POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

Evaluation of UWB positioning on a mobile robot using precise optical tracking system

Supervisors

Prof. Marcello CHIABERGE

Dott. Mauro MARTINI

Dott. Alessandro NAVONE

Dott. Marco AMBROSIO

Dott. Umberto ALBERTIN

Candidate

Claudia GALLIANO

October 2024

Summary

Knowing the pose of a robot is essential for the successful execution of most of the tasks it usually performs. Consequently, mobile robot localization becomes a crucial problem to address.
Odometry is a commonly used technique for the localization of Unmanned Ground Vehicles (UGVs). It exploits the information coming from the wheel encoders and the inertial measurement unit (IMU) to estimate the pose of the robot. However, it is very sensible to slippage, and, since it calculates the position by integrating the encoders' data, it accumulates errors, becoming more and more inaccurate in short time. For GPS-denied zones and indoor environments, Ultra-Wideband (UWB) seems to be a promising technique for the precise tracking of moving objects. The huge bandwidth (>500 MHz) allows high resolution in time and consequently in range too, leading to a positioning error of less than 30 cm.

This study aims to implement and analyze different localization algorithms that rely only on UWB data for the estimation of the position of a rover. Three different algorithms for nonlinear problems have been implemented, ranging from standard approaches to more innovative ones. Firstly, two of the mainly used probabilistic algorithms have been developed, the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF). While the EKF linearizes the state and the measurement model at the point of current estimate, the UKF uses a set of sigma points to better capture the nonlinearity present in the models, making it more robust against high nonlinearity. An alternative to the more classical and widely accepted probabilistic algorithms is represented by the combination of a Neural Network (NN) with the EKF. The main idea is to make the EKF adaptive by adding a NN to better estimate the error on the measurements coming from the UWB antennas at each time step, and then use this estimation, converted into variance, in the filter. Thus, using different variance values for the various situations allows to better understand which measurements to trust more and which not, potentially leading to better localization results for the robot.
A fundamental step for the performance evaluation of the different algorithms and

the training of the NN is the creation of a new dataset. Therefore, all experimental tests were conducted using the Jackal UGV, equipped with several sensors, including an IMU, to get information about the orientation, acceleration, and angular velocity of the rover, the wheel encoders for the linear and angular velocity, and a UWB antenna, to get the distances between the tag and each of the four anchors. Also, a Vicon system was used as ground truth. The data coming from the sensors was collected both under Line of Sight (LOS) and Non-line of Sight (NLOS) conditions, where different obstacles in different positions were used in order to test their effect on the radio performance.

Table of Contents

List of Tables

List of Figures

Acronyms

AI
Artificial Intelligence

ANN
Artificial Neural Network

EKF
Extended Kalman Filter

KF
Kalman Filter

LOS
Line of Sight

ML
Machine Learning

NLOS
Non-line of Sight

NN
Neural Network

OS
Operating System

ROS
Robotic Operating System

UGV

Unmanned Ground Vehicle

UKF

Unscented Kalman Filter

UT

Unscented Transform

Chapter 1

Introduction

1.1 Objective of the thesis

The ability to precisely determine a robot's position is essential for most of the task it usually undertakes. Hence, mobile robot localization is an extremely important challenge to address.
Odometry is a widely used technique for localizing Unmanned Ground Vehicles (UGVs), however, it suffers from slippage and accumulates errors over time, becoming more and more inaccurate. For GPS-denied zones and indoor environments, Ultra-Wideband (UWB) emerges as a promising technique for the precise tracking of moving objects. The huge bandwidth allows for fine resolution in time, and so in range too, resulting in a positioning error of less than 30 cm.

The objective of this thesis is to develop different localization algorithms to track the Jackal UGV in 2D, exploiting the UWB technology. The standard approach consists of an Extended Kalman Filter (EKF) or an Unscented Kalman Filter (UKF) using UWB data alone to estimate the position (EKF_{UWB} and UKF_{UWB}). However, a commonly used strategy for enhancing localization accuracy is sensor fusion, so both the EKF and the UKF were implemented with sensor fusion too (EKF_{SF} and UKF_{SF}). Additionally, an innovative approach was developed by combining a Neural Network with the EKF_{UWB}, offering a more advanced and adaptive method for localization.
A fundamental step for the performance evaluation of the different algorithms and the training of the NN is the creation of a new dataset. Several experimental tests were performed in different scenarios, both under Line of Sight (LOS) and Non-line of Sight (NLOS) conditions, involving obstacles of different materials, such as metal, cardboard, styrofoam, and wood, placed in various positions.

1.2 Overview of the chapters

This section provides a brief description of each chapter and aims to give readers a clear understanding of the organization and content of the thesis.

Chapter 1 introduces the context and the objective of the thesis, summarizing the adopted approaches.

Chapter 2 introduces UWB technology, providing a historical overview and its definition and regulations. Then, UWB advantages are highlighted, demonstrating why it is particularly promising for indoor localization applications. Finally, the chapter analyzes the different UWB positioning techniques and presents a comparison with other widely used localization technologies.

Chapter 3 introduces the localization problem, exploring its different types. Then, illustrates two of the most commonly used probabilistic algorithms, the Extended Kalman Filter and the Unscented Kalman Filter. The last part of the chapter focuses on a more innovative approach, represented by the combination of a Neural Network with the EKF, and provides a brief overview of Neural Networks.

Chapter 4 offers a brief description of the instruments used during the conducted experimental tests and the software platforms employed. Then, delves into the creation of a new dataset, outlining the environment setup and the data acquisition process.

Chapter 5 offers a comparative analysis of the different implemented localization algorithms, analyzing their performance based on experimental tests. It presents an in-depth analysis of the results obtained from each algorithm, considering key metrics such as Root Mean Square Error, both along x and y, and on the overall position, as well as Mean Absolute Position Error in different scenarios.

Chapter 6 concludes the thesis, outlining the conclusions of the work and possible future works.

Chapter 2

Ultra-Wideband Technology for Indoor Localization

2.1 Introduction to UWB Technology

2.1.1 Historical Overview

Despite the common belief, UWB history dates back to the end of the XIX century, when Guglielmo Marconi conceived the first radio transmitter using the electromagnetic waves generated by a spark gap. Due to the short duration of the spark gap pulses, these waves spread across a wide range of frequencies, thus reflecting the principles of UWB.
At the beginning of the XX century, wireless technology began to spread more, arising the first issues. Spark gap transmission, in fact, was occupying large part of the radio spectrum, and without means of synchronization, two stations could not broadcast simultaneously in the same area. Additionally, spark transmitters were heavy and very energy consuming, so better spark gap generators were under research in those years.
A new interest in UWB technology emerged in the late '60s with significant contributions.
Between 1969 and 1984, H. F. Harmuth published several papers and books, exposing the basics for UWB transmitters and receivers.
Almost during the same period, Ross and Robbins extended the use of UWB not only for communication, but also for radar and sensing applications. In particular, Ross filed a milestone patent to the US Patent office on 17th April 1973, number US 3,728,632: "Transmission and reception system for generating and receiving base-band pulse duration pulse signals without distortion for short base-band communication system". Actually, this was the first modern UWB communication

system.
In 1989, the term "Ultra-WideBand" was coined for the first time by the U.S. Department of Defense to signify terms like impulse, carrier-free, baseband, time domain, and much more.
In 1994, T. E. McEwan invented the "Micropower Impulse Radar". This was the first time for a UWB to operate at ultra-low power, to be extremely compact, and inexpensive.
The first commercial approvals came in 2002 by the Federal Communication Commission (FCC). So, the UWB became commercially viable, allowing it to operate at low power in an unlicensed spectrum ranging from 3.1 to 10.6 GHz, with a power spectral density (PSD) lower than -41.3 dBm/MHz.
In 2006, the Institute of Electrical and Electronics Engineers (IEEE) produced a regulation and a standard for UWB communication, still used nowadays: "IEEE 802.15.4a UWB– Low-Rate Wireless Personal Area Networks (WPANs), Standard ECMA368 High Rate Ultra Wideband PHY and MAC Standard, Standard ECMA-369 MAC-PHY Interface for ECMA-368, Standard ISO/IEC 26907:2007, Standard ISO/IEC 26908:2007". [1]
UWB became more and more common when Apple released the iPhone 11 with its UWB U1 chip in 2019, followed later on by Samsung, Google and many others.

2.1.2 Definition and Regulations

According to the previously mentioned IEEE 802-15a standard, an UWB signal is either a signal with an absolute bandwidth B equal or greater than 500 MHz, or a signal with a fractional (relative) bandwidth B_r larger than 20%.

$$B \geq 500 \ MHz, \ or \ B_r = \frac{(f_u - f_l)}{f_c} > 20\%, \tag{2.1}$$

where f_u and f_l represent the upper and lower frequencies at which the power spectral density is 10 dB below its maximum and $f_c = \frac{(f_u+f_l)}{2}$ is the central frequency.
During the years, different countries have released their own regulations for UWB devices. In particular, these regulations cover the applications of UWB technology, the allocated frequency ranges, the emission limits, and the techniques to reduce possible interference caused by the UWB device. The first regulations were released in February 2002 by the US FCC, setting the usable frequency range for UWB indoor applications between 3.1 and 10.6 GHz, and the emission limit to -41.3 dBm/MHz.
In Europe, instead, regulations have been available a bit later, in March 2006. The usable frequency range here is divided into two bands: 4.2-4.8 and 6-8.5 GHz, where on the first band a mitigation technique has to be used. Without mitigation,

the requirement is -70 dBm/MHz instead of -41.3 dBm/MHz.
Emission levels for both FCC and EU are shown in Figure 2.1.

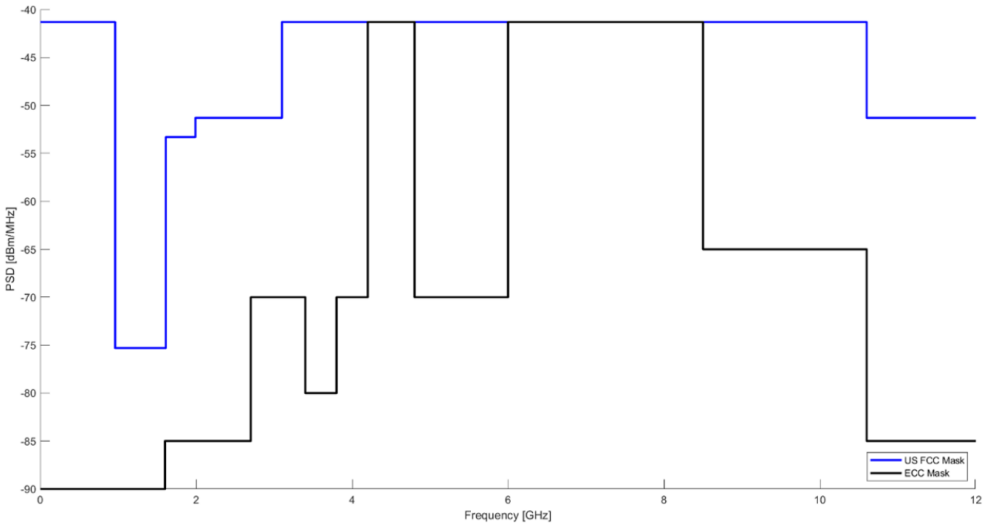


Figure 2.1: FCC and EU spectral mask for Indoor UWB [2]

UWB is able to transmit high data rates thanks to its wide bandwidth. However, the emitted power has to be very low in order to respect the regulations. The limitation to -41.3 dBm/MHz for the range 3.1-10.6 GHz results in a total emitted power of only 0.56 mW for the FCC mask. For the EU, it is even smaller, since regulations are more strict. Due to this, UWB can only be used for short range applications.
There are two main approaches to exploit the UWB frequency range:

- *Impulse Radio* (IR): the transmission is based on very short pulses radiated directly via the UWB antenna (no need for a carrier) that cover an ultra-wide bandwidth.

- *Orthogonal Frequency Division Multiplexing* (OFDM): the UWB spectrum is divided into a set of broadband OFDM channels, each one transmitted through a sub-carrier frequency. The orthogonality of the sub-carriers avoids cross-talks, leading to a more efficient exploitation of the band and a more robust transmission in case of noise. However, regarding the signal processing, the complexity is increased.

