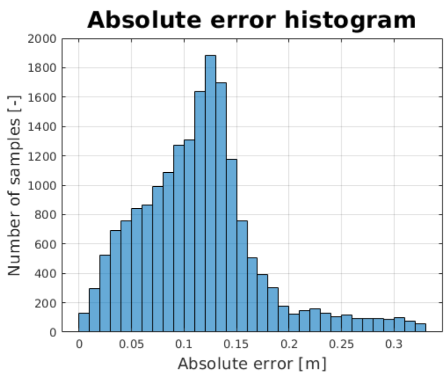


2D histogram of noise.

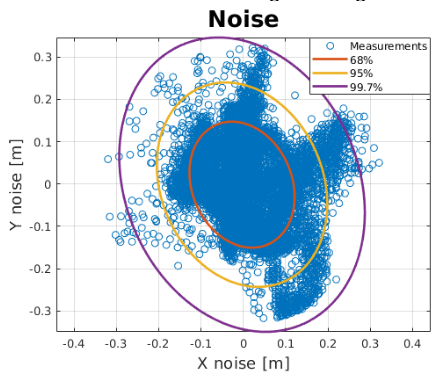
**Figure 4.4.** Demonstration of measurement without KF processing.



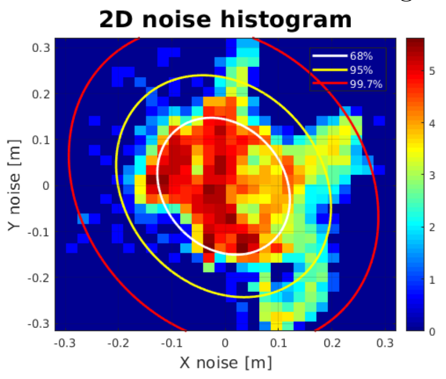
Measurement data against ground truth.



The absolute error histogram.

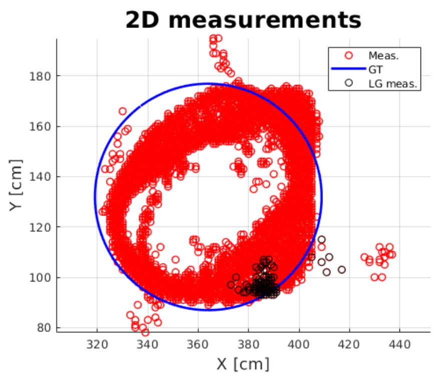


Noise against PDF approximation.

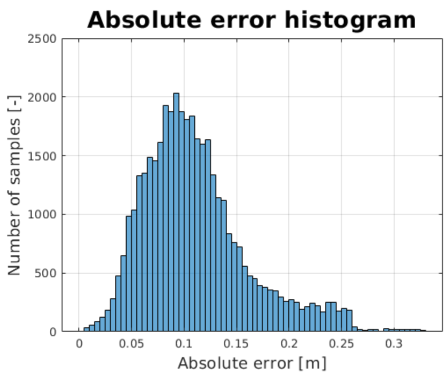


2D histogram of noise.

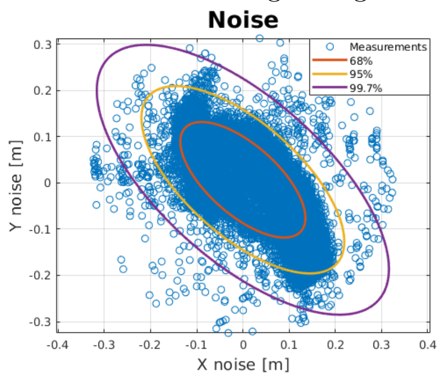
**Figure 4.5.** Demonstration of measurement with KF mode 3.



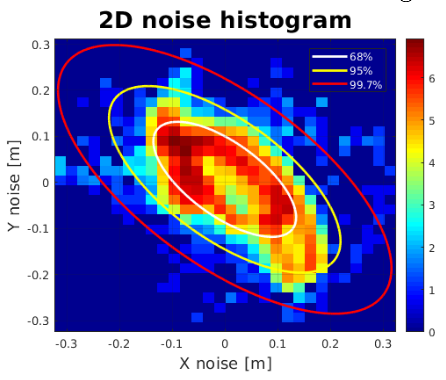
Measurement data against ground truth.



The absolute error histogram.



Noise against PDF approximation.



2D histogram of noise.

**Figure 4.6.** Demonstration of measurement with KF mode 7.

As described in the Section 3.3.2, the UWB localization system provides data processing using a KF. Because of that, the following measurements are divided into several groups. In each group, the influence of a single parameter is evaluated, while the other parameters stay the same.

The first scenario is designed to evaluate the accuracy of the localization system with different KF mode settings. The experiments are conducted with a fixed position of the platform.

| Meas. | $\mu_x$ [m] | $\mu_y$ [m] | $\sigma_x^2$ [m$^2$] | $\sigma_{xy}$ [m$^2$] | $\sigma_y^2$ [m$^2$] | $\mu_e$ [m] | $max_e$ [m] |
|---|---|---|---|---|---|---|---|
| KF$_0$ | -0.0020 | 0.0088 | 0.0071 | -0.0006 | 0.0168 | 0.1351 | 0.3843 |
| KF$_1$ | -0.0063 | 0.0025 | 0.0088 | -0.0027 | 0.0130 | 0.1312 | 0.4109 |
| KF$_2$ | -0.0024 | 0.0097 | 0.0072 | -0.0043 | 0.0102 | 0.1189 | 0.3433 |
| KF$_3$ | -0.0055 | 0.0047 | 0.0080 | -0.0039 | 0.0077 | 0.1139 | 0.3056 |
| KF$_4$ | -0.0016 | 0.0047 | 0.0055 | -0.0029 | 0.0055 | 0.0961 | 0.2429 |
| KF$_5$ | -0.0018 | 0.0057 | 0.0069 | -0.0043 | 0.0070 | 0.1090 | 0.2479 |
| KF$_6$ | 0.0002 | -0.0005 | 0.0066 | -0.0040 | 0.0062 | 0.1028 | 0.2487 |
| KF$_7$ | -0.0002 | 0.0067 | 0.0080 | -0.0049 | 0.0068 | 0.1112 | 0.3285 |
| KF$_8$ | -0.0031 | 0.0056 | 0.0122 | -0.0076 | 0.0110 | 0.1409 | 0.2855 |

**Table 4.2.** First set of UWB measurements, examining the influence of the KF mode.

Another set of measurements was conducted with the platform placed in a different position.

| Meas. | $\mu_x$ [m] | $\mu_y$ [m] | $\sigma_x^2$ [m$^2$] | $\sigma_{xy}$ [m$^2$] | $\sigma_y^2$ [m$^2$] | $\mu_e$ [m] | $max_e$ [m] |
|---|---|---|---|---|---|---|---|
| KF$_0$ | -0.0070 | 0.0071 | 0.0093 | -0.0007 | 0.0198 | 0.1478 | 0.4370 |
| KF$_1$ | -0.0041 | 0.0032 | 0.0073 | -0.0018 | 0.0120 | 0.1241 | 0.3394 |
| KF$_2$ | -0.0030 | 0.0005 | 0.0074 | -0.0020 | 0.0109 | 0.1216 | 0.3174 |
| KF$_3$ | -0.0032 | -0.0015 | 0.0068 | -0.0016 | 0.0097 | 0.1146 | 0.3269 |
| KF$_4$ | -0.0011 | 0.0021 | 0.0063 | -0.0016 | 0.0087 | 0.1093 | 0.3081 |
| KF$_5$ | -0.0044 | -0.0012 | 0.0064 | -0.0021 | 0.0089 | 0.1118 | 0.2887 |
| KF$_6$ | 0.0007 | -0.0039 | 0.0054 | -0.0020 | 0.0076 | 0.1034 | 0.2603 |
| KF$_7$ | -0.0014 | -0.0025 | 0.0054 | -0.0016 | 0.0072 | 0.1006 | 0.2817 |
| KF$_8$ | -0.0022 | -0.0011 | 0.0054 | -0.0018 | 0.0069 | 0.1003 | 0.2707 |

**Table 4.3.** Second set of UWB measurements, examining the influence of the KF mode.

By observing the mean and maximum errors from Tables 4.2 and 4.3, it can be seen that the KF provided by the UWB localization system manufacturer greatly influences the data accuracy. Both measurement sets indicate that the raw data (KF mode 0) shows nearly no correlation between x and y measurements. By employing the KF, the x, y measurements tend to be negatively correlated, meaning that, e.g., the large error in x results in a small error in y. It can also be observed that the variance on the y is significantly higher than the variance on the x. On the other hand, KF processing seems to reduce the difference between variances on both x, y. Finally, a large correlation discrepancies of same KF modes can be observed between measurement sets.

Another set of measurements was proposed to study the platform position influence on the covariance matrix, and the results are shown in Table 4.4. As the correlation of raw measurements seems to be practically zero, to observe an influence of the position on the data correlation, the KF mode was set to 3 for this set of measurements.

| Meas. | pos$_x$ [m] | pos$_y$ [m] | $\sigma_x^2$ [m$^2$] | $\sigma_{xy}$ [m$^2$] | $\sigma_y^2$ [m$^2$] | $\mu_e$ [m] | $max_e$ [m] |
|---|---|---|---|---|---|---|---|
| Pos. A | 3.5690 | 1.2820 | 0.0059 | -0.0022 | 0.0109 | 0.1148 | 0.3396 |
| Pos. B | 3.3920 | 1.0380 | 0.0033 | -0.0012 | 0.0150 | 0.1216 | 0.3289 |
| Pos. C | 3.5510 | 1.5480 | 0.0070 | -0.0031 | 0.0147 | 0.1315 | 0.2910 |
| Pos. D | 4.9570 | 1.0780 | 0.0121 | -0.0067 | 0.0175 | 0.1557 | 0.4108 |
| Pos. E | 3.4190 | 0.7080 | 0.0033 | -0.0001 | 0.0121 | 0.1098 | 0.2722 |
| Pos. F | 3.5140 | 1.1630 | 0.0034 | -0.0034 | 0.0187 | 0.1328 | 0.3864 |

**Table 4.4.** Third set of UWB measurements, examining the platform position influence.

As Table 4.4 indicates, the platform position has a high impact on localization characteristics. However, the cause of this impact does not seem to have an obvious relation to the center of the room, located at $x \approx 3$ m and $y \approx 1.35$ m. It is worth noting that due to the furniture and corridor, the platform could not be placed arbitrarily in space, and the lack of measurements from the other half of the room might have brought additional information. Thus, it was concluded that the influence is due to the environment, although no particular cause was determined.