The choice between which of the two approaches should be followed depends on the the application. [2]

2.1.3 UWB Advantages

UWB has many unique advantages that have led to its rapid spread. Firstly, as said before, UWB is able to transmit high data rates over short ranges, in fact, for localization applications we can talk about real time tracking, since it can reach a capacity of hundreds of Mbps thanks to its wide bandwidth.
UWB also presents good noise immunity since it works at high frequencies, from 3.1 GHz to 10.6 GHz, thus limiting signal interference with devices on more commonly used frequencies. [3]
Moreover, the huge bandwidth ensures low transmit power, extending battery life.
Another advantage of UWB signals is the high time resolution that allows an accurate determination of the Time of Flight (TOF). These signals, in fact, are so short due to the inverse relationship between time and bandwidth.
Additionally, one of the most common phenomenon in wireless communications is the multipath propagation. Radio signal, indeed, can reflect, diffract, or scatter off the different obstacles in the environment, like walls or floor, causing the original signal to get to the receiver in different moments in time and with different intensities, since it followed different paths. Luckly, thanks to the high time resolution and short wave length, an UWB system is able to distinguish two events that occur with a short time distance to each other, making it much more robust against multipath propagation.
To conclude, great system design flexibility can be achieved thanks to the large spectrum, adjusting parameters based on the required data rate, power, and range. [4]

2.2 UWB for Indoor Localization

UWB seems to be a promising technology for the precise tracking of moving objects in GPS denied zones and indoor environments.
An UWB system is usually composed of two different types of devices: the *tag*, that is the moving transceiver attached to the object to be tracked, and the *anchor*, that is the fixed transceiver used as reference point.
At least three anchors are required for 2D positioning. Each anchor, in fact, represents a circle with the anchor itself as the center and with a radius equal to the distance between the tag and the anchor. So, in order to uniquely determine the position of the moving tag, at least three circles are required, since the position is represented by their intersection. For 3D positioning, instead, at least four anchors are required, as each one represents a sphere. Additional anchors can be used in order to get more precise measurements.

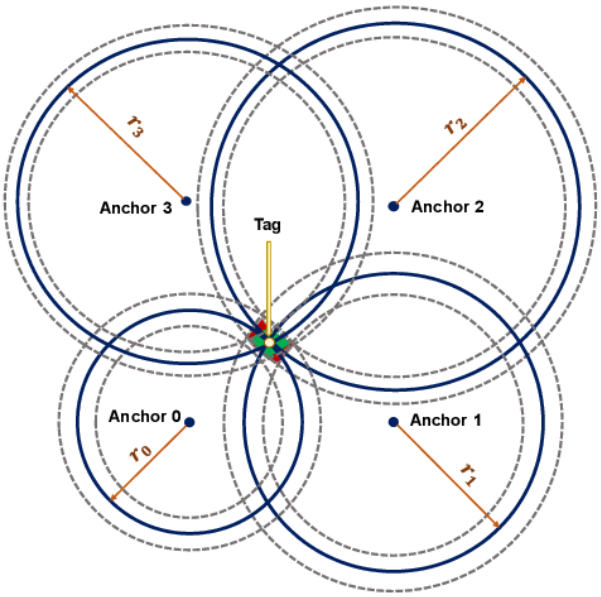


Figure 2.2: Visualization of tag positioning using four anchors, each represented as the center of a circle.

Moreover, the UWB system is able to calculate the distance between two points via Time of Flight (TOF), that is the time required by the radio pulses to travel from one point to another. In fact, the distance of the tag from an anchor is obtained by dividing the measured TOF by the speed of radio waves.
The huge bandwidth (>500 MHz) allows high resolution in time and consequently in range too. In a single nanosecond, a wave can travel almost 30 cm. [5]

2.2.1 Positioning Techniques

For UWB positioning, several techniques can be used based on the use case or application.

Received Signal Strength (RSS)

As long as the radio signal goes far from its origin, it becomes weaker and weaker, so the further two points are, the smaller the received signal strength will be. Actually, the RSS value can be used to derive the distance between two points, the transmitter and the receiver.
Nevertheless, the received signal is not so good, since the indoor environment is usually full of obstacles, generating lots of reflections of the signal. Due to this, the error on the RSS value can be very large, making it unsuitable for accurate localization. [1], [2]

Creating a map of the power of the received signals in the environment is one of the approaches to mitigate this problem. It is called *fingerprint*, and the tag's position can be obtained by matching the RSS values of the different anchors to the previously constructed map. However, also this solution is not the best one, since it is time-consuming and a new map has to be created for each new configuration of the environment. [2]

Time of Arrival (TOA)

This technique is one of the mainly used for UWB positioning applications. It uses the Time of Flight (TOF), so the time required by the signal to travel from the transmitter to the receiver, to estimate the distance between them. TOF can be computed since the received message contains the starting time.
However, this arise the first problem, in fact, without a perfect clock synchronization between the transmitter and the receiver it is not possible to get an accurate measurement of the TOF and so of the distance too.
In order to avoid this issue, TOF protocols have been introduce, like Single-Sided Two Way Ranging (SS-TWR) or Double-Sided TWR (DS-TWR), thus requiring the time measurement on one device only. In TWR methods, instead of using direct timestamps, a set of time periods, like t_{round} and t_{reply}, is used to get the distance between two transceivers. This is because a certain measured time interval is the same for all devices, regardless their own clock references.
For the SS-TWR method, the TOF is computed as:

$$T_{tof} = \frac{1}{2}\left(t_{round_A} - t_{reply_B}\right), \tag{2.2}$$

where $t_{round_A} = \tau_{ARx} - \tau_{ATx}$ is the round-trip time of a signal measured at Device A, and $t_{reply_B} = \tau_{BTx} - \tau_{BRx}$ is the reply time of a signal measured at Device B.

For the DS-TWR it is quite similar, the same measurements are performed on both sides, so two round-trip times and two reply times are required. The TOF in this case is computed as:

$$T_{tof} = \frac{1}{4}\left[\left(t_{round_A} - t_{reply_B}\right) + \left(t_{round_B} - t_{reply_A}\right)\right] \tag{2.3}$$

To compare the two methods, SS-TWR is easier since it requires only one device to get the measurements, but may be less precise than DS-TWR, which measures time on both devices to improve accuracy.

The following figure illustrates the previously described protocols: Single-Sided Two Way Ranging (SS-) and Double-Sided TWR (DS-).

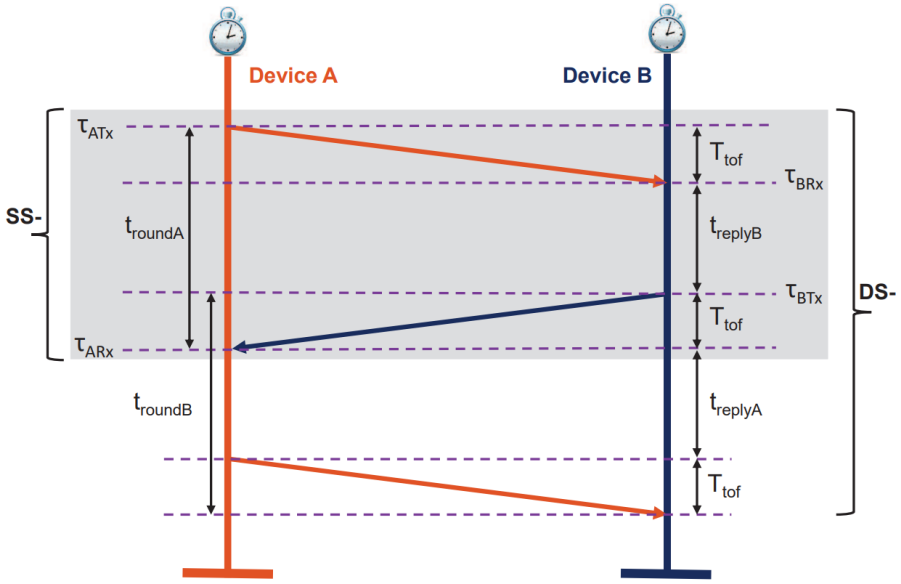


Figure 2.3: Illustration of single- and double-sided Two-Way Ranging (TWR) methods (©2018 IEEE)

However, this method is subject to other errors such as Propagation-Time Delay, Transmission-Time Delay, Receiving-Time Delay, and Preamble Accumulation-Time Delay, which are all analyzed in deep in [6].

Time Difference of Arrival (TDOA)

Another widely used method for UWB positioning is the TDOA. The time differences of the arriving signal between the different anchors are used to locate the tag in the environment. This technique, in fact, requires almost perfect synchronization between the anchors, placed in known positions.
Once all anchors have received the tag's signal and recorded the respective timestamp, the synchronization process is performed. Through the difference in the arrival time it is possible to calculate a hyperbola, which is, by definition, the locus of points such that the difference of the distances from two fixed points (two anchors), is constant. The intersection between the hyperbolas represents the actual position of the tag.
However, with this method, even slight noise can cause big localization errors. [1], [2]

Angle of Arrival (AOA)

Unlike the other techniques, AOA does not require distance measurements, but only the angle of arrival of a signal with respect to a reference line, usually the anchor's axis. The advantage of this approach is that it requires only two measurements

to localize in 2D, so only the measured angles from two anchors are needed. By intersecting the lines that represent the direction of the signals, the position of the tag is retrieved.
Nevertheless, these measurements are more difficult to obtain, and a more complex hardware, like an antenna array, is required on each of the anchors to measure the angle of arrival of a signal. [1], [2]

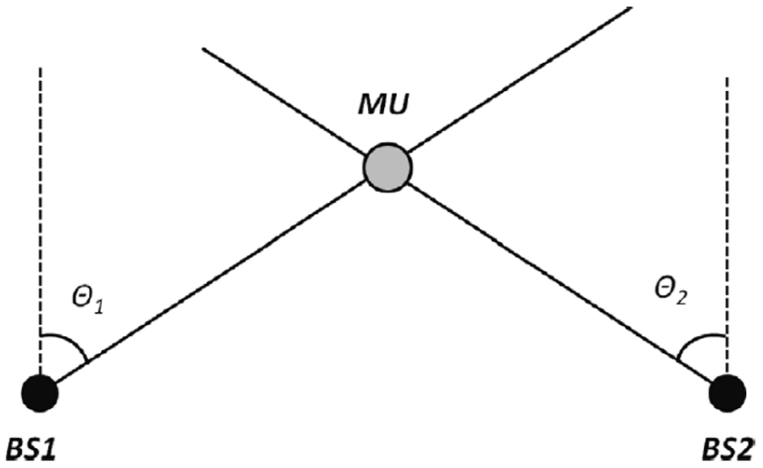


Figure 2.4: Positioning using the Angle of Arrival method [2]

2.3 Comparison with Other Localization Technologies

This section provides a brief comparison between the different technologies commonly employed in the localization field, highlighting their advantages, limitations, and suitability for different use cases.

Global Positioning System (GPS)

GPS is one of the most widely used localization technologies in the world, mainly employed for outdoor localization applications, such as road navigation and vehicle tracking, with an accuracy ranging from 3 to 10 meters. GPS is a navigation system made up of a constellation of satellites orbiting at about 20,000 km above Earth. The working principle is very similar to that of UWB: it is able to determine the receiver's position by calculating the time required by the signal to travel from the satellite to the GPS device, located on the Earth surface. Since each satellite only gives information about the distance between the satellite itself and the receiver, four satellites are required to determine the location of the GPS device. Indeed,

each satellite generates a sphere with a radius equal to the distance between the satellite and the device on Earth, and since three spheres produce two points of intersection, an additional satellite is needed.
However, GPS highly suffer from signal multipath propagation and requires a free line of sight to multiple satellites for accurate positioning. As a result, in indoor environments, the GPS signal becomes very weak, making it unreliable for indoor localization. [7], [8]

Radio Frequency Identification (RFID)

RFID is a wireless technology that uses radio waves to identify and localize tags or labels attached to objects. A RFDI system is made up of two components: a tag and a reader (or antenna). In particular, two different types of tags can be distinguished, passive and active ones. Passive tags are not provided with an energy source, they are activated only when they receive a radio signal from the antenna, while active tags have their own power supply, so they are able to continuously transmit data, allowing for real-time tracking. Depending on the tag used, RFID can be employed for different applications.
Passive RFID does not allow for real-time positioning, however, these tags are very cheap and are used in a variety of applications, like in the access doors, when you have to scan a card on the reader to let the door open. On the other side, for asset tracking it is not very suitable, since if the tag is not activated by the antenna, the localization information is lost.
Active RFID, like UWB, allows for real-time tracking, but its accuracy is on the order of meters, which is much more than the UWB one, which ranges from 10 to 30 cm. Additionally, active RFID is even more expensive. [9]

Bluetooth Low Energy (BLE)

BLE, as the name suggests, is a Bluetooth version designed to reduce energy consumption and extend battery life. Although they both operate in the 2.4 GHz band, they are quite different, and so are their use cases.
Despite the constantly active connection initiated by Bluetooth, BLE is able to reduce energy consumption by staying in sleep mode unless a connection is initialized, allowing only for a small amount of data transfer.
For indoor localization applications, BLE calculates the position of a tag through the Received Signal Strength (RSS) technique, better explained in section 2.2.1. However, this is not the best way of calculating the position, indeed, an accuracy of around 5 meters is reached, much bigger than the 10-30 cm of the UWB, obtained by exploiting the Time of Flight (ToF) technique, analyzed in section 2.2.1. In terms of battery life, both BLE and UWB tags can last several years on a single battery with location updates every few seconds. However, UWB tags are about

twice as expensive as BLE tags.
Introduced in early 2020, BLE 5.1 also supports the Angle of Arrival (AOA) technique, which improves localization performance, despite the more complex implementation, leading to an accuracy of less than 1 meter. [9], [10]

Wi-Fi Positioning Systems (WPS)

Wi-Fi Positioning Systems exploit the already existing Wi-Fi networks to determine the location of a device, so no additional hardware is required.
Like the BLE, WPS uses the Received Signal Strength (RSS) technique to estimate the distance of the device from the nearby routers. An accuracy of 5-15 meters can be reached, which is a lot with respect to UWB. Moreover, its accuracy can be affected by factors such as signal interference, the density of access points, and environmental obstacles. [9]

Camera-based tracking

With the advent of AI-based image processing, camera-based positioning has become more feasible. Two different approaches can be followed: camera infrastructure and camera trackers.
In the first approach, a system of fixed cameras is used to track objects or people in the viewed environment. By using previously trained AI models, it is possible to identify the tracked objects in real time. However, when the same object is moving in different areas, so it is viewed by different cameras, the system should be able to recognize that it is the exactly same object that is just moving around. Actually, this is not that simple to achieve, and it is not even easy to know where the objects are located. Moreover, for situations in which objects appear partially covered or badly illuminated, it is even more difficult. As a result, this approach is less suitable for asset tracking with respect to UWB. Indeed, with UWB, each tracked object has its own tag, so the identification problem does not exist. The only disadvantage is that a lot of tags are required.
Instead of using a system of cameras, camera trackers can also be employed. In this approach, each object that needs to be tracked has its own camera attached to it. In this way the identification problem is solved. However, to make the localization process easier, some landmarks are added to the environment. This approach can provide the same accuracy of an UWB system, but it is much more expensive. [9]

Light Detection And Ranging (LIDAR)

LIDAR uses a rotating laser to measure the distance and the angle to obstacles precisely. However, LIDAR alone is not able to provide the location of an object, it is mainly used to check for obstacles. In combination with other localization technologies, such as UWB, it allows for absolute positioning with very high precision,

on the order of millimeters.
Typically, LIDAR is used for mobile robots in autonomous navigation, when the robot needs to sense the environment in order to check for possible obstacles along the path. [9]

To conclude, the table below summarizes the advantages and disadvantages of each of the previously described localization technologies.

	UWB	BLE	Wi-Fi	GPS	Camera	LIDAR	Passive RFID
Positioning Accuracy	10-30cm	5m	5-15m	5-10m (outdoor only)	10-30cm	1cm	NA
Scalability	✓	✓	✓	✓	✗	✗	✓
Real-time	✓	✓	∼	∼	✓	✓	✗
No dedicated infrastructure	✗	✗*	✗/✓	✓	✓	✓	✗
Range	20-30m	20m	30m	NA	20m	60-100m	5cm-5m
Tracker cost	low	very low	medium	medium	very high	very high	ultra low

Table 2.1: Different localization technologies in comparison. [9]

where the symbol ∼ means that Wi-Fi and GPS allow for real-time tracking, but with a very low update rate, while NA stands for Not Applicable. "The asterisk * for BLE infrastructures indicates that in certain scenarios (with WiFi access points that support BLE positioning), there is no need for additional infrastructure. However, this is not the case for most access points" [9].

Chapter 3

Localization in Robotics: Principles and Algorithms

3.1 Introduction to the Localization Problem

Knowing the pose of a robot is essential to most of the tasks a robot usually performs. Hence, mobile robot localization is an extremely important problem to tackle.
The pose of a robot, obtained by exploiting the information coming from the wheel encoders and the inertial measurement unit (IMU), is known as *odometry*, and it is one of the mainly used techniques for the localization of UGVs. However, it is very sensible to slippage, and, since it calculates the position by integrating the encoders' data, it accumulates error, becoming more and more inaccurate in short time. For this reason, for GPS denied zones and indoor environments, UWB seems to be a promising technique for the precise tracking of moving objects.

Localization problems can be very difficult to be solved, depending on the nature of the environment and the knowledge a robot may have initially and at run-time. Three types of localization problems can be distinguished.

- *Position tracking* assumes that the initial position of the robot is known and consists in the continuous updating of the robot's pose as it moves through the environment. This type of localization problem is local, since the uncertainty is limited to the region around the robot's actual pose.

- *Global localization* is more difficult than positional tracking, in fact, it does not make any assumption on the initial pose of the robot, which is at first placed somewhere in the environment.

- *Kidnapped robot problem* is even more difficult than the global localization problem. During its own operation, in fact, the robot is suddenly kidnapped from its current position and it is moved to a new unknown one. This is actually a variant of the global localization problem, where each time the robot is kidnapped, it has to re-localize itself again from scratch. Moreover, this problem has a practical importance, in fact, it is very useful to test algorithms' ability to recover from global localization failures.

Another discriminating factor for the difficulty of a localization problem is related to the environment, which can be static or dynamic.
An environment is called *static* when the only moving object is the robot, while all other objects remain at the same position. On the other side, an environment is *dynamic* when objects other than the robot can change their location over time.
Of course, localization in dynamic environments is more difficult than in static ones.

The possible approaches can be passive or active. In *passive* ones, the localization algorithm does not control the motion of the robot, it just observes the robot operating in the area. In *active* ones, instead, the algorithm is also responsible of minimizing localization errors controlling the motion of the robot too, so it typically leads to better localization results.

To conclude, the difficulty of the localization problem is of course related to the number of robots to be tracked. So, a multi-robot localization problem will be much more difficult to solve. [11], [12]

3.2 Probabilistic Algorithms

Probabilistic robotics is a new approach to robotics able to explicitly represent the uncertainty in robot perception and action by using the probability theory. These algorithms, indeed, use probability distributions across the entire space of possible hypotheses to represent robot's information.
Uncertainty in robotics arises from five different factors: environments, sensors, robots, models, computation. Usually, this uncertainty has always been ignored, but as robots are moving to much more complex environments than before, it is of extreme importance to be able to represent it in order to build successful robots. Probabilistic robotics is capable of doing exactly this. [11]

Below are listed and explained two of the mainly used probabilistic algorithms in localization problems.

3.2.1 Extended Kalman Filter

The Kalman Filter is a powerful algorithm for state estimation, and it is one of the mainly used in several fields, from target tracking and navigation to microeconomics and digital image processing. [13]
However, it does not work with nonlinear problems, and since most real-world problems are nonlinear, an extension of it is required.
Problems can be nonlinear in two different ways: in the motion model and in the measurement model. The Extended Kalman Filter (EKF) is designed to handle both of these cases by linearizing the system at the point of current estimate and then applying the equations of the linear KF.

Just like the KF, the EKF is made up of two main steps: prediction and update. In the prediction step both the state and the covariance for the next time step are estimated, while in the update step the prediction previously made is corrected with the measurements coming from the sensors.

The equations for the *prediction* step are the following ones:

$$x = f(x, u) \tag{3.1}$$

$$F = \left.\frac{\partial f(x_t, u_t)}{\partial x}\right|_{x_t, u_t} \tag{3.2}$$

$$P = FPF^T + Q \tag{3.3}$$

where x, the state, is represented by the nonlinear state transition function *f(x,u)*, where u is the possible control input. The state transition function is linearized by taking its partial derivative with respect to the state. The Jacobian is then evaluated at the point of current estimate to obtain matrix F, which is the state transition matrix.
The covariance matrix P is predicted in the same way as in the linear KF, where Q is the process noise covariance matrix, which accounts for the uncertainty in the process over time, due to factors not modelled in the state transition function. So, the new uncertainty is predicted as the sum of the old one and some additional uncertainty that comes from the environment (Q).

The equations for the *update* step are the following ones:

$$H = \left.\frac{\partial h(x_t)}{\partial x}\right|_{x_t} \tag{3.4}$$

$$y = z - h(x) \tag{3.5}$$

$$K = PH^T(HPH^T + R)^{-1} \tag{3.6}$$

$$x = x + Ky \tag{3.7}$$

$$P = (I - KH)P \tag{3.8}$$

As previously stated, nonlinearity can be present in the measurement model too. Thus, the approach is the same as in the prediction step: the measurement function *h(x)* is linearized by taking its partial derivative with respect to the state and then the Jacobian is evaluated at the point of current estimate in order to obtain matrix *H*. This matrix, called measurement matrix, describes how to convert the state into a measurement, modelling the sensors. In fact, for example, units and scale of the data coming from the sensors may not be the same as the ones of the state, so this matrix is responsible of reporting the state into the measurement space.
Then, the residual y, the Kalman gain K, and the updated state and covariance are computed through the same formulas as in the linear KF. The R matrix used to calculate the Kalman gain is the measurement noise matrix, which represents the noise introduced by the sensors during the measurements.
Moreover, for the EKF, matrices F and H are calculated at each time step since the evaluation point changes each time. [14]

To conclude, while the KF is an optimal filter because the estimate uncertainty is minimized, all Kalman Filter modifications for nonlinear systems, like the EKF and the later explained UKF, are sub-optimal filters since approximated models are used. [15]

3.2.2 Unscented Kalman Filter

Until now the EKF has been the standard approach to solve nonlinear problems, anyway, another, and sometimes even more suitable algorithm for these kind of problems is the Unscented Kalman Filter (UKF).
The first and main difference with the EKF is that, instead of using all the points of a probability distribution, it uses only the so called *sigma points*. These points are a set of representative points chosen in order to capture important statistical properties of the system's state distribution, like its mean and covariance. The key idea behind sigma points is to approximate the system's state distribution without relying on assumptions of linearity or Gaussianity.

In Python there exist several already implemented functions to choose sigma

points, like the FilterPy function *MerweScaledSigmaPoints()* or the *JulierSigmaPoints()* one. They both need only some parameters to be set in order to better choose the sigma points that represent the distribution.
In particular, since 2005, Van der Merwe's algorithm is the mainly used in research and industry. It requires only three parameters to control how the sigma points are distributed and weighted: α, β, and κ.
A good choice for α is to use $0 \leq \alpha \leq 1$, where the larger α the more it will spread the points far from the mean. Moreover, a larger value for α weights the mean much more than the other points.
The figure below illustrates how a different value for α can be used to choose different sigma points.