Two additional raw measurements (i.e., with KF mode set to zero) were conducted as the raw measurements are less correlated then processed measurements. The measurements were taken at random positions, and the results can be seen in Table 4.5.

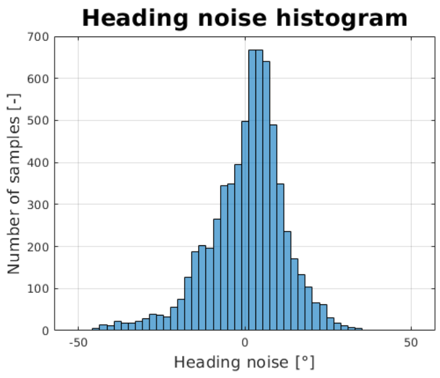
| Meas. | $\mu_x$ [m] | $\mu_y$ [m] | $\sigma_x^2$ [m$^2$] | $\sigma_{xy}$ [m$^2$] | $\sigma_y^2$ [m$^2$] | $\mu_e$ [m] | $max_e$ [m] |
|---|---|---|---|---|---|---|---|
| Pos I. | -0.0105 | -0.0118 | 0.0058 | -0.0016 | 0.0171 | 0.1264 | 0.4877 |
| Pos II. | -0.0040 | -0.0167 | 0.0033 | -0.0010 | 0.0164 | 0.1168 | 0.4933 |

**Table 4.5.** Pair of raw measurements at random positions.

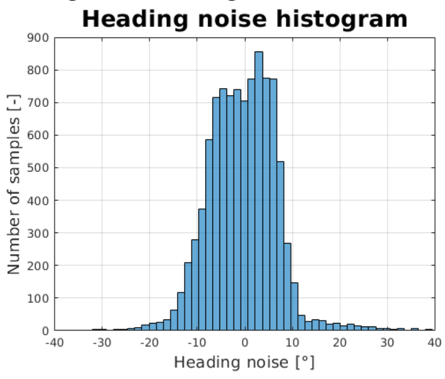
As discussed in Section 3.3.1, the two-tag heading estimation, without correlating it with x, y estimates, is unreliable. A final set of measurements was conducted at a fixed position with different KF settings to support this claim, examining the heading calculation.

| Meas. | $\mu$ [rad] | $\sigma^2$ [rad$^2$] | $3\sigma$ [rad] |
|---|---|---|---|
| KF$_0$ | 0.0129 | 0.0454 | 0.6391 |
| KF$_1$ | -0.0223 | 0.0195 | 0.4192 |
| KF$_2$ | -0.0107 | 0.0210 | 0.4345 |
| KF$_3$ | -0.0037 | 0.0201 | 0.4255 |
| KF$_4$ | -0.0029 | 0.0182 | 0.4052 |
| KF$_5$ | 0.0172 | 0.0232 | 0.4566 |
| KF$_6$ | -0.0040 | 0.0136 | 0.3499 |
| KF$_7$ | 0.0082 | 0.0110 | 0.3148 |
| KF$_8$ | 0.0244 | 0.0141 | 0.3564 |

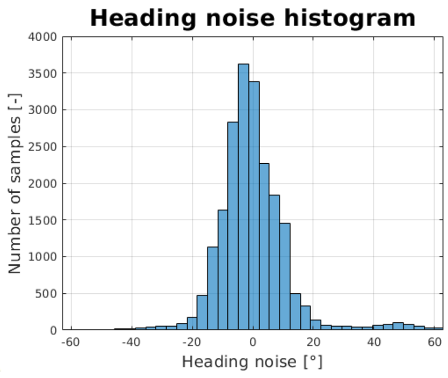
**Table 4.6.** Calculation of heading from two UWB tags measurements.



Heading noise histogram for KF mode 0.



Heading noise histogram for KF mode 3.



Heading noise histogram for KF mode 7.

**Figure 4.7.** Demonstration of UWB heading calculations.

From Table 4.6 and Figure 4.7, it can be seen that although the heading calculation follows a normal distribution, its variance is way too high for reliable heading estimation. Because of that, it was decided that this estimation will not be used in simulations. Instead, a gyroscope heading rate[1] that provides a reliable heading estimate on a short time horizon is used in the experiments as the sole source of the heading information. It is worth noting that if longer experiments were designed, the estimate from the gyroscope heading rate would suffer from increased drift.

### Conclusion

The UWB localization system measurement noise is not a perfect 2D Gaussian. However, based on the Figures 4.4, 4.5, and 4.6, along with Table 4.1, it was concluded that the 2D Gaussian approximation is sufficient. Properties of the approximation were then estimated. Throughout all discussed measurements, the absolute values of means for both variables were smaller than 2 cm, which was determined to be sufficient to declare the approximation to be white Gaussian noise. Due to the environmental influence on the correlation of the measurements, and the fact that the MRS state fusion algorithm assumes uncorrelated measurement noise, the tag model measurement noise was chosen to be uncorrelated. However, the measurements indicate that the uncorrelated assumption is not entirely true even for raw measurements. Two UWB tag models were considered, both assuming a white uncorrelated Gaussian noise. A more optimistic model assumes that the variances $\sigma^2$ are $\sigma_x^2 = \sigma_y^2 = 0.01778\ \mathrm{m}^2$, while the pessimistic one assumes $\sigma_x^2 = \sigma_y^2 = 0.02778\ \mathrm{m}^2$. It is worth noting that the measurements were obtained by employing the TWR-ToF technology that has a significant time delay ($\approx\ 200 - 300$ ms). Based on the information from the manufacturer, the TDoA technology provides the measurements of the same accuracy with a time delay of $\approx\ 40$ ms, and this value was used for the UWB tag simulation model. The support with the TDoA setup of the UWB localization system was promised. However, due to the pandemic situation, the support was not received. Finally, the uncorrelated heading calculation was demonstrated to support the claim that this simple approach is not reliable for heading estimation.

[1] Which would be used along with the UWB heading estimation in the heading KF.

# Chapter 5 Implementation

The first part of this chapter is dedicated to the simulation model of the UWB tag. In the second part, the implementation required for integrating the external UWB localization system into the MRS framework is described, along with additional software developed for conducting the experiments.

## 5.1 UWB Tag Simulation Model

Since the system provided by the MRS does not include support for UWB technology, the design of the UWB tag models for the simulation experiments was necessary. One of the possible approaches is to model the whole UWB localization system with all its components, i.e., the anchors and the tags, and apply one of the localization techniques described in Section 2.1.4 on the transmitted signals. However, this model would require complicated and expensive simulation tools for wave propagation in the environment. Because of that, another more straightforward approach was chosen.

The *gazebo_uwb_plugin* was developed to simulate the accuracy of the UWB localization system output estimated from the experiments in Section 4.2. Apart from the noise parameters, the time delay, frequency, ID, and position offsets in each axis can be specified to provide more trustworthy behavior. The time delay of the measurements is implemented as the FIFO priority queue, into which the ground truth positions are stored. After a time delay period passes, the queue starts popping the stored positions. The generated Gaussian noise is added to each position, and the result is converted into a ROS message propagated further into the system.

The interaction of the queue and message publishing is implemented as a solution to the conditioned producer-consumer problem. The queue acts as the producer, and each time a new data is stored into the queue, the process responsible for adding the noise and publishing the message is notified, acting as the consumer.

Two models representing the tags were then placed on the base of the Tarot 650 Sport UAV model, which is part of the MRS framework, based on their specified position offsets using the xacro format.

## 5.2 Software Integration

This section describes the software solution necessary for the proposed architecture realization. The first part describes the integration of the UWB localization into the MRS framework. The second part briefly introduces the package used for path planning. In the third and fourth part, the newly designed mission and failsafe managers responsible for autonomous execution of the mission, fault detections and their handling, is introduced. In the fifth and final part, the model of the UWB tag used in simulations is described.

### 5.2.1 UWB Localization Integration

The output of the UWB tag drivers is transmitted as a newly defined ROS message *UwbTagStamped*, providing mainly the information about the tag ID, tag 2D position, and timestamp of the measurement. The information about the tag ID is necessary to include because all tags are transmitting on the same topic */uwb_data_raw*.

The UAV position in the plane is estimated from the positions of both tags on its base. However, in general, neither tag is located in the UAV's body mass center. Thus, these tag measurements (T) have to be transformed to obtain the 2D position estimate of the UAV's body mass center. This transformation is based on the 3D rotation from the UAV frame to the inertial frame, which can be seen in Equation (5.1).

$$\boldsymbol{T} = \begin{bmatrix} \mathrm{c}(\psi)\mathrm{c}(\theta) & \mathrm{c}(\psi)\mathrm{s}(\theta)\mathrm{s}(\phi) - \mathrm{s}(\psi)\mathrm{c}(\phi) & \mathrm{c}(\psi)\mathrm{s}(\theta)\mathrm{c}(\phi) + \mathrm{s}(\psi)\mathrm{s}(\phi) \\ \mathrm{s}(\psi)\mathrm{c}(\theta) & \mathrm{s}(\psi)\mathrm{s}(\theta)\mathrm{s}(\phi) + \mathrm{c}(\psi)\mathrm{c}(\phi) & \mathrm{s}(\psi)\mathrm{s}(\theta)\mathrm{c}(\phi) - \mathrm{c}(\psi)\mathrm{s}(\phi) \end{bmatrix}, \tag{5.1}$$

where

- $\psi$ is the *yaw* angle [rad],
- $\theta$ is the *pitch* angle [rad],
- $\phi$ is the *roll* angle [rad].

Evaluating this rotation matrix at each measurement time sample $k$, along with a fixed relative position of the tag with respect to the UAV's body mass center, can be used to transform the measurement into the position estimate of the UAV's body mass center. This transformation can be seen in Equation (5.2).

$$\begin{bmatrix} x(k) \\ y(k) \end{bmatrix} = \begin{bmatrix} x_m(k) \\ y_m(k) \end{bmatrix} - \boldsymbol{T}(k) \begin{bmatrix} x_O \\ y_O \\ z_O \end{bmatrix}, \tag{5.2}$$

where

- $x, y$ are the calculated $x, y$ positions of the UAV's body mass center [m],
- $x_m, y_m$ are the $x, y$ position measurements [m],
- $x_O, y_O, z_O$ are the tag $x, y, z$ offsets from the UAV's body mass center [m].

After the UAV's 2D position is calculated, a simple check is used if a new position does not differ from previous by more then defined threshold. If yes, the position change is saturated.

Finally, the 2D KF correction step is invoked. As described in Section 2.2, a measurement covariance matrix $\boldsymbol{R}$ is used during a correction step. Unlike the process covariance matrix $\boldsymbol{Q}$ used in the prediction step, which does not depend on the sensor from which the measurement is obtained, the $\boldsymbol{R}$ representing the measurement covariance matrix must be specified[1] for the UWB position sensor separately.

The whole process of data propagation can be seen in Figure 5.1.

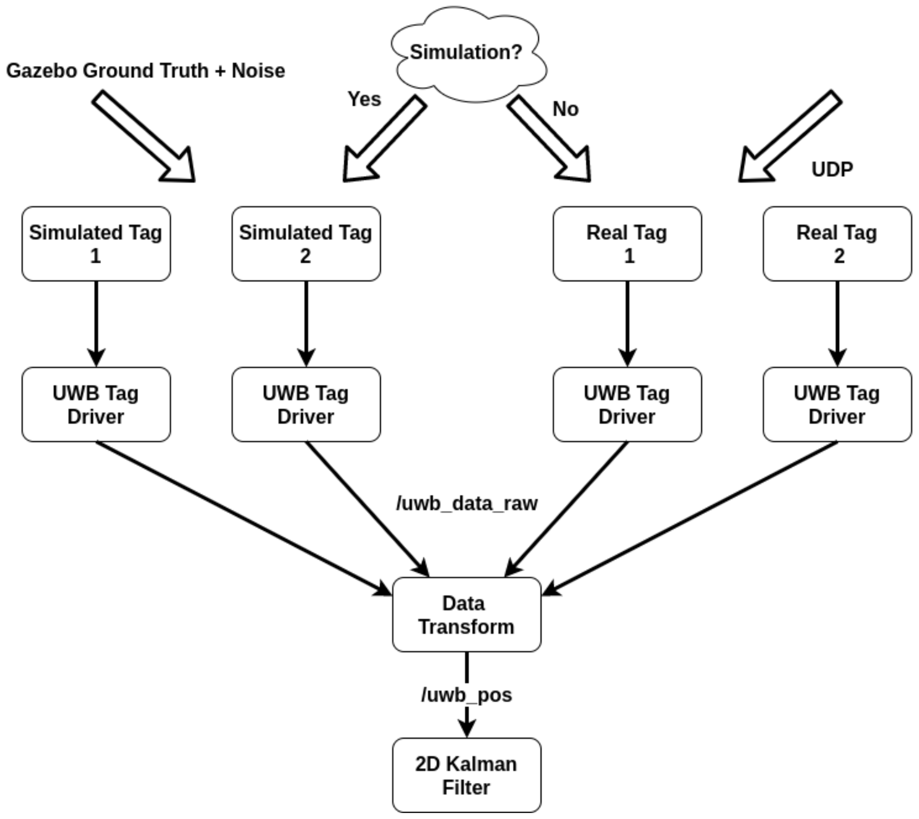


**Figure 5.1.** UWB data propagation process.

### 5.2.2 Navigation

For the simulation and application goals, it is necessary that the UAV can properly navigate itself in the environment. For the sake of this thesis, a known static environment is assumed. Thus, the mapping of the environment is not needed, as the map can be obtained externally. For this end, the *map_server* package, part of the ROS navigation stack [30], was chosen. This package converts a given image representing

[1] This matrix is typically obtained from static measurement evaluation.

the environment's occupancy data into a representation compatible with the ROS. The map's metadata, such as the resolution of the map, its origin, and threshold for determining if each cell is an obstacle or a free space, must be specified. In Figure 5.2, the simulation environment in Gazebo can be seen along with its image representation.

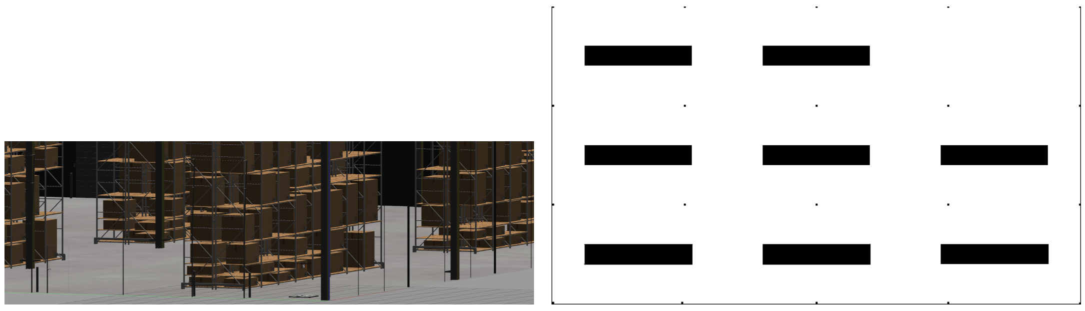

Gazebo world. The 2D image representation.

**Figure 5.2.** Illustration of the simulation environment.

Once the map is available, the path planning algorithm must be employed. The *global_planner* that is also part of the navigation stack was chosen to be the most suitable as it is already compatible with the map representation of the environment. In its basic form, the package offers two planning algorithms, the *A** and the *Dijkstra's* algorithms. The difference between these algorithms is that the *A** is an informed algorithm, and as such, it is a faster option to use if an admissible heuristics can be defined. The *Dijkstra's* algorithm, on the other hand, is a better choice provided that the complete information about the map cannot be obtained. For this reason, the *A** algorithm was chosen due to the static known environment assumption.

The *global_planner* package provides a service to compute a plan from point A to point B. However, for autonomous UAV warehouse inventorying, it is necessary to fly through several waypoints[1] in a specific order rather than flying from only one position to another. Because of that, an algorithm for automatic calling of the planning service *make_multiple_plans* was integrated into the package to solve this issue.

The algorithm assumes a text file as an input, specifying the sequence of at least two waypoints as the 3D position and heading. The algorithm simply calls the planning service between two consecutive waypoints, and if the plan is feasible, then iterates until the end of the sequence. Each trajectory is saved into a separate text file.

A part of the *global_planner* package is also an orientation filter. This filter offers seven modes, allowing post-processing of the planned trajectory and adding the orientation to each trajectory point. For this thesis, none of these modes was found suitable. Because of that, an extra mode was implemented, adding the goal orientation to each point along the trajectory. This way, the UAV is forced to the goal orientation as soon as possible, which is vital for proper spatial coverage planning.

The planned trajectories are unnecessary dense, and thus a *euclidean_distance_threshold*

[1] The algorithm for generating these waypoints was not developed as a part of this thesis, but the underlying idea is mentioned in 7.1.

can be specified to skip points very close to already defined point. However, the specified threshold must not be too high. Otherwise, the risk of cutting corners planned around obstacles is introduced.

The planning process is illustrated in Figure 5.3.

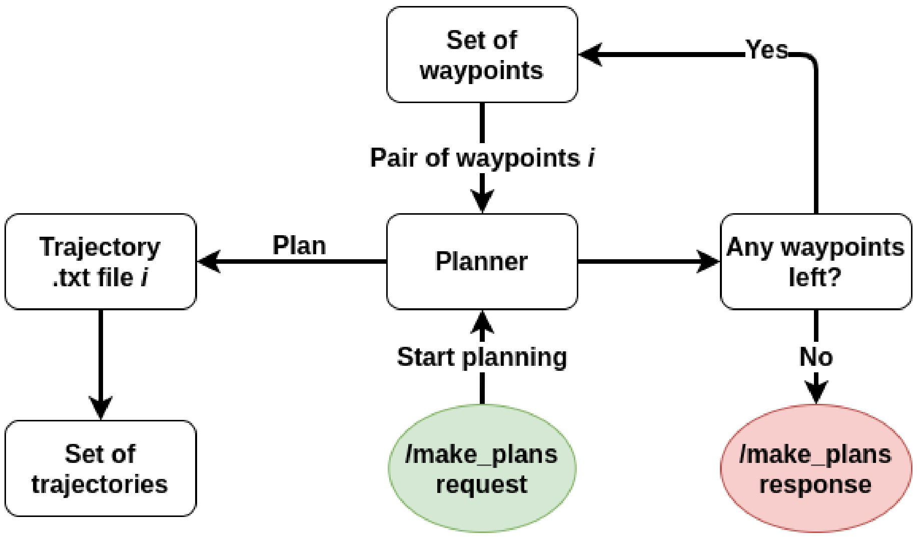


**Figure 5.3.** Planning process.

As described above, the trajectories planned with the *global_planner* package are stored in the form of several text files. For the loading and immediate following of these trajectories, a service *start_following_trajectories_from_txt_files* was designed. The call of this service creates a goal for the action server that implements the loading algorithm. The algorithm iterates over all trajectory files, and, assuming that they are not corrupted, it converts the data from each file into MRS compatible format and calls MRS service to set a single trajectory. The iteration is then put on hold until the current trajectory following is finished.

Another service, *cancel_loading_trajectories_from_txt_files*, can be called to cancel the current trajectory following. While it internally calls the MRS service that cancels the trajectory following, it also aborts the execution of the loading action, preventing an accidental loading of the next trajectory. The trajectory loading process can be seen in Figure 5.4.