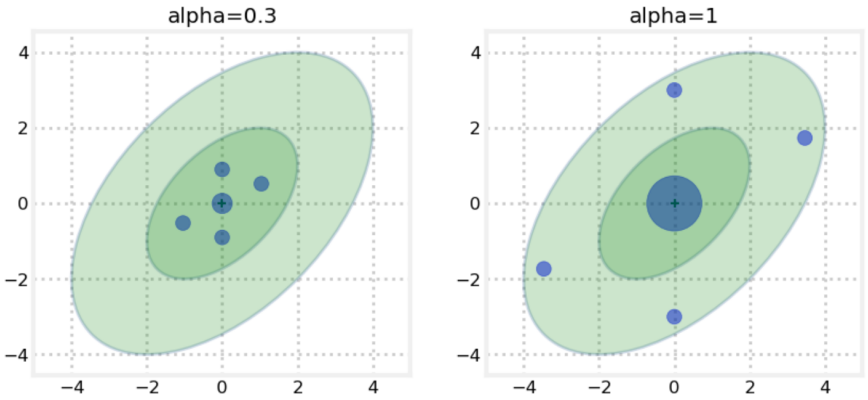


Figure 3.1: Effect of using different values for α on the sigma points. [14]

$\beta = 2$ is a good choice for Gaussian problems, while κ is set equal to $3 - n$ where n is the dimension of the state.
Mathematically, the first sigma point is the mean of the input, $\chi_0 = \mu$, and it is the one at the center of the ellipses in Figure 3.1. The other sigma points, instead, are computed as:

$$\chi_i = \begin{cases} \mu + \left[\sqrt{(n+\lambda)\Sigma}\right]_i & \text{for i=1 .. n} \\ \mu - \left[\sqrt{(n+\lambda)\Sigma}\right]_{i-n} & \text{for i=(n+1) .. 2n} \end{cases} \tag{3.9}$$

where $\lambda = \alpha^2(n + \kappa) - n$. In words, by using these formulas, the covariance matrix Σ is firstly scaled by a constant and then square-rooted. The symmetry is ensured by both adding and subtracting it from the mean.
Weights for the mean and covariance of χ_0 are computed in different ways.

$$W_0^m = \frac{\lambda}{n + \lambda} \tag{3.10}$$

$$W_0^c = \frac{\lambda}{n+\lambda} + 1 - \alpha^2 + \beta \tag{3.11}$$

where (3.10) is the formula to calculate the weight for the mean, and (3.11) for the covariance.
The weights for the other sigma points $\chi_1 ... \chi_{2n}$ are calculated in the same way for both the mean and the covariance.

$$W_i^m = W_i^c = \frac{1}{2(n+\lambda)},\ i = 1 \ ..\ 2\text{n} \tag{3.12}$$

As the EKF, the UKF is made up of two main steps: prediction and update.
Once the sigma points and their corresponding weights have been generated using some function, in the *prediction* step each sigma point is passed through the non-linear function $f()$, generating a new set of sigma points, Y, projected forward in time according to $f()$. Then, the mean and covariance of the prior are computed by applying the Unscented Transform (UT) on the transformed sigma points.
The equations are the following ones:

$$Y = f(\chi) \tag{3.13}$$

$$x = \sum w^m Y \tag{3.14}$$

$$P = \sum w^c (Y - x)(Y - x)^T + Q \tag{3.15}$$

where (3.14) and (3.15) are the equations for the UT.
Since the *update* is performed in the measurement space, the sigma points of the prior must be converted into measurements through a measurement function $h()$. So, Z becomes the new set of measurement sigma points. Just like the update step, the mean and covariance of these points are computed by applying the UT.

$$Z = h(Y) \tag{3.16}$$

$$\mu_z = \sum w^m Z \tag{3.17}$$

$$P_z = \sum w^c (Z - \mu_z)(Z - \mu_z)^T + R \tag{3.18}$$

Then, the residual and the Kalman gain are computed as:

$$y = z - \mu_z \tag{3.19}$$

$$K = \left[\sum w^c (Y - x)(Z - \mu_z)^T\right] P_z^{-1} \tag{3.20}$$

Finally, the updated state and covariance can be computed through the following formulas:

$$x = x + Ky \tag{3.21}$$

$$P = P - KP_zK^T \tag{3.22}$$

Even though EKF is considered the standard approach for nonlinear problems, it can be quite difficult to understand and use. UKF, on the other side, is easy to implement and can lead to even more accurate results than the EKF, especially in very nonlinear situations. Indeed, the main difference between them is in the way they treat nonlinearity. The EKF linearizes the state and the measurement model at the point of current estimate, while the UKF uses a set of sigma points to better capture the nonlinearity present in the models, making it more robust against high nonlinearity. The main disadvantage of the UKF is that it can be slower than the EKF, but this actually depends on how the EKF solves the Jacobian, if numerically or analytically. If numerically, the UKF is faster. [14]

3.3 Innovative Approach: NN + EKF

3.3.1 Neural Networks Overview

Neural Networks (NN) are powerful models in AI that mimic the behavior of the human brain. The idea behind NNs, indeed, is to make a machine able to learn, adapt, and make decisions all alone.
Just like for the human brain, the fundamental unit of an Artificial Neural Network (ANN) to process and transmit information is the *neuron*, also called *node*.
The brain neuron consists of three main parts: the *cell body*, which contains the nucleus that controls the functioning of the cell, the *dendrites*, that receive the information coming from the other neurons, and the *axon*, that transmits the information away. On the other side, the artificial neuron is a mathematical model designed to reflect the behavior of the biological one. It takes as input $x \in R^n$ and a +1 intercept term. Then the sum of the weighted inputs is computed, and a bias

term is added.
This process is represented in the figure below as the linear function.

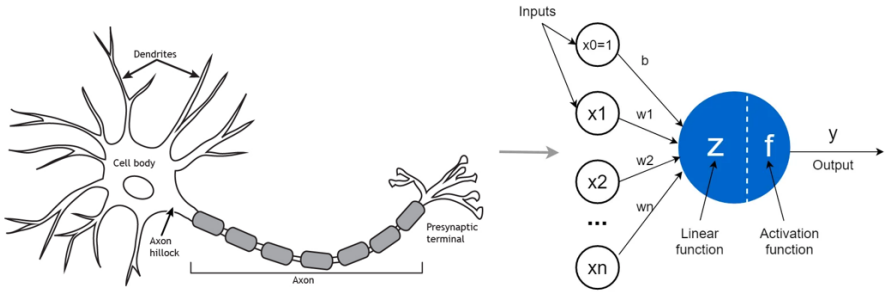


Figure 3.2: Biological similitude between a brain neuron and an artificial neuron

The output of the node is defined as:

$$y = f\left(b + \sum w_i x_i\right) \tag{3.23}$$

where f is the activation function, a nonlinear function that decides if a neuron is activated or not, just like the action potential for the biological neuron .
Different activation functions can be used, but one of the most common is the sigmoid function

$$f(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}, \tag{3.24}$$

which takes values in [0, 1].
Other popular activation functions are:

- Hyperbolic tangent (tanh): $f(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \in [-1, 1]$
- Rectified linear unit (ReLU): $f(z) = max(0, z)$

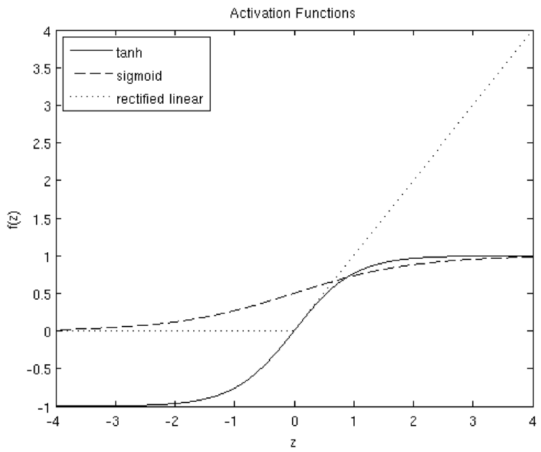


Figure 3.3: Plot of some of the most commonly used activation functions [16]

A NN is composed of many interconnected nodes, organized in layers, where each node has its own weights and bias. Weights are used to determine how much influence each input has on the output.
As a result, each layer is characterized by two parameters: a matrix W, that contains the weights for that layer, and a vector b, that contains the biases.
In each NN three types of layers can be distinguished: the leftmost layer, called *input layer*, the rightmost layer, known as the *output layer*, and finally the *hidden layers*, that are the middle ones. In small and more simple NNs, like the one in Figure 3.5, the hidden layer is a single layer.
Furthermore, the output of a NN can be used as a final result or as an input to the same NN (loopback), or to another NN (cascade).

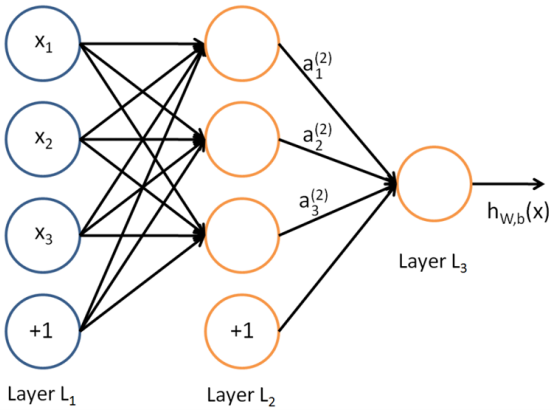


Figure 3.4: Example of a small Neural Network architecture [16]

The most common NNs are the so-called Feedforward Neural Networks (FFNNs), whose graphs does not present any feedback or loops. In fact, the information can travel only one way, from the input layer, to the hidden layer(s), and finally to the output layer. Instead, if the information can flow both forward and backward, the NN is called Feedback Neural Network. Obviously, it is more powerful and complex than the FFNN.

Neural Networks are so used in a variety of problems because they have some unique advantages with respect to classical statistical techniques. They are able to solve very complex and nonlinear problems by dynamically adapting themselves to them, and to reconstruct the rules that govern the optimal solution for these problems.
A fundamental part for NNs is the training. Learning from the data, the network is able to adapt itself each time a new data is available and to find values for the network parameters (W, b) that make the training error smaller. Usually, the more data is available, the smaller the error will be.

Once the NN has been sufficiently trained, it is able to perform well also on new unseen data.

NNs are suitable for an infinity of problems, like classification, and pattern, speech or image recognition, but in particular they are employed when the problem is complex, when classical techniques have not led to good results, and when an approximate solution can be considered as well. [17] [16] [18]

3.3.2 Adaptive Localization Algorithm

An alternative to the more classical and widely accepted probabilistic algorithms, like the previously described EKF and UKF, is represented by combining a Neural Network (NN) with the EKF.
UWB measurements, indeed, are highly influenced by the surrounding environment, therefore, being able to identify and mitigate Non-line of Sight (NLOS) conditions, by exploiting machine learning and deep learning techniques, could lead to better results. [19]
The main idea is to make the EKF adaptive, so to add a NN to better estimate the error on the measurements coming from the UWB antennas at each time step, and then use this estimation, converted into variance, in the filter.
Indeed, by only using the EKF, the measurement matrix R is fixed, so the variance is the same at each time step. Actually, this may not be very accurate, since, for example, in NLOS conditions, the error could be greater, and it would be better to trust less the measurements with respect to the dynamical model by using a bigger variance value. Vice versa, when the robot is quite far from the obstacles and the line of sight is free it would be better to use a smaller variance for the measurements. This capacity to change the variance in the filter based on the different situations could lead to better results in the localization of the robot.

Novelty detection refers to a set of machine learning techniques that are able to identify new or unknown patterns with respect to the nominal data learned during the training phase. In particular, in the context of UWB signals, novelty detection can be employed to identify NLOS and multipath conditions as novelties, providing an estimation of the error on the information coming from the sensor.
To this aim, the NN was trained on a dataset consisting of different rosbags registered in the same fixed environment, all under Line of Sight (LOS) conditions, intentionally inducing overfitting in the model.
During the inference phase, to evaluate its performance in different scenarios, the NN was tested under both LOS and NLOS conditions. Indeed, introducing an obstacle in the environment causes UWB signal reflections and NLOS conditions, leading to a distorted signal compared to the original one. As a result, the model

detects this deviation with respect to the nominal data, indicating a novelty at the specific location where the altered signal is detected. [19]

The architecture proposed for this purpose is an autoencoder, a specific kind of NN consisting of two parts: the encoder and the decoder. The encoder transforms the input data into a lower dimensional latent representation, capturing its essential features, while the decoder produces a reconstruction of the input from this condensed information. The general structure of an autoencoder is the one depicted in the following figure.

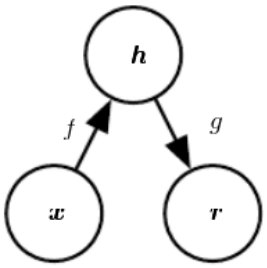


Figure 3.5: The general structure of an autoencoder: the input $\boldsymbol{x}$ is mapped by the encoder function $\boldsymbol{f}$ into $\boldsymbol{h}$, which is then mapped by the decoder function $\boldsymbol{g}$ into $\boldsymbol{r}$, which is the final output and a reconstruction of the original input. [20]

Autoencoders are not designed to learn to set $g(f(x)) = x$ everywhere since this is not very useful. Instead, they are designed to learn to copy only some specific input features, those that resemble the training data.
Usually, the dimensionality of the latent space p is lower than that of the input n, as the information is compressed. However, in this case, an overcomplete autoencoder architecture was employed. A higher dimensionality for the latent space ($p > n$) allows the model to capture more features from the input data, providing a more detailed and potentially redundant representation of the information.