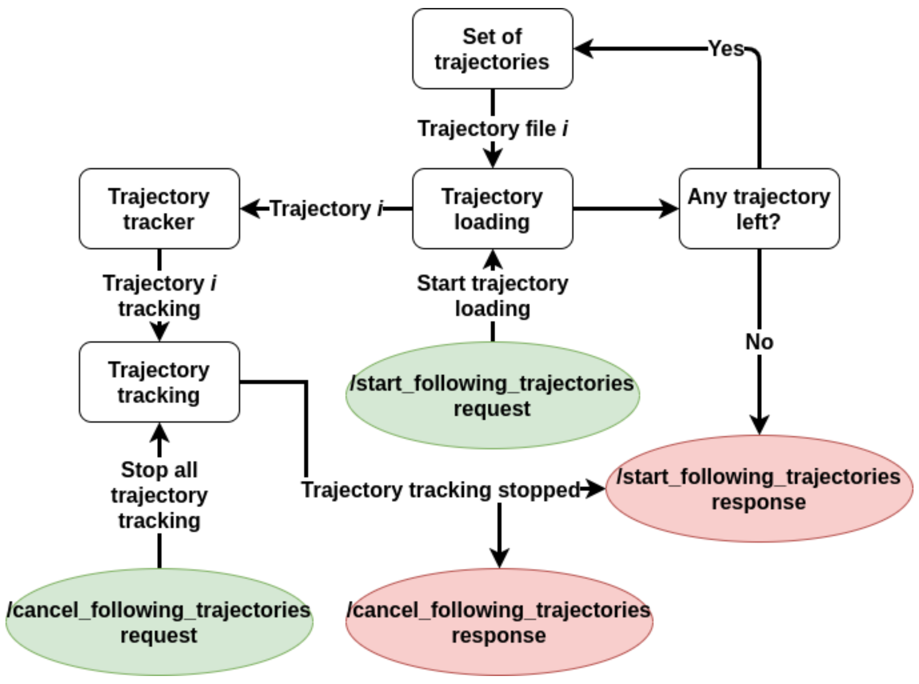

Set of trajectories
Yes
Trajectory file *i*
Trajectory tracker
Trajectory *i*
Trajectory loading
Any trajectory left?
Trajectory *i* tracking
Start trajectory loading
No
Trajectory tracking
/start_following_trajectories request
Stop all trajectory tracking
Trajectory tracking stopped
/start_following_trajectories response
/cancel_following_trajectories request
/cancel_following_trajectories response

**Figure 5.4.** Trajectory planning process.

### 5.2.3 Mission Manager

An essential part of the UAV system's autonomy is the ability to switch between different mission phases while fulfilling the application for which it was designed. For this purpose, the *mission_manager* node was developed. The *mission_manager* defines a finite state machine with six possible states and six possible events responsible for the transition between the states. The diagram illustrating the state machine can be seen in Figure 5.5. The following set of states is defined:

- **DISARM** - The UAV is not flying, its motors are turned off, and it is disarmed.
- **TAKE OFF** - The UAV is armed and is taking off to the desired altitude.
- **MISSION** - The UAV is currently fulfilling the defined mission (e.g., following the set of trajectories).
- **RETURN HOME** - The UAV is returning to the defined altitude above its home position.
- **LANDING** - The UAV is landing on the defined position.
- **EMERGENCY LANDING** - The UAV is landing as soon as possible as some error occurred.

For the transition between states above, the following set of transition events is defined:

- **NO EVENT** - No other event was registered, the UAV should stay in the current state.
- **ACTION COMPLETE** - Event signalizing that the current state action (e.g., desired altitude reached during the **TAKE OFF**) is fulfilled.
- **BATTERY CRITICAL** - Error event signalizing that the battery reached the critical threshold.

- **BATTERY WARNING** - Warning event signalizing that the battery is estimated to reach a defined threshold if immediate **RETURN HOME** action is invoked.
- **COMMUNICATION LOST** - Warning event signalizing that the communication with a base hub is lost, no data can be transmitted between the base hub and the UAV.
- **UWB NOT RELIABLE** - Error event signalizing that the UWB localization system is transmitting bad data or no data at all.

In case when multiple events are registered at the same time, the *mission_manager* prioritizes error events, then warning events and only afterward the action complete event.

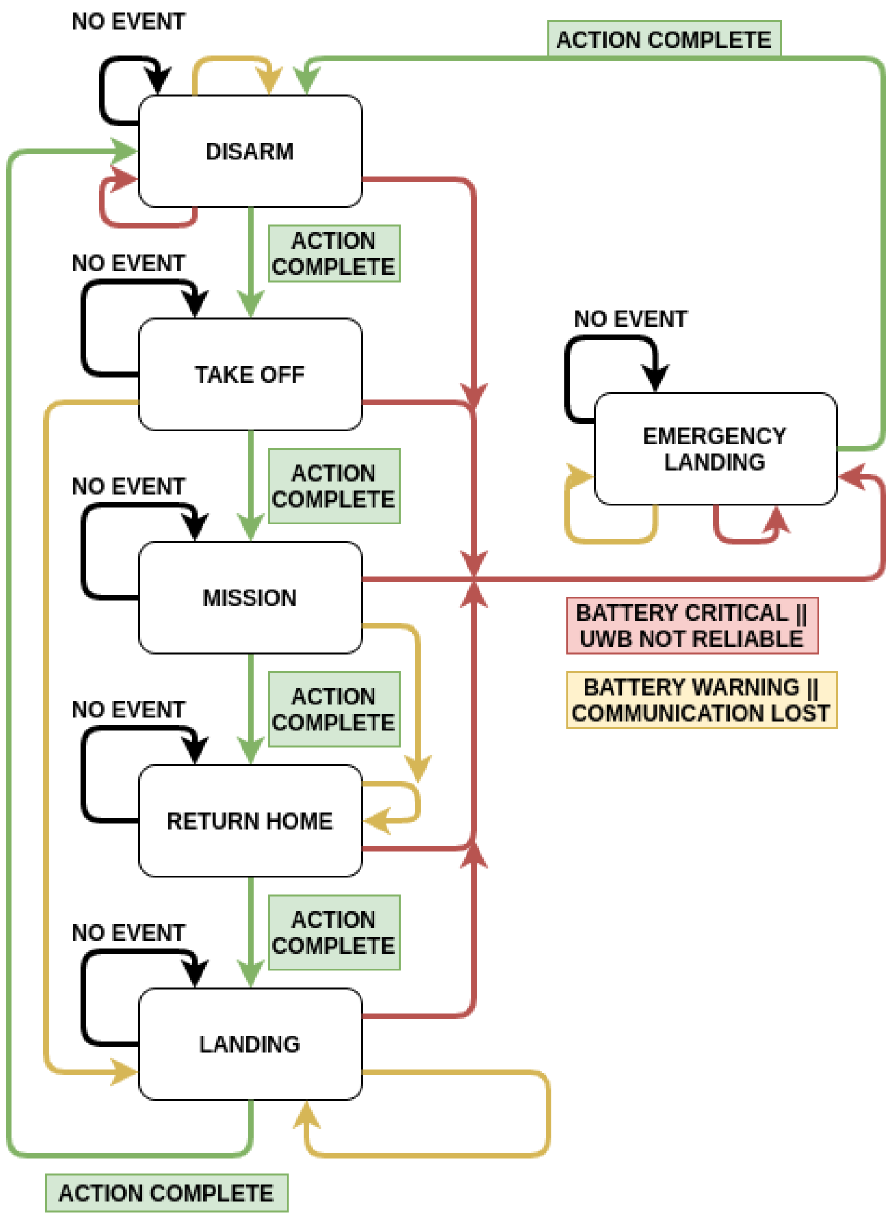


**Figure 5.5.** The mission manager state machine diagram.[1].

[1] For clarity, **NO EVENT** is colored black, warning events yellow, error events red and **ACTION COMPLETE** event in green.

### 5.2.4 Failsafe Manager

Another ROS node, the *failsafe_manager*, was created to generate the information necessary for correct registration of warning and error events in the *mission_manager*.

For registration of a **BATTERY WARNING** event, an algorithm estimating the battery level if the UAV would immediately return home is implemented. An algorithm assumes that the battery is modeled as follows:

$$b_L(t) = Ae^{Bt} + Ce^{Dt}, \tag{5.3}$$

where

- $b_L$ is the estimated battery level in range $[0, 1]$ [-],
- $A, B, C, D$ are battery model coefficients [-],
- $t$ is the time at which the battery level is estimated [s],
- $e$ is the Euler number.

The time of the battery level estimation is determined by calculating the Manhattan distance[1] from the UAV's current position to its home position, taking into account the maximum allowed UAV's velocity. Additionally, a *speed_factor* parameter can be specified to force the estimate to be more or less strict by multiplying the maximum allowed velocity.

If a battery level estimated this way is lower than specified *battery_warning_threshold* parameter, the information about battery warning is offered for processing to the *mission_manager*.

Additionally, a check if a current battery level reaches the *battery_critical_threshold* is implemented, and the information is propagated to the *mission_manager*.

The UWB localization system's reliability evaluation is implemented as a simple check whether the new messages are received or not (watchdog). For the sake of this thesis, and demonstration of the *mission_manager* reaction, it is considered to be sufficient. However, for a real application, the uncertainty of the measurements should be considered as well.

The ability to communicate with the base hub can be checked by sending a heartbeat signal, i.e., a constantly repeating message that, if received, is acknowledged.

The *failsafe_manager* provides several service servers that can be utilized and are used in the *mission_manager*. These are *land* and *emergency_land* services, which internally only call the MRS services responsible for landing and emergency landing,

[1] $A*$ could be used instead, but it was decided that in a large distance from the home position, the planning could take a considerably longer time.

respectively, or *disarm*, which calls the Pixhawk autopilot disarm service. The *return_home* is responsible for calling the action server, implementing the algorithm for the return home process. The UAV's movement is at first canceled. Once the UAV is hovering, the *global_planner* planning service between the current steady position and home position is called to obtain a trajectory for returning home. A simple state machine is designed for the action's feedback.

# Chapter 6
# Experiments

This chapter is dedicated to the experimental evaluation of the UWB localization system integration with the MRS platform in simulation. Unfortunately, due to unforeseen complications caused by the pandemic situation, the localization system's deployment on the real UAV and experimental evaluation in the real-world scenario was not done as the premises chosen for its conducting were not accessible. The experiments were conducted under the assumption that the UWB localization system is the sole provider of the position measurements.

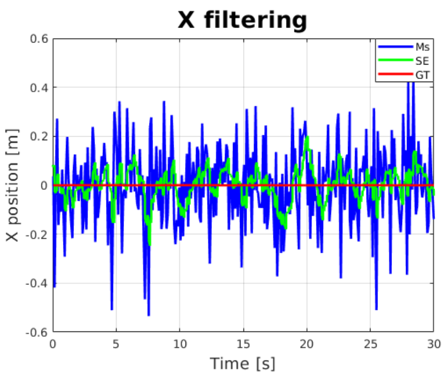
## 6.1 Simulation Experiments

In this section, the results of the conducted simulation experiments are demonstrated. As discussed in Section 4.2, two UWB tag models are considered that differ in the accuracy. Throughout this section, the tag model with $\sigma_x^2 = \sigma_y^2 = 0.02778 \text{ m}^2$ will be referred to as the tag model A, while the tag model with $\sigma_x^2 = \sigma_y^2 = 0.01778 \text{ m}^2$ one as the tag model B. Four experimental scenarios were considered: a state estimation while the UAV is static and on the ground, following a straight-line trajectory, following a rectangle trajectory, and following a circular trajectory. During the experiment, the UAV is controlled based on the state estimate from the UWB. Because of that, a difference between the planned trajectory and the true trajectory can be observed. Figures 6.1-6.8 show the comparison of the state estimation against the ground truth, as well as the histogram of state estimation error[1].

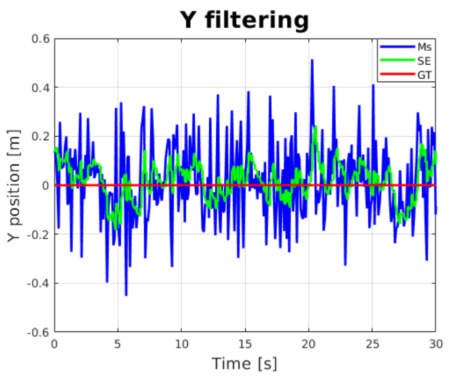
The following abbreviations are used in Figures:

- GT $\sim$ ground truth from the simulation,
- SE $\sim$ state estimator's position estimate,
- Ms $\sim$ measurements,
- Plan $\sim$ planned trajectory,
- RH $\sim$ return home.

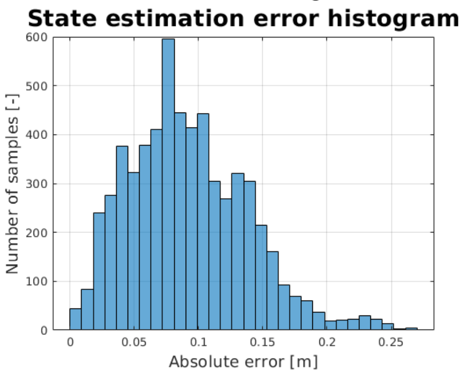
[1] An absolute 2D Euclidean distance between ground truth and position estimate.



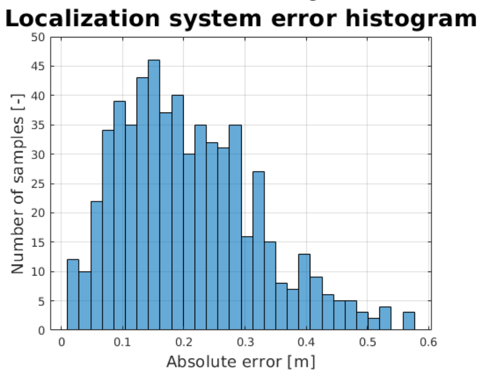
X data filtering.



Y data filtering.

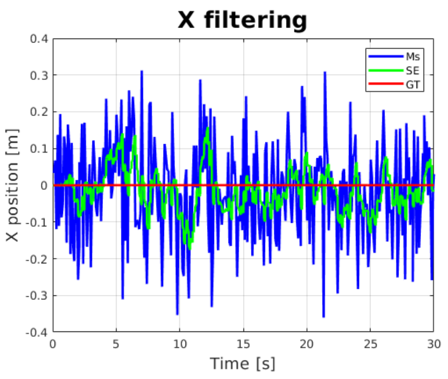


SE error histogram.

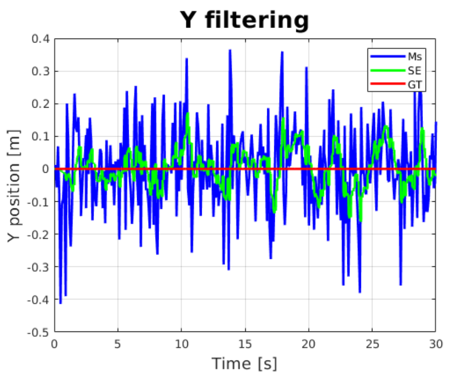


Measurements error histogram.

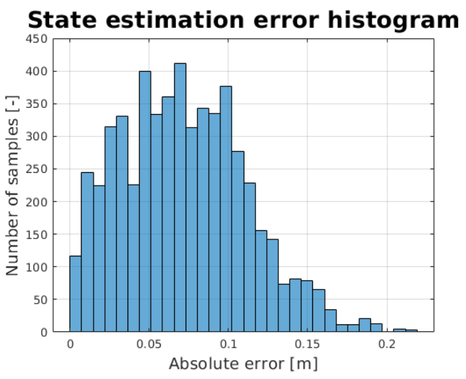
**Figure 6.1.** Demonstration of static measurement with tag model A.



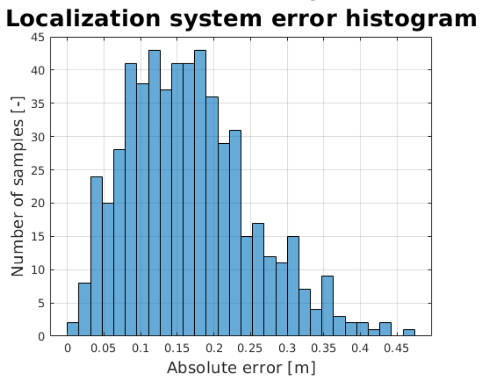
X data filtering.



Y data filtering.

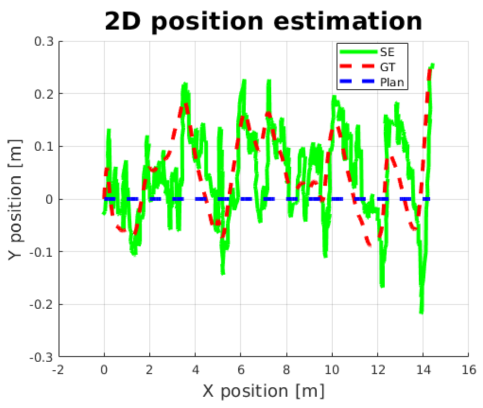


SE error histogram.

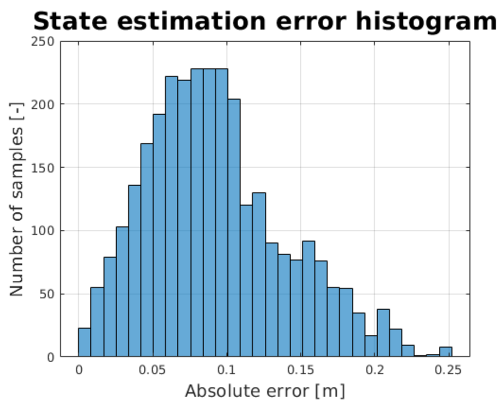


Measurements error histogram.

**Figure 6.2.** Demonstration of static measurement with tag model B.

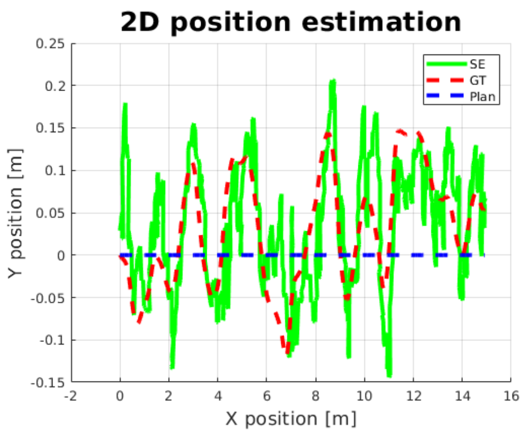


Position estimation.

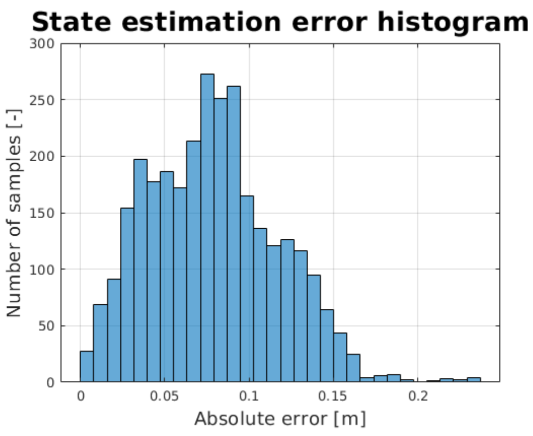


Estimation error histogram.

**Figure 6.3.** Demonstration of line trajectory with tag model A.

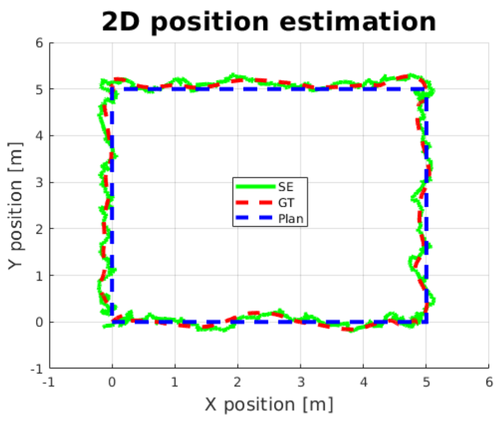


Position estimation.

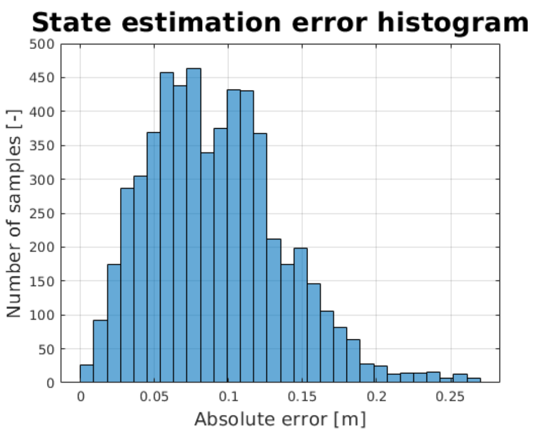


Estimation error histogram.