In particular, the architecture employed in this work is depicted in Figure 3.6. The encoder stage consists of three dense layers, respectively with dimensions $N = n$, N_{E1}, and $N_{E2} = p$, where $N < N_{E1} < N_{E2}$. Each layer is activated by its own ReLU function. The decoder stage, instead, comprises two dense layers with dimensions N_{D1} and N, where $N_{D1} > N$, restoring the initial dimension. While the first decoder layer is activated by a ReLU function, the second one uses a Leaky ReLU activation function, introducing a slight negative slope to address potential dead neurons and enhance the robustness of the model. [19], [20]

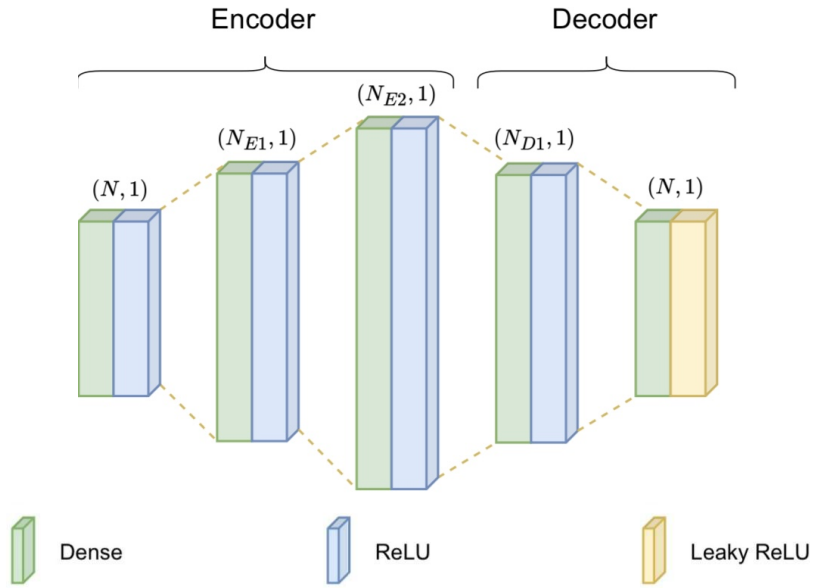


Figure 3.6: The architecture of the employed Neural Network consisting of an overcomplete autoencoder. [19]

Once the NN has been trained, it is able to predict the distance of the tag from each anchor in nominal conditions. So, to detect the novelty, an error for each of the anchors is calculated as the difference between the predicted distances (outputs) and the measurements coming from each anchor (inputs). This error is then converted into variance, in order to use it in the filter. In particular, it is better to approximate the errors with a line with two saturation points, accurately choosing the two saturation points.

Figure 3.7 illustrates the relationship between the error and the variance used in the filter. For small errors (below 0.001) the variance is saturated at a minimum value of 10^{-4}, indicating that the measurements are reliable and subject to minimal uncertainty. As the error increases beyond 0.001, the variance grows linearly, reflecting the increased uncertainty with larger errors. Finally, for errors greater than 0.6, the variance saturates at 0.1, chosen as an upper limit for the uncertainty.

These values were chosen based on the variance values employed in the base EKF and UKF algorithms. Indeed, for experiments conducted under LOS conditions, a value of 0.01 (10 cm) was selected, while for experiments conducted under hard NLOS conditions, the upper limit of 0.1 (approximately 30 cm) was chosen to account for the worst-case scenario.

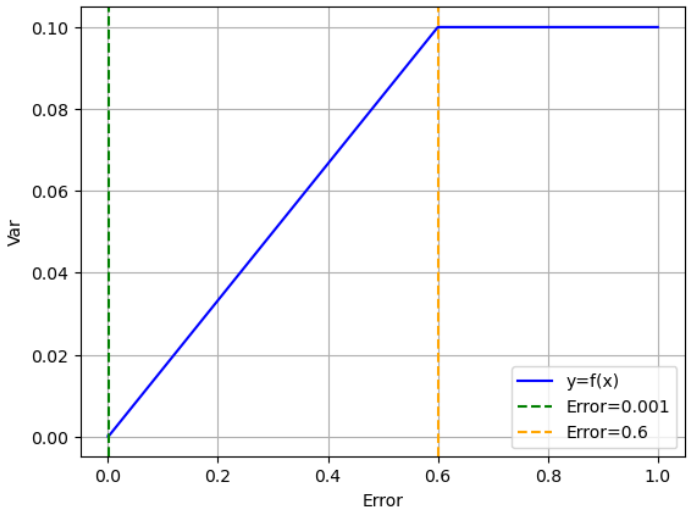


Figure 3.7: Plot showing the relationship between the error obtained by the Neural Network and the variance used in the filter.

Then, the EKF base algorithm is modified to interact with the Neural Network. The covariance matrix R becomes dynamic since at each time step it is updated with the new values estimated by the NN. As a result, the filter is able to adapt itself to different situations, and better understand which measurements to trust more and which not, potentially leading to better results in the localization of the robot.

Chapter 4

Experimental Work

4.1 Instrumentation

In the experimental tests conducted at the PIC4SeR laboratory, the following instruments were used. A brief description of each of them is provided below.

4.1.1 Jackal UGV

All experimental tests were conducted using the Jackal UGV, a small, fast and versatile robot platform designed by Clearpath Robotics.
Equipped with an on-board computer, IMU and GPS fully integrated with ROS, it is completely customizable with a lot of compatible sensors, cameras and other accessories that make it especially suitable for research. Moreover, to meet the need for additional computing power or storage, two payload mounting areas are available. [21]

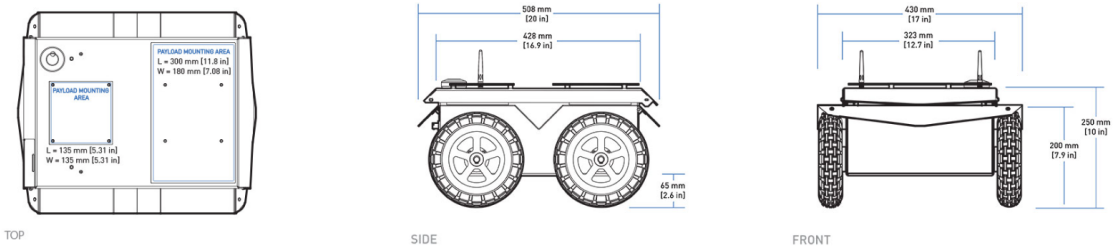


Figure 4.1: Top, side, and front views of the Jackal UGV with detailed dimensions of the payload mounting areas and overall rover size. [21]

Some of the most important technical specifications of the Jackal UGV can be found in the table below.

External Dimensions	508 x 430 x 250 mm
Internal Dimensions	250 x 100 x 85 mm
Weight	17 kg
Max Payload	20 kg
Max Speed	2.0 m/s
Battery	270 Wh Lithium Ion
Run Time	Heavy Usage: 2 h Basic Usage: 8 h
User Power	5V at 5A, 12V at 10A, 24V at 20A
Drivers and APIs	Packaged with ROS Kinetic
Integrated Accessories	Wireless Game controller, GPS, IMU, On-Board Computer, WIFI Adapter, Accessory Mounting Plates

Table 4.1: Jackal UGV Technical Specifications. [21]

In particular, on the Jackal UGV employed in the experimental tests, an UWB antenna was added. The focus of these tests, in fact, is on the possibility to use the UWB technology for indoor localization in different scenarios. So, an UWB antenna used as a tag was added on top of the rover in the smallest payload mounting area. Other sensors employed, already present on the rover, include: an IMU, to get information about the orientation, acceleration, and angular velocity of the rover, and the encoders for the linear and angular velocity.
For a detailed discussion of the rover's kinematic model, refer to Appendix A.

Figure 4.2: Jackal UGV configuration used for all the experimental tests.

4.1.2 Qorvo's DWM1001C modules

Qorvo's DWM1001 module is an UWB and Bluetooth module based on Decawave's DW1000 IC and Nordic Semiconductor nRF52832 SoC, also provided with an on board motion sensor. [22] This module was then integrated on an evaluation board, where the firmware is executed by a microcontroller.
Despite an embedded firmware was already available, another firmware, developed by the Dynamic Distributed Decentralized Systems Group (D3S), a cross-institutional research group based in Trento (Italy), was used. Depending on it, the single module can be used both as a tag or as an anchor in an UWB system.

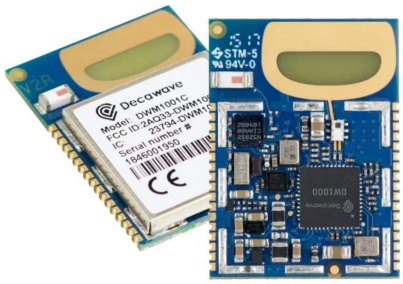


Figure 4.3: Qorvo's DWM1001C module. [22]

A close-up of the HW features of the Qorvo's DWM1001C module is provided by the figure below, while some of the most important technical specifications can be found in Table 4.2.

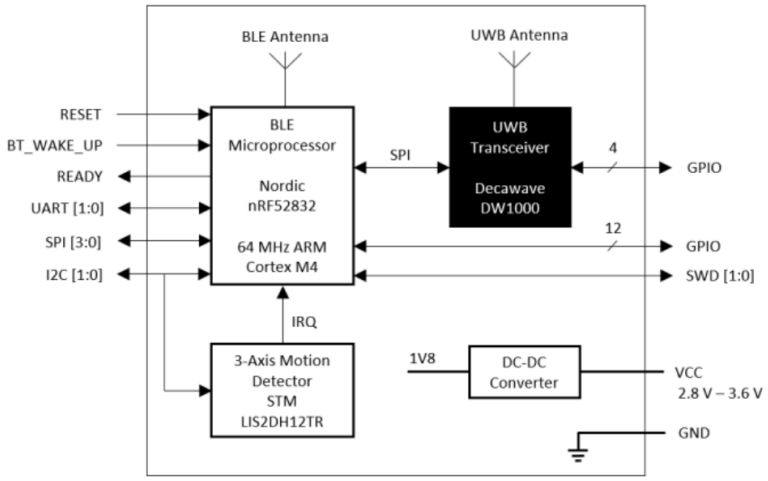


Figure 4.4: HW features of the Qorvo's DWM1001C module. [22]

Size	19.1 mm x 26.2 mm x 2.6 mm
Operating Band	UWB Channel 5: 6.5 GHz
Max PSD	- 41 dBm/MHz
Data Rate	6.8 Mbps (IEEE 802.15.4-2011 UWB compliant)
Integrated Antennas	Decawave DW1000 UWB transceiver Bluetooth chip antenna
Firmware	D3S Contiki Multi-Ranging
Ranging Accuracy	10 cm (max)
Max Range	Up to 60 m in LOS
Ranging Technique	SS-TWR or DS-TWR
Max Ranging Frequency	10 Hz (depends on the number of anchors)

Table 4.2: Technical Specifications of Qorvo's DWM1001C module. [22]

4.1.3 Vicon System

A Vicon motion capture system, consisting of 10 high-resolution infrared cameras, was used as ground truth to evaluate the accuracy of the different implemented algorithms. Figure 4.5 shows the configuration of the cameras, highlighted in green, in the laboratory where all the experimental tests were conducted. In orange, instead, the current position of the Jackal UGV is highlighted.

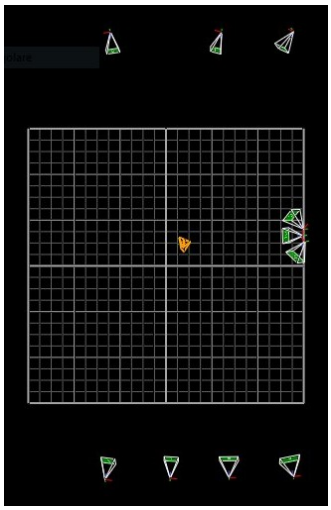


Figure 4.5: Configuration of the Vicon system cameras employed in the experimental tests.

4.2 Software Platforms

This section describes the software development tools employed throughout the experimental work.
In particular, all the software was executed on a machine running Ubuntu 22.04.4 (LTS). For the real-time collection of the data coming from the onboard sensors of the Jackal UGV, the second version of Robotic Operating System (ROS) was used. The ROS2 distribution employed for this work was Humble.
The implemented algorithms were all developed using *Python*, a widely used high-level programming language, in the Visual Studio Code environment.