**Figure 6.4.** Demonstration of line trajectory with tag model B.

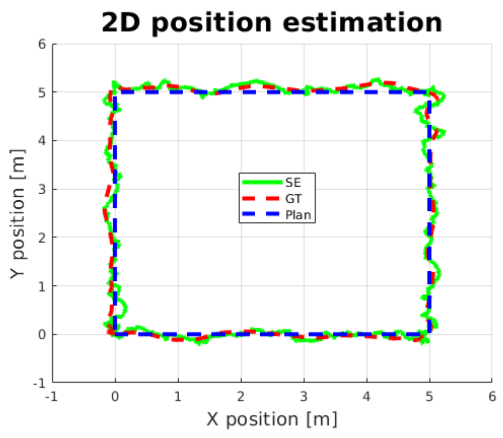


Position estimation.

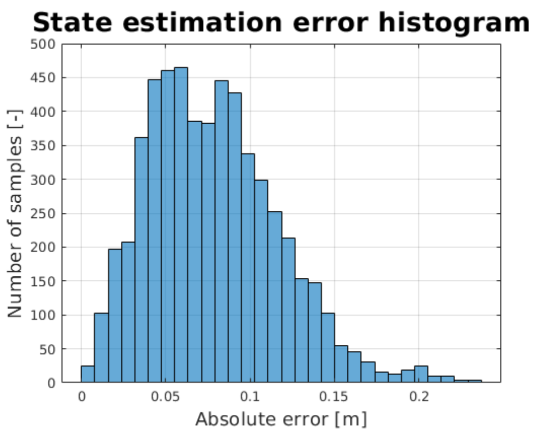


Estimation error histogram.

**Figure 6.5.** Demonstration of rectangle trajectory with tag model A.

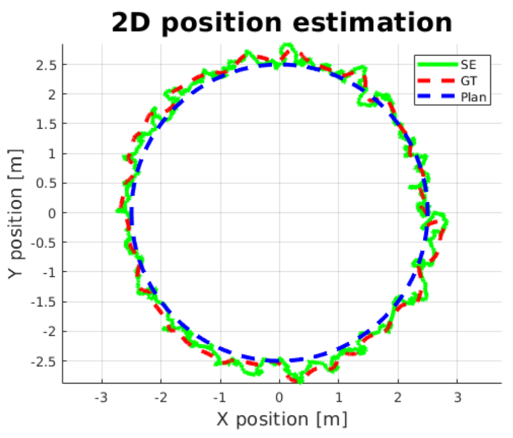


Position estimation.

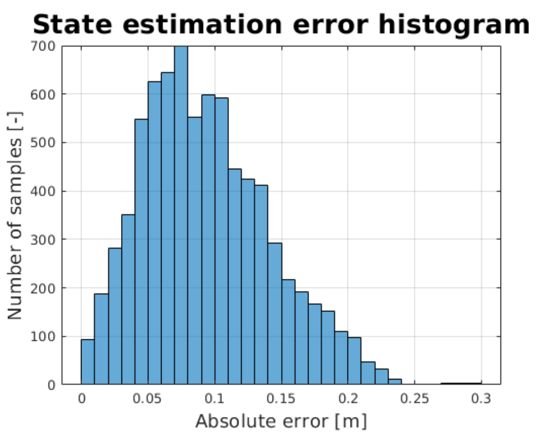


Estimation error histogram.

**Figure 6.6.** Demonstration of rectangle trajectory with tag model B.

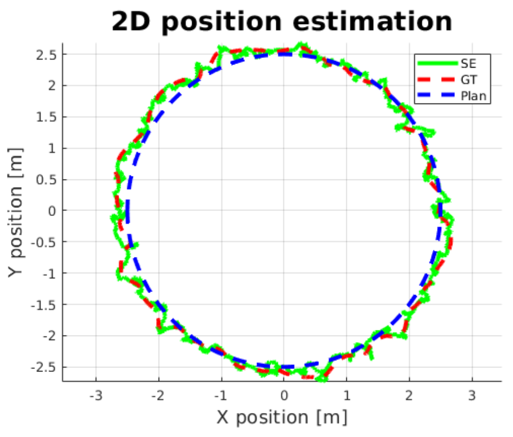


Position estimation.

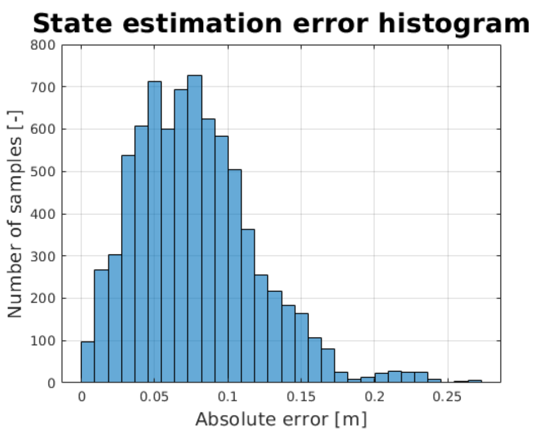


Estimation error histogram.

**Figure 6.7.** Demonstration of circle trajectory with tag model A.



Position estimation.



Estimation error histogram.

**Figure 6.8.** Demonstration of circle trajectory with tag model B.

The experiments were also evaluated statistically, considering the position estimation mean, maximum value, and the accuracy with 95 % precision. The results can be seen in Table 6.1.

| Trajectory | Tag model | $\mu_e$ [m] | $\max_e$ [m] | $\text{accuracy}_{0.95}$ [m] |
|---|---|---|---|---|
| Static | A | 0.0917 | 0.2688 | 0.1712 |
| Line | A | 0.0926 | 0.2515 | 0.1817 |
| Rectangle | A | 0.0911 | 0.2680 | 0.1702 |
| Circle | A | 0.0939 | 0.2976 | 0.1838 |
| Static | B | 0.0719 | 0.2181 | 0.1436 |
| Line | B | 0.0786 | 0.2367 | 0.1432 |
| Rectangle | B | 0.0782 | 0.2363 | 0.1457 |
| Circle | B | 0.0780 | 0.2711 | 0.1525 |

**Table 6.1.** Evaluation of simulation experiments.

**Conclusion**
For the evaluation of the simulation experiments, two distinct models of UWB tags were used. It is worth noting that during these experiments, the state estimation from the UWB measurement was also used for the UAV control. It can be seen that the simulated UWB localization system is able to track the desired trajectory.

Both tags assumed uncorrelated 2D Gaussian noise. However, tag model A assumed the magnitude of the noise to be below 33.33 cm in 95 % of the cases, while tag model B was more optimistic and assumed that the magnitude of the noise was below 26.67 cm in 95 % of the cases.

The position estimation employing tag model A reached an accuracy of 18.4 cm with 95 % precision throughout all four simulation experiments, while the position estimation employing tag model B reached an accuracy of 15.3 cm with 95 % precision. The maximum observed error in both cases throughout all experiments was below 30 cm. Based on the result of the simulation experiments, the UWB localization technology seems viable for indoor localization with high accuracy demands.

## 6.2 Experiments In Complex Scenario

As conducting the real-world examples was impossible due to the pandemic situation, the simulation part was extended. Instead, a complex scenario was included in the simulation part. This scenario is designed based on the autonomous UAV warehouse inventorying, the main application motivation behind this thesis. In this scenario, accurate localization is vital not only to prevent collisions but also to localize the inventory precisely. The warehouse is filled with shelves containing boxes identified by AprilTags [31]. Inside a warehouse, a drone is spawned on a fixed position (home position). The drone is equipped with IMU and two UWB tags located with 30 cm offsets from its body center on both sides of its $x$ axes. Directly under the base, the gimbal model is placed for camera stabilization, and the Garmin rangefinder is placed on the bottom of the full HD camera.

In this complex environment, which can be seen in Figures 6.11 and 6.12, the algorithms described in Section 5.2 concerning the possible application in autonomous UAV warehouse inventorying were tested.

For AprilTag recognition, a package available at `http://wiki.ros.org/apriltag_ros` was used, and it can be seen in Figure 6.12 that it can identify tags in real-time. For each detection, the package also provides information about the tag's position in the image and an estimated distance from the camera. This information can be converted to a 3D estimate of the tag's position employing the UAV state estimate and the known camera offsets. Because the package provides continuous detection of AprilTags, the algorithm for taking images at specific positions to provide a full spatial coverage was not developed in this thesis but is considered for future work. The AprilTag recognition was successfully tested as 32 out of 32 AprilTags placed on a shelf side were correctly recognized. The 3D localization of tags achieved a worst-case accuracy of 60 cm throughout all detections, but a majority of detections achieved an accuracy of about 30 cm. It is worth noting that the algorithm for post-processing of tag localization output was not developed. Therefore, it is expected that higher accuracy is achievable for the same simulation configuration.

Trajectory planning and execution can be seen in Figures 6.9 and 6.10, where the grey area around obstacles represent their inflation for planning purposes. The demonstrative mission execution was fully autonomous, employing the mission and failsafe managers discussed in Sections 5.2.3 and 5.2.4. The mission consisted of all states from the takeoff to landing. As the duration of the mission is short, battery failsafe mechanisms were not activated. However, these mechanisms were successfully tested in other experiments. The reaction of the UAV can be seen in video attachments. For the simulation of the outage of the UWB localization system, a service was implemented that prevents the UAV onboard estimator from receiving UWB measurements.

Based on the UAV behavior in this complex simulation scenario, it can be expected that the deployment in the real-world scenario should work as well.

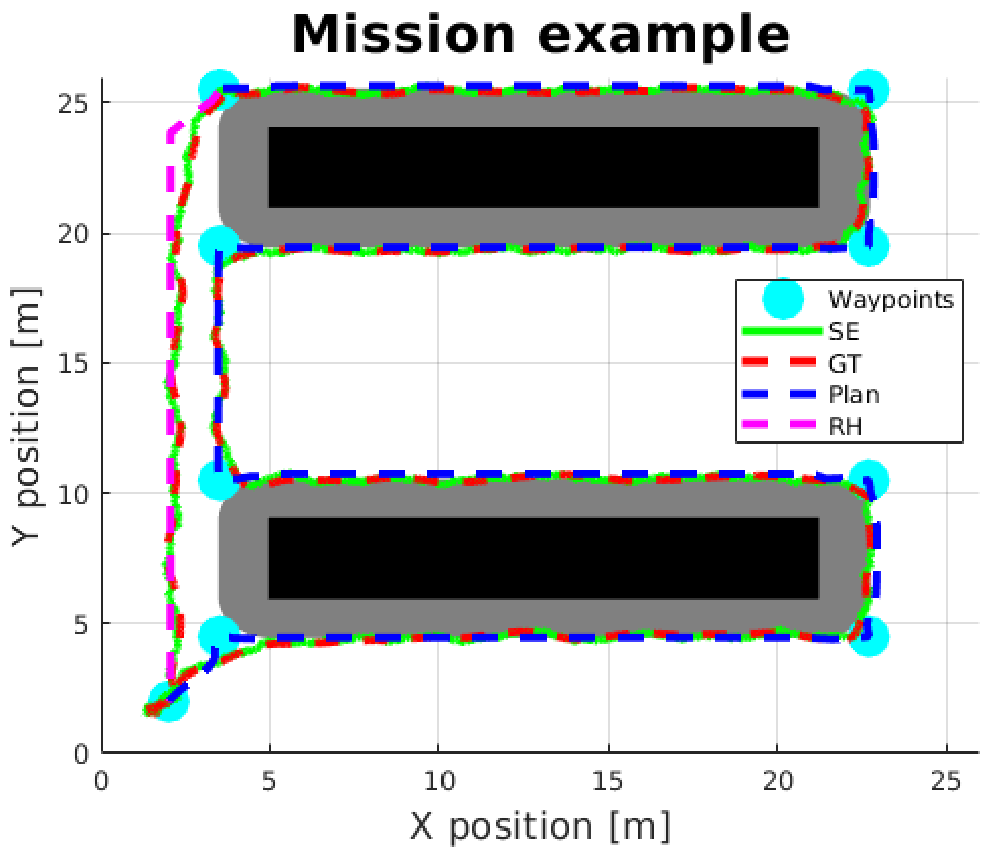


**Figure 6.9.** Example of a mission conducting. The black rectangles represent shelves, while the grey area around them represent their inflation for the planning algorithm.

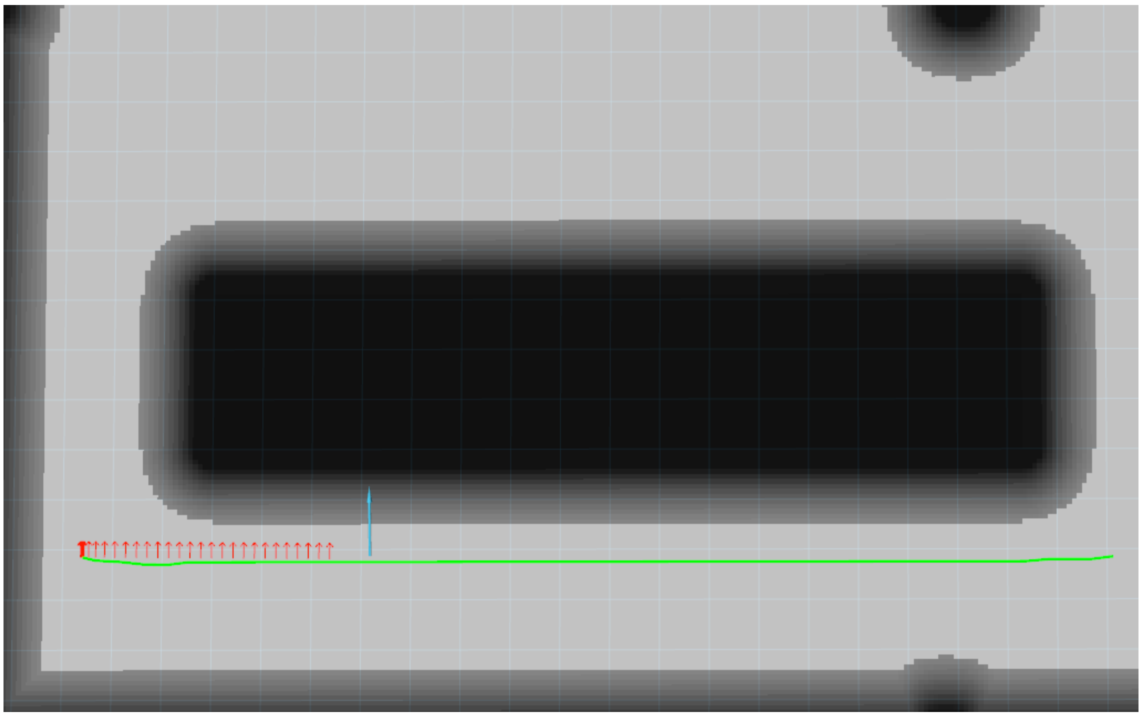


**Figure 6.10.** An rviz image of UAV mission conducting in the warehouse. The green line represents the planned trajectory, the red arrows are the MRS tracker points, and the blue arrow represents the UAV.

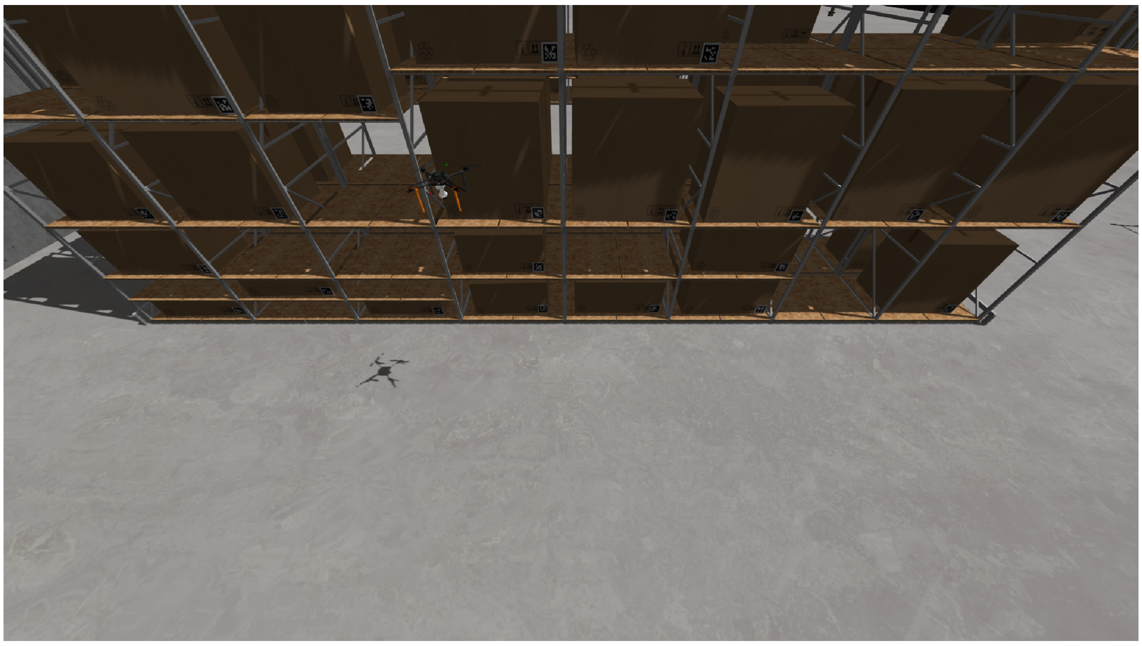


**Figure 6.11.** An image of the warehouse and UAV models while mission conducting.

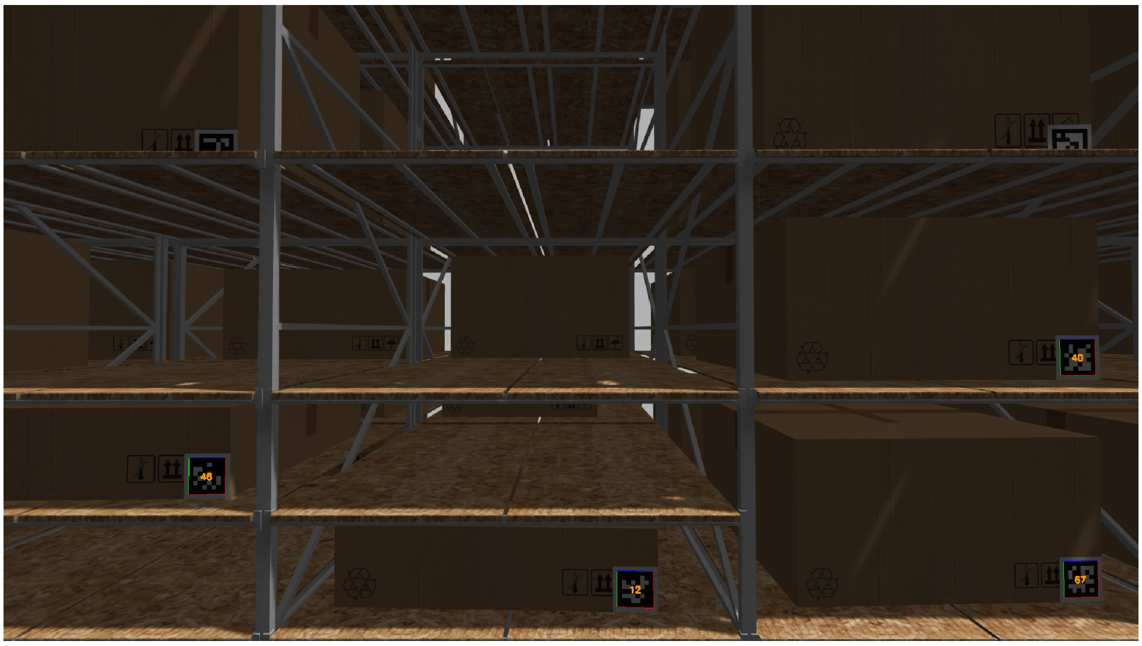


**Figure 6.12.** An image captured by the onboard camera for tag recognition.

# Chapter 7 Conclusions

In this Master's thesis, the UWB localization system was integrated with the MRS UAV framework to replace the most commonly used GNSS localization, which is not reliable indoors. For this purpose, the acquaintance with the UWB localization technique was necessary. The actual UWB localization system was obtained by the Czech company ALIS Tech. However, no documentation nor datasheets were received along with the system. Therefore, it was necessary to conduct experiments to evaluate the performance of the UWB localization system. For this purpose, an experimental platform was developed. Based on the experiment results, the UWB tag was modeled in the Gazebo. All necessary software for integrating the UWB localization systems was implemented, supporting both the simulated and real UWB localization systems. Additionally, supporting packages for UAV experiments were either newly designed or obtained and enhanced. Finally, UAV experiments were conducted and evaluated. Unfortunately, only the simulation experiments were carried, as several things prevented the conducting of the real-world experiments.