4.2.1 ROS2

The Robotic Operating System (ROS) is a free and open source meta-operating system for robots that provides some OS-like functionalities, like hardware abstraction and low-level device control. As a middleware, ROS connects software and hardware components, or software with other software, by providing common services and tools. Actually, it is widely used to develop robotic applications since it simplifies complex tasks such as communication and data management, allowing the developer to focus only on the implementation of the higher-level software, without worrying about low-level hardware management. ROS2 is ROS latest version, designed to address the limitations of ROS and to improve its capabilities.

A robotic application is a collection of nodes, each providing a specific task, organized in packages. Unlike the first version of ROS, ROS2 does not rely on a central ROS Master anymore. Indeed, in ROS, this node was the main one, responsible for the initialization and the management of the communication between nodes. ROS2, instead, uses DDS (Data Distribution Service) for the node communication, allowing for direct interaction between them.
In particular, each node can send and receive data from other nodes via topics, services, actions, or parameters.
Nodes can publish or subscribe to a topic. For example, a sensor node may publish data on a topic, while other nodes subscribe to it, reading the content of that topic, and so retrieving the information coming from the sensor. This way of communication is asynchronous. On the other side, to implement a synchronous communication between nodes, services are used. A service client and a service server are required to implement a request/response communication.
The main difference between these two kinds of communication is that topics are used for continuous and unidirectional data streams, while services are mainly employed for requests, so for a bi-directional flow of data.
Actions, instead, are designed for long running tasks, when status updates are

required, and consist of three parts: a goal, a feedback, and a result. They use the server-client model, indeed, there is an action client node that sends a goal to an action server node that acknowledges the goal and returns a stream of feedback and a result. Finally, parameters allow nodes to dynamically store and retrieve configuration values. [23]

4.2.2 Python Machine Learning Tools

TensorFlow 2

TensorFlow is an end-to-end open source platform, initially developed by Google in 2015, for Machine Learning (ML) and AI that offers a variety of tools, libraries, and community resources. It can be used with a lot of programming languages, including Python, JavaScript, C++, and Java. Its versatility makes it suitable both for research and real-world applications in several fields, including image and speech recognition, generative models, robotics and automation, and many others. In particular, in this work, TensorFlow Keras was used to implement the NN model for the last of the three algorithms (NN + EKF).

Keras

Keras is a high-level API that runs on top of the TensorFlow 2 library, used as the backend. It provides a user-friendly interface, making it easier for developers and researchers to build and experiment with deep learning models. Another advantage is its modularity, in fact, it allows users to easily configure layers, optimizers and loss functions. Moreover, Keras supports both CPUs and GPUs, making it widely used in several fields.

4.3 Dataset Creation

A fundamental step for the performance evaluation of the different algorithms and the training of the NN is the creation of a new dataset.

4.3.1 Environment Setup and Measurements

The dataset was collected at the PIC4SeR (Polito Interdipartimental Center for Service Robotics) laboratory by teleoperating the Jackal UGV within a rectangular area of 5.50 by 3 meters.
For simplicity, a fixed starting point, that coincides with the fixed inertial reference frame used for the measurements, was chosen. Each time a new bag was recorded, indeed, the rover was repositioned at the starting point, with the orientation pointing forward, along the x-axis.

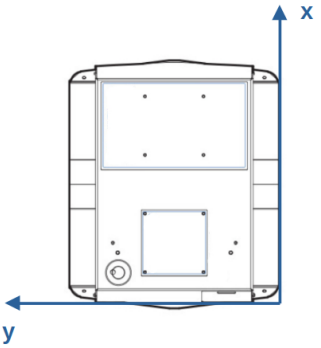


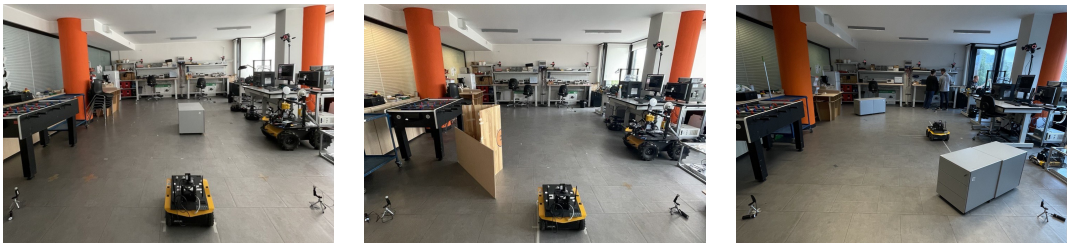
Figure 4.6: Fixed inertial reference frame used for the measurements.

In particular, for the experimental setup four anchors and a tag have been used. Actually, three anchors were already sufficient to track the rover in 2D, but in order to get a more precise measurement, another anchor was added.

Anchor	x [m]	y [m]
0	5.50	-1.25
1	5.50	1.75
2	0	-1.25
3	0	1.75

Table 4.3: Positions of the four anchors with respect to the fixed inertial reference frame.

Different scenarios have been recorded in order to have a dataset as broad as possible. Indeed, different trajectories were traced by the rover, ranging from simple square paths to more random ones.
The data coming from the sensors was collected into rosbags both under LOS and NLOS conditions. In NLOS scenarios, different obstacles in various positions have been used. The effect of obstacles on the radio performance, indeed, depends on the material of the obstacle itself. Depending on it, the signal will pass through the object with little effect or will be completely absorbed. Since radio waves are electromagnetic waves, conductor material will have a big impact on the signal, making it less powerful when it reaches the anchor and leading to a worse accuracy in the measurements.
Obstacles in metal, cardboard, styrofoam and wood in different positions were employed in the tests.
The figures below show some of the obstacles' configurations during the conducted experimental tests.



(a) (b) (c)

Figure 4.7: (a) Metal obstacle centered (b) Wood obstacle obstructing one anchor (c) Multiple metal obstacles obstructing two anchors.

A total of 61 rosbags were recorded, including 15 under LOS conditions, with 6 containing square paths and 9 following some more random trajectories. In addition, 40 bags were recorded under NLOS conditions, with 12 in the presence of a single metal obstacle, 6 with multiple metal obstacles, 6 with cardboard, 6 with styrofoam, and the remaining ones with wood, in different positions.
All the relevant information regarding each experiment, such as the duration and the obstacle type and position, was collected into an Excel file.

4.3.2 Data Acquisition

The data acquisition process was carried out in several sessions, with each session involving the recording of the ROS2 topics published by the different sensors.
In particular, the following topics were recorded:

- /imu/data: contains the data coming from the IMU, such as the angular velocities and linear accelerations along the three axis, and the orientation expressed in quaternions.
- /joint_states: contains the position and velocity of each of the four wheels.
- /odom: contains the data coming from the odometry, such as the position, the linear and angular velocities along the three axis, and the orientation expressed in quaternions.
- /uwb_ranging: contains the data coming from the UWB antennas, so the ranging distance of the tag from each of the four anchors, and other not used information regarding the power.
- /vicon/Jackal/Jackal: contains the data coming from the Vicon system, so the position and orientation expressed in quaternions, used as ground truth.

- /taranis/cmd_vel: contains the linear and angular velocities along the three axis coming from the Taranis radio.

The data coming from each topic was then converted into a Comma-Separated Value (CSV) file. Therefore, for each recording, several CSV files were generated, each containing as columns the different data and as rows the single data acquisitions with their UNIX-16 timestamp (that is the time past from the midnight of the 1st of January 1970, measured in nanoseconds).
Moreover, most of the data required some post-processing operations in order to make it suitable for use in the algorithms, such as the conversion from quaternions to Euler angles, or from one unit of measurement to another. So, once all the data had been processed, it was collected into a single CSV file for better clarity.

Chapter 5

Localization Algorithms: Analysis and Results

Different algorithms have been developed to evaluate localization performance under various conditions. The standard approach consists of an Extended Kalman Filter (EKF) or an Unscented Kalman Filter (UKF) using UWB data alone to estimate the position (EKF_{UWB} and UKF_{UWB}). Sensor fusion is one of the most commonly used strategies to improve localization accuracy, so both the EKF and the UKF were implemented with sensor fusion too (EKF_{SF} and UKF_{SF}). Additionally, an innovative approach was introduced by combining a Neural Network with the EKF_{UWB}, offering a more advanced and adaptive method for localization.

5.1 Sensor Fusion

Sensor fusion is the process of merging the information coming from different sensors to provide a better understanding of the environment or the system being observed. Each sensor, indeed, has some limitations in terms of range, precision, or sensitivity to noise, that can be compensated by integrating data from different sources.
Sensor fusion is highly employed in fields like robotics, autonomous vehicles, and automation, which require a lot of sensors, such as cameras, LIDAR, GPS, IMU, and so on. Merging all the information reduces uncertainty and increases robustness, enhancing overall performance.

The sensors used during the conducted experimental tests have very different sampling frequencies, raising the problem of data synchronization. However, in all the implemented algorithms, no synchronization was performed. Indeed, the filter starts predicting when it receives the first data from the UWB, since it is the

one with the lowest sampling frequency, and then predicts the entire state every time a new measurement arrives, updating only the state variables affected by that specific measurement.

In particular, in the following algorithms, UWB data was used to estimate the x and y coordinates, the odometry data for the velocities, both linear and angular, and the IMU data for the orientation, acceleration and angular velocity of the rover.

5.2 Localization Algorithms in Comparison

In this section the different implemented localization algorithms are briefly described and compared, showing the respective obtained results.

All the developed filters, so EKF and UKF, were implemented using the FilterPy library, a powerful tool designed for implementing and working with Kalman filters and other filtering algorithms. EKF and UKF algorithms have a very similar structure, the main difference is in the way they treat nonlinearity, by linearizing at the point of the current estimate or by using a set of sigma points to capture the nonlinearity present in the models.
Nonlinearity can be present both in the dynamical model and the sensor model. In this case, it is present only in the measurement model of the UWB sensor, so in the function responsible for converting the state into a measurement. Therefore, a *h(x)* function returning the Euclidean distance d (5.1) between the current position (x, y) and the anchor (x_a, y_a), was required for each of the anchors for both EKF and UKF.

$$d = \sqrt{(x - x_a)^2 + (y - y_a)^2} \tag{5.1}$$

For the SF algorithms, an 8-dimensional state vector was used, consisting of x, v_x, a_x, y, v_y, a_y, $theta$, w. In contrast, for the UWB algorithms, the state vector is 4-dimensional, with x, v_x, y, v_y.
For the EKF only, the measurement function *h(x)* is linearized by taking its partial derivative with respect to the state, and then the Jacobian is evaluated at the point of current estimate in order to obtain matrix H, defined as follows.

$$dx = x - x_a$$

$$dy = y - y_a$$

$$H = \begin{bmatrix} \frac{dx}{d} & 0 & 0 & \frac{dy}{d} & 0 & 0 & 0 & 0 \end{bmatrix}$$

In EKF_{SF} and UKF_{SF} algorithms, for measurements from other sensors, the function *h(x)* is linear and simply extracts the corresponding state variable. As a result, in EKF_{SF}, matrix H is mostly composed of zeros, with a single 1 in the position corresponding to the state variable being measured.

Moreover, for both the sensor fusion and UWB implementations of EKF and UKF, the variance of the UWB measurements is fixed. In particular, for the experiments conducted under LOS conditions, a very low variance of 0.01 (10 cm) was chosen since measurements should not present high uncertainty in the absence of obstacles and should be trusted more with respect to the dynamical model. Then, the experiments conducted under NLOS conditions were divided into two categories: soft NLOS and hard NLOS. The first category includes those experiments conducted in the presence of a single obstacle, regardless of the material. For them, a fixed variance of 0.04 (20 cm) was chosen. The worst-case scenario, involving multiple metal obstacles, falls under the second category. In this case, a fixed variance of 0.1 (around 30 cm) was chosen.

Unscented Kalman Filter

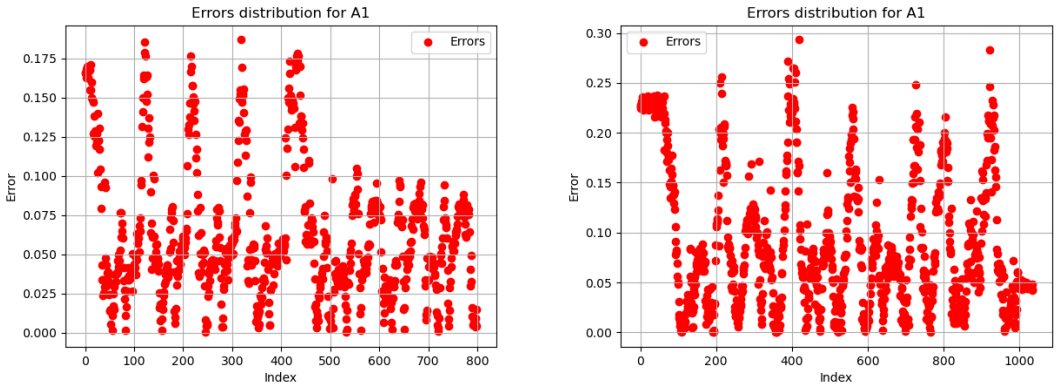
UKF uses a set of sigma points to better capture the nonlinearity present in the model. In this case, sigma points were selected using the function *MerweScaled-SigmaPoints()* from the filterpy.kalman module.
To properly select the sigma points, the function requires four parameters: n, that represents the dimension of the state, α, that regulates how spread the points are from the mean and its weight with respect to the other points, β, that is chosen equal to 2 for Gaussian problems, and κ, that is equal to *3-n*.
For the UKF_{UWB}, the following values were chosen: $n = 4$, $\alpha = 0.1$, $\beta = 2$, and $\kappa = -1$. For the UKF_{SF}, instead, $n = 8$, $\alpha = 0.1$, $\beta = 2$, and $\kappa = -5$.

As it is implemented, the UKF generates numerical errors, in particular regarding the P matrix, that easily becomes non-positive definite after the prediction step, not allowing the Cholesky decomposition of the P matrix itself. To cope with this problem, a function, called *nearestPD()*, was added. This function finds the nearest positive definite matrix to input. So, each time the P matrix is predicted or updated, a check on its positiveness is performed. If it is non-positive definite, the *nearestPD()* function is called. The Python/Numpy implementation for finding

the nearest positive-definite matrix was taken from John D'Errico's port of the MATLAB code (2013) [24], which credits the work of N.J. Higham (1988) [25]. In this way, the algorithm works for most of the experiments.

Innovative Approach: NN + EKF

As explained in subsection 3.3.2, this last approach makes the EKF adaptive by adding a NN to better estimate the error on the measurements coming from the UWB antennas at each time step, and then use this estimation, converted into variance, in the filter. As a result, the variance is now dynamic.
By analyzing the errors produced by the Neural Network, the filter is able to recognize which measurements to trust more and which not.
Below there is a comparison between the errors produced by the NN for the same anchor A1 in different scenarios. On the left, the errors are very small, most of them are below 10 cm, indeed, they refer to a LOS experiment. In contrast, on the right, the worst-case scenario involving multiple metal obstacles is depicted, where the errors are larger, reaching up to 30 cm in some cases.



(a) **(b)**

Figure 5.1: (a) Errors distribution for anchor A1 under LOS conditions (b) Errors distribution for anchor A1 under hard NLOS conditions (presence of multiple metal obstacles).

Then, these errors are converted into variance, in order to use them in the filter. In particular, it is better to approximate the errors with a line with two saturation points, accurately choosing the two saturation points at 10^{-4} and 0.1, as depicted in Figure 3.7.

5.3 Tests and Results

This section focuses on the performance evaluation of the different approaches on the collected dataset. Tests were conducted under both LOS and NLOS conditions, where LOS is intended only within the area delimited by the four anchors. Outside this area, the laboratory is filled with potential obstacles, such as people moving, furniture, and metal objects. NLOS experiments are categorized into two types: soft NLOS, involving a single obstacle of any material, and hard NLOS, involving multiple metal obstacles, which represents the worst-case scenario.

LOS conditions

The first set of experiments was conducted under LOS conditions tracing different trajectories, ranging from simple square paths to more random ones. These tests show how the different algorithms perform in an almost ideal scenario, free from obstacles. Comparing these results with the ones coming from more complex and obstructed scenarios allows for a better understanding of the impact of obstacles on the localization performance.

Experiment 1: Square paths

Table 5.1 presents a comparison of different positioning algorithms in terms of their performance metrics, including Root Mean Square Error (RMSE), both along x and y, and on the overall position, and Mean Absolute Position Error (Mean APE). The reported values are obtained as averages from six experiments of the same type.

$Algorithms$	$RMSE_x[cm]$	$RMSE_y[cm]$	$RMSE_{pos}[cm]$	$MeanAPE[cm]$
EKF_{UWB}	23.314	27.337	36.05	31.283
UKF_{UWB}	26.874	23.077	35.462	30.407
$NN + EKF_{UWB}$	23.095	29.407	37.54	33.111
EKF_{SF}	22.93	25.617	34.482	30.475
UKF_{SF}	27.212	22.942	35.649	31.334

Table 5.1: Comparison of different positioning algorithms' performance in terms of RMSE, both along x and y, and on the position, and Mean APE. The reported values are obtained as averages from six experiments of the same type (LOS conditions - square paths).

In summary, the EKF_{SF} algorithm is the one that improves the most the results obtained with the EKF_{UWB}, considered as the standard/base approach. Indeed, it has lower values for all the employed metrics. The other approaches also perform

well and show some improvements, but only for some of the metrics.

Among the six experiments conducted under similar conditions, a further analysis of one of them is provided below. In particular, the experiment labeled as *bag1* is examined.
Figure 5.2 shows the partial trajectory on (x, y) plane generated by the different algorithms, with the ground truth depicted in black. To enhance clarity, only the partial trajectory is represented, rather than the full one consisting of multiple identical loops. The following trajectories appear not very smooth due to a lack of synchronization in the sensor data. Indeed, the employed sensors work at very different frequencies, ranging from the 5 Hz of the UWB antennas to the 40-50 Hz of the odometry and IMU. So, to preserve data integrity, it was preferred not to synchronize them and develop an asynchronous filter that predicts the entire state every time a new measurement arrives, updating only the state variables affected by that specific measurement. As a result, for x and y, there are significantly more predictions than corrections, which causes the trajectory to appear less smooth.

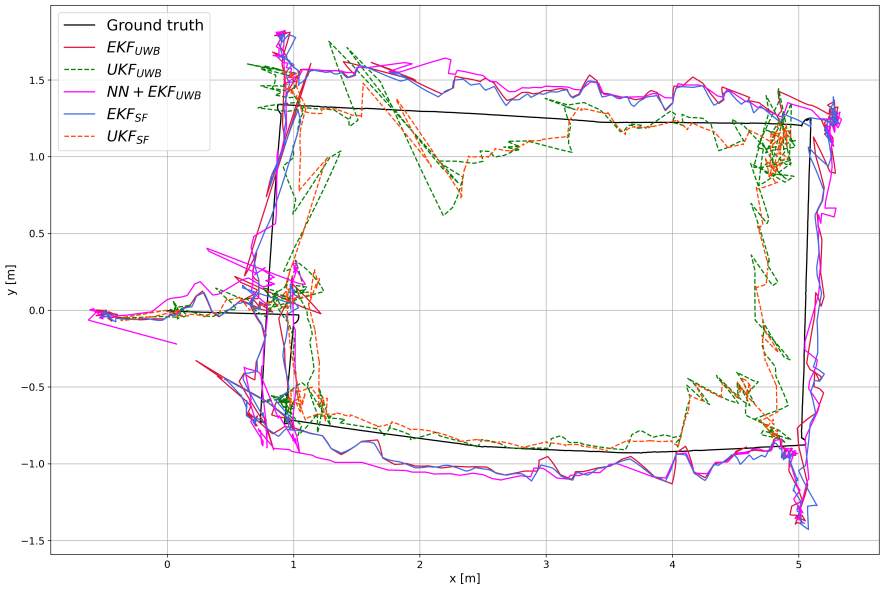
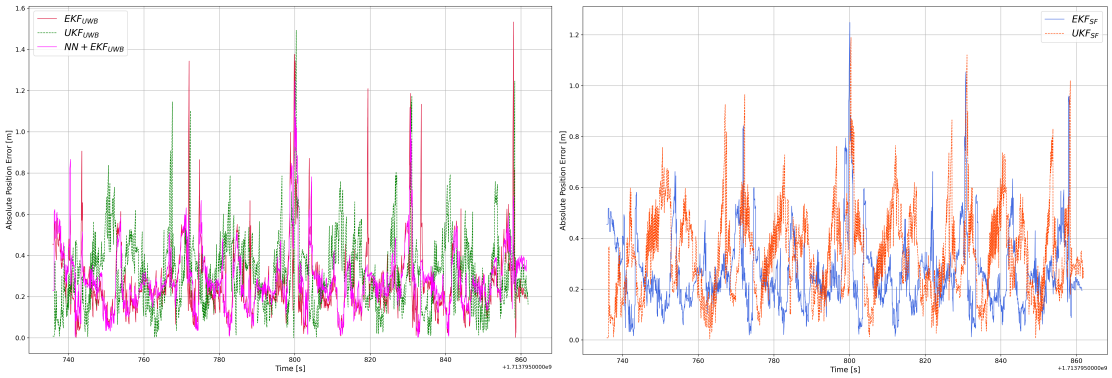


Figure 5.2: Partial trajectory on (x, y) plane - bag1

As said before, outside the area delimited by the four anchors, the laboratory is filled with furniture or people moving, which could influence the UWB signals. This is particularly evident in some areas of the graph, where the trajectories of the algorithms deviate much more from the ground truth compared to other areas, especially when the UGV is near the boundaries.
Figures below show the Absolute Position Error over the time. In particular,

Figure 5.3a represents the APE for the UWB algorithms, while Figure 5.3b for the SF algorithms.



(a) **(b)**

Figure 5.3: (a) Absolute Position Error for UWB algorithms - bag1 (b) Absolute Position Error for SF algorithms - bag1.

A more intuitive plot is the one depicted in Figure 5.4. It shows the Cumulative Distribution Function (CDF) of APE for each algorithm. It tells which is the probability for an algorithm to have an APE smaller or equal to a certain value. The further the curve is to the left, the lower the APE will be. So, for example, for bag1, the algorithm that performs the best is EKF_{SF}.

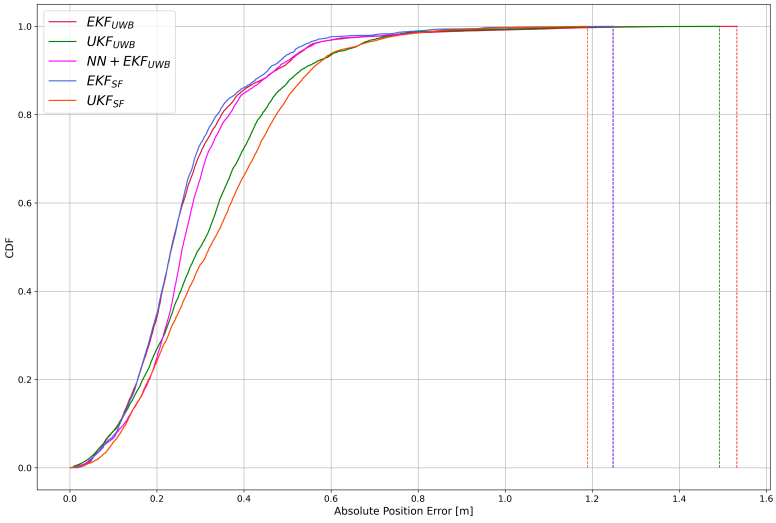


Figure 5.4: Cumulative Distribution Function (CDF) of Absolute Position Error - bag1

Experiment 2: Random trajectories

The second experiment consists in teleoperating the Jackal UGV always under LOS conditions, but tracing some more random trajectories. As in the first experiment, from Table 5.2, it is visible that the EKF_{SF} algorithm is the one that improves the most the results obtained with the EKF_{UWB}.

Algorithms	$RMSE_x[cm]$	$RMSE_y[cm]$	$RMSE_{pos}[cm]$	$MeanAPE[cm]$
EKF_{UWB}	29.906	32.639	44.355	40.667
UKF_{UWB}	31.919	32.240	45.395	40.619
$NN + EKF_{UWB}$	30.97	37.082	48.48	45.075
EKF_{SF}	29.469	31.722	43.43	40.405
UKF_{SF}	33.665	33.895	47.795	44.222

Table 5.2: Comparison of different positioning algorithms' performance in terms of RMSE, both along x and y, and on the position, and Mean APE. The reported values are obtained as averages from six experiments of the same type (LOS conditions - random trajectories).

Among the six experiments conducted under similar conditions, a further analysis of one of them is provided below. In particular, the experiment labeled as *bag12* is examined.

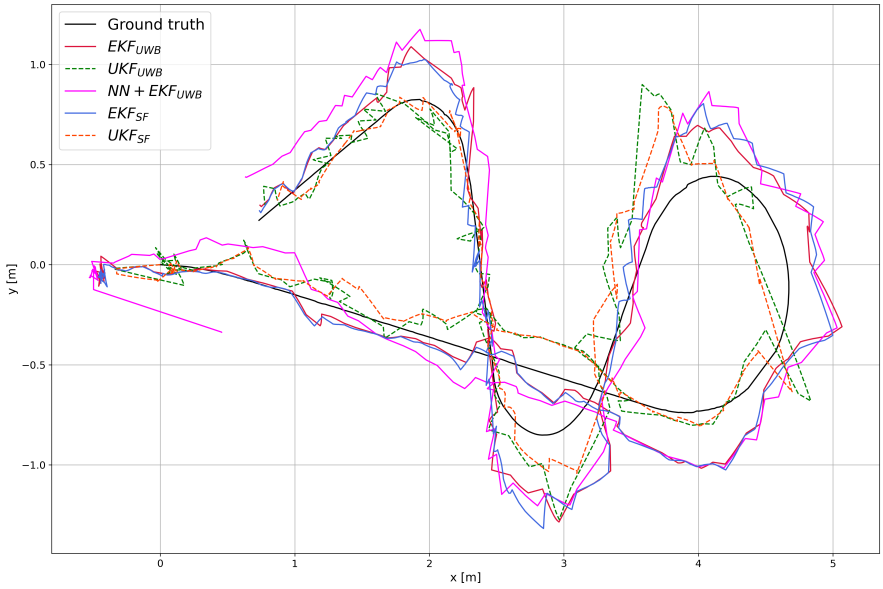


Figure 5.5: Partial trajectory on (x, y) plane - bag12

To enhance clarity, Figure 5.5 shows only the partial trajectory on (x, y) plane generated by the different algorithms, with the ground truth depicted in black. However, the obtained results refer to full trajectory, since the algorithms were executed on the full recording of the experiment. The complete trajectory can be observed in Figure 5.6.

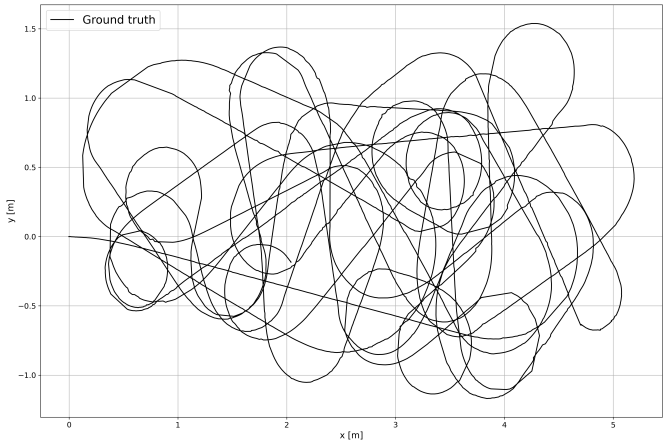
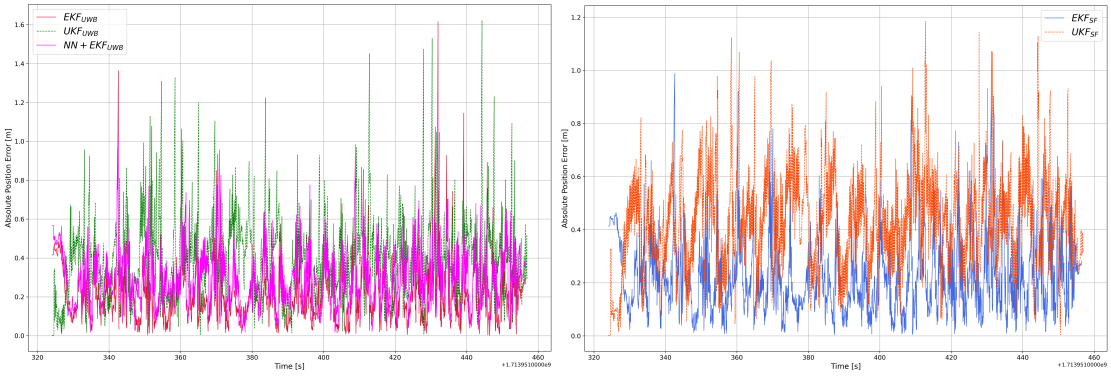


Figure 5.6: Full trajectory on (x, y) plane - bag12

Figures below show the Absolute Position Error over the time. In particular, Figure 5.7a represents the APE for the UWB algorithms, while Figure 5.7b for the SF algorithms.



(a) **(b)**

Figure 5.7: (a) Absolute Position Error for UWB algorithms - bag1 (b) Absolute Position Error for SF algorithms - bag12.

The CDF of APE below shows that, even though EKF_{SF} and EKF_{UWB} have a very similar curve, EKF_{UWB} presents some bigger peak values for the Absolute

Position Error, confirming that, in this particular case, EKF_{SF} performs the best in localizing the rover.

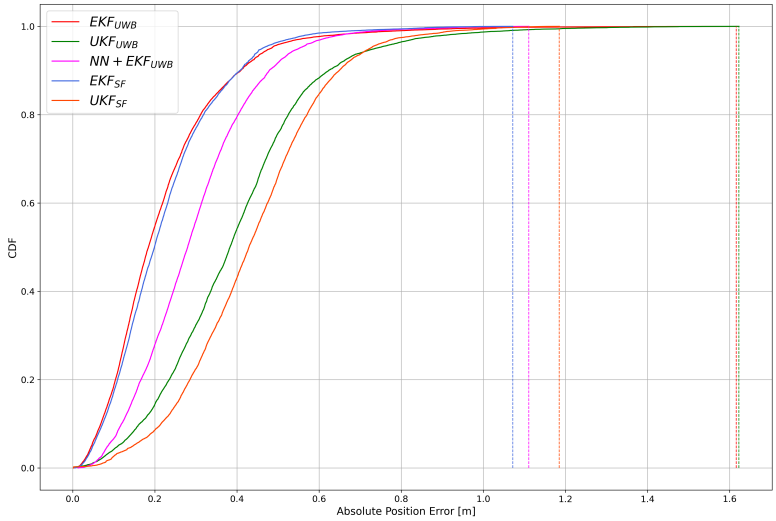


Figure 5.8: Cumulative Distribution Function (CDF) of Absolute Position Error - bag12

Soft NLOS conditions

The second set of experiments was conducted under NLOS conditions, with a single obstacle of different materials, such as metal, cardboard, styrofoam and wood, placed in various positions. These tests show how the different algorithms are influenced by the presence of an obstacle based on its type, and how they perform in a softly obstructed scenario.

Experiment 3: Metal obstacle

The third experiment was conducted in the presence of a metal obstacle, and since radio waves are electromagnetic waves, it is expected to significantly impact the signal, making it less powerful when it reaches the anchor and leading to a worse accuracy in the measurements. In some experimental tests, the scenario is the one depicted in Figure 5.9a, while in others, the only thing that changes is the obstacle position.

Table 5.3 shows that all the analyzed algorithms provide considerable improvements with respect to the EKF_{UWB}. In particular, in terms of $RMSE_{pos}$, EKF_{SF}, UKF_{UWB} and UKF_{SF} perform better, with a decrease of 10 cm with respect to the standard approach. Instead, in terms of Mean APE, the UKF_{UWB} seems to be the best. $NN + EKF_{UWB}$ also improves the results, but less compared to the

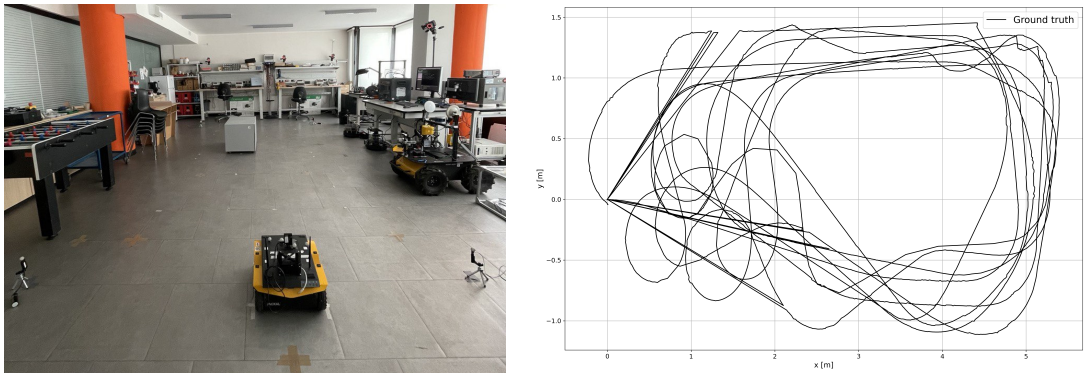
other algorithms. As expected, localization performance is highly influenced by the presence of a metal obstacle, indeed, the values of the different metrics are higher compared to the experiments conducted under LOS conditions.

Algorithms	$RMSE_x[cm]$	$RMSE_y[cm]$	$RMSE_{pos}[cm]$	$MeanAPE[cm]$
EKF_{UWB}	48.052	37.355	61.272	45.663
UKF_{UWB}	35.616	36.198	50.890	41.166
$NN + EKF_{UWB}$	37.344	37.466	53.186	46.413
EKF_{SF}	36.225	34.637	50.343	43.526
UKF_{SF}	38.158	33.308	50.736	43.643

Table 5.3: Comparison of different positioning algorithms' performance in terms of RMSE, both along x and y, and on the position, and Mean APE. The reported values are obtained as averages from six experiments of the same type (Soft NLOS conditions - metal obstacle).

Among the six experiments conducted under similar conditions, a further analysis of one of them is provided below. In particular, the experiment labeled as bag27 is examined.

To enhance clarity, Figure 5.10 shows only the partial trajectory on (x, y) plane generated by the different algorithms, with the ground truth depicted in black. However, the obtained results refer to full trajectory, since the algorithms were executed on the full recording of the experiment. The complete trajectory can be observed in Figure 5.9b.



(a) **(b)**

Figure 5.9: (a) Soft NLOS scenario - metal obstacle centered (b) Full trajectory on (x, y) plane - bag27.

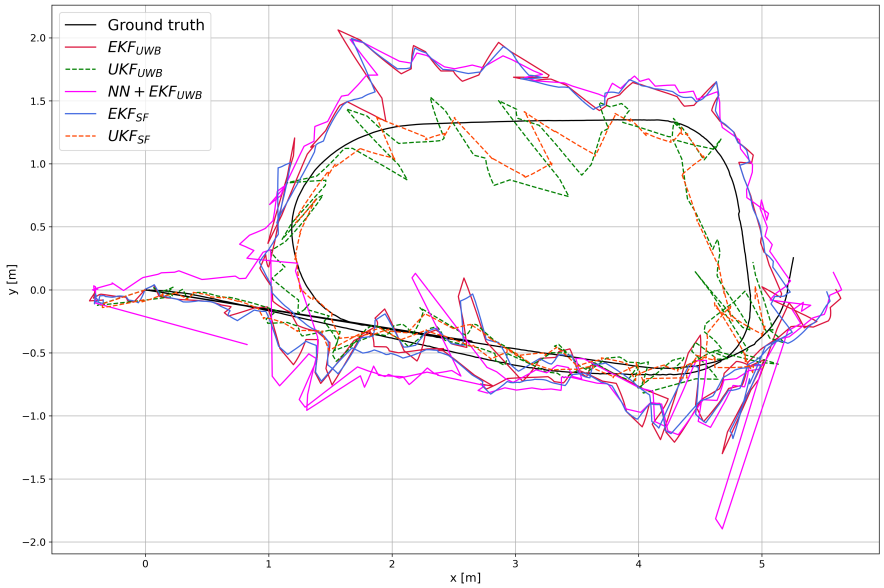