The goals of this thesis, according to the assignment, are commented in the following lines.

The research into the state-of-the-art methods of indoor UAV localization was presented in Section 2.1, along with a comparison of chosen methods.

Three ROS packages for sensor fusion and UAV localization based on the Kalman filtering were studied in Section 3.2.

The design and the implementation of the necessary software for the integration of the UWB localization system with the rest of the onboard control system were discussed in Chapter 5.

The simulation experiments and the verification of all the developed algorithms were presented in Chapter 6.

Unfortunately, the real-world experiments were not carried out due to several issues caused by a pandemic situation lasting through most of the time dedicated to the work on the thesis. The promised support from the UWB localization system manufacturer regarding the correct setup of the TDoA localization technique was not received. Due to that, only the TWR-ToF localization technique with a large time delay was available. As researched in Section 2.1, the TDoA technique provides a faster response due to the reduced number of messages necessary for localization. Also, the premises chosen for the real-world experiments were not accessible due to strict hygienic requirements. Nevertheless, the experiments are expected to be carried out in the foreseeable future. As a compensation for the missing real-world experiments, the simulation experiments were extended to include a complex warehouse scenario described in 6.2.

By presenting this thesis, the documentation of the system was provided.

All the UAV experiments were conducted with the UWB localization system as the only direct source of position measurements. As the state estimator employing the UWB localization system was integrated into the feedback control loop. Based on the results of the UAV simulation experiments, the work in this thesis, enhanced by solutions to problems mentioned in Section 7.1, is expected to be successfully deployed in a real-world application of autonomous UAV warehouse inventorying.

## 7.1 Future Work

#### Heading Estimation

As the heading is usually calculated employing the magnetometers, this approach is not sufficient in the indoor environment where high disturbances in the magnetic field are expected due to the presence of ferromagnetic materials. Therefore, it would be nice to obtain a reliable heading estimation from the tags. As discussed in 3.3, the simple calculation of the heading from two tags placed at known positions relative to the UAV body center is not reliable due to a high ratio of noise magnitude and the small distance between tags limited by the UAV dimensions.

One possible way to obtain a more reliable heading estimate is to use more tags on the UAV body. Therefore, the influence of adding third and fourth tags will be examined concerning the heading estimation by approximating their positions with a line.

Research into the advanced variants of Kalman filter with state constraint will be carried out to determine whether the information about the relative position of the tags can be included to obtain a better heading estimate.

The last proposed approach is the implementation of cascade KFs used to preprocess the raw UWB measurements incorporating the UAV IMU linear acceleration and angular rate readings. This way, the modularity of the KF provided by the MRS UAV odometry package could be preserved. Before the heading KF, a preprocessing KF for each UWB tag should reduce the covariance of the UWB measurements, resulting in a more accurate heading estimate.

#### UAV Warehouse Inventorying

Based on the work implemented in this thesis, a project implementing the autonomous UAV warehouse inventorying is in development. For the project, the development of a more sophisticated mission and failsafe manager Section 5.2.3 that includes the collision detection and avoidance algorithms is necessary.

An algorithm that can read identification codes commonly used (barcodes, QR codes, etc., instead of AprilTags) is also required for the warehouse inventorying.

Additionally, a mission planner, currently only planning trajectories, needs to be extended to generating the set of waypoints on which it is necessary to take a picture to provide full coverage of the warehouse inventory. In an ideal case, the UAV is able to fly through the middle of the corridor between two shelves. That way, a distance from identification codes can be obtained. Based on this distance, the size of the identification code, and the camera parameters, the size of the identification code projection onto the image in pixels can be calculated. While taking into account the maximum expected deviation from the planned trajectory, the overlapping ratio can be calculated to ensure the full coverage.

# References

[1] ”CHRobotics”. *”Understanding Euler Angles”*.
http://www.chrobotics.com/library/understanding-euler-angles.

[2] ”Muthukrishnan. In: *”Location- and Context-Awareness”*. ”Berlin: ”Springer Berlin Heidelberg”, ”2005”.

[3] Jeffrey, Charles. *An Introduction to GNSS GPS, GLONASS, BeiDou, Galileo and other Global Navigation Satellite Systems*. 2nd edition ed. Calgary, Alberta, Canada: NovAtel Inc., 2015. ISBN 978-0-9813754-0-3.

[4] Ogaja, Clement A. *Applied GPS for Engineers and Project Managers*. American Society of Civil Engineers, 2011. Available from DOI 10.1061/9780784411506.
https://ascelibrary.org/doi/abs/10.1061/9780784411506.

[5] J. Sanz Subirana, J.M. Juan Zornoza. *Ionospheric Delay — ESA Navipedia*.
https://gssc.esa.int/navipedia/index.php?title=Ionospheric_Delay&oldid=13741.

[6] Packard, Hewlett. Fundamentals of quartz oscillators. *Electronic Counters Series*. 1997.

[7] Sahinoglu, Zafer, Sinan Gezici, and Ismail Guvenc. Ultra-wideband positioning systems: Theoretical limits, ranging algorithms, and protocols. 01, 2008, pp. 6-10. Available from DOI 10.1017/CBO9780511541056.

[8] Kshetrimayum, Rakhesh Singh. An introduction to UWB communication systems. *Ieee Potentials*. IEEE, 2009, Vol. 28, No. 2, pp. 9–13.

[9] Lakkundi, V. Ultra wideband communications: History, evolution and emergence. *Acta Polytechnica*. 2006, Vol. 46, No. 4.

[10] Tiemann, Janis, and Christian Wietfeld. Scalable and precise multi-UAV indoor navigation using TDOA-based UWB localization. In: *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2017. pp. 1–7.

[11] Raja, Asad Khalid, and Zhibo Pang. High accuracy indoor localization for robot-based fine-grain inspection of smart buildings. In: *2016 IEEE International Conference on Industrial Technology (ICIT)*. 2016. pp. 2010–2015.

[12] Mautz, Rainer, and Sebastian Tilch. Survey of optical indoor positioning systems. In: *2011 international conference on indoor positioning and indoor navigation*. 2011. pp. 1–7.

[13] Mautz, Rainer. Indoor positioning technologies. ETH Zurich, Department of Civil, Environmental and Geomatic Engineering ..., 2012.

[14] Mraz, Lubomir. *Accuracy Considerations for UWB Indoor Tracking in an Industrial Environment*.
https://www.sewio.net/accuracy-considerations-for-uwb-indoor-tracking-in-an-industrial-environment/.

[15] Wang, Kai, Xing Yu, Qingyu Xiong, Qiwu Zhu, Wang Lu, Ya Huang, and Linyu Zhao. Learning to improve WLAN indoor positioning accuracy based on

DBSCAN-KRF algorithm from RSS fingerprint data. *IEEE Access*. IEEE, 2019, Vol. 7, pp. 72308–72315.

[16] Havlena, Vladimır, and Jan Stecha. Moderni teorie rizeni. *Modern Control Theory, Text Book, CTU in Prague (in Czech)*. 1999.

[17] Kalman, Rudolph Emil. A new approach to linear filtering and prediction problems. 1960.

[18] Julier, Simon J, and Jeffrey K Uhlmann. New extension of the Kalman filter to nonlinear systems. In: *Signal processing, sensor fusion, and target recognition VI*. 1997. pp. 182–193.

[19] Servi, L, and Y Ho. Recursive estimation in the presence of uniformly distributed measurement noise. *IEEE Transactions on Automatic Control*. IEEE, 1981, Vol. 26, No. 2, pp. 563–565.

[20] Simon, Dan. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[21] Quigley, Morgan, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source Robot Operating System. In: *ICRA workshop on open source software*. 2009. pp. 5.

[22] Koenig, Nathan, and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. pp. 2149–2154.

[23] MRS. *ctu-mrs*.
`https://github.com/ctu-mrs`.

[24] Petrlik, M., T. Baca, D. Hert, M. Vrba, T. Krajnik, and M. Saska. A Robust UAV System for Operations in a Constrained Environment. *IEEE Robotics and Automation Letters*. 4, 2020, Vol. 5. ISSN 2169-2176. Available from DOI 10.1109/LRA.2020.2970980.
`https://ieeexplore.ieee.org/document/8979150`.

[25] PX4. *PX4 Drone Autopilot*.
`https://github.com/PX4`.

[26] Moore, T., and D. Stouch. A Generalized Extended Kalman Filter Implementation for the Robot Operating System. In: *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, 2014.

[27] Lynen, Simon, Markus W Achtelik, Stephan Weiss, Margarita Chli, and Roland Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In: *2013 IEEE/RSJ international conference on intelligent robots and systems*. 2013. pp. 3923–3929.

[28] ”Microcon”. *”Hybridní dvoufázové krokové motory řady SX”*.
`https://dspace.tul.cz/bitstream/handle/15240/49554/Krokovy_motor_%28rady_SX17%29.pdf`.

[29] ”Vishay Semiconductors”. *”Transmissive Optical Sensor with Phototransistor Output”*.
`https://www.vishay.com/docs/81147/tcst2103.pdf?fbclid=IwAR1kks1zrwa8DlXoXuiwbaDljwh-NpeXoKg-G9nSW7yOozlV8unnsFFCSU8`.

[30] Marder-Eppstein, Eitan. *navigation*.
`https://github.com/ros-planning/navigation`.

[31] Olson, Edwin. AprilTag: A robust and flexible visual fiducial system. In: *2011 IEEE International Conference on Robotics and Automation*. 2011. pp. 3400–3407.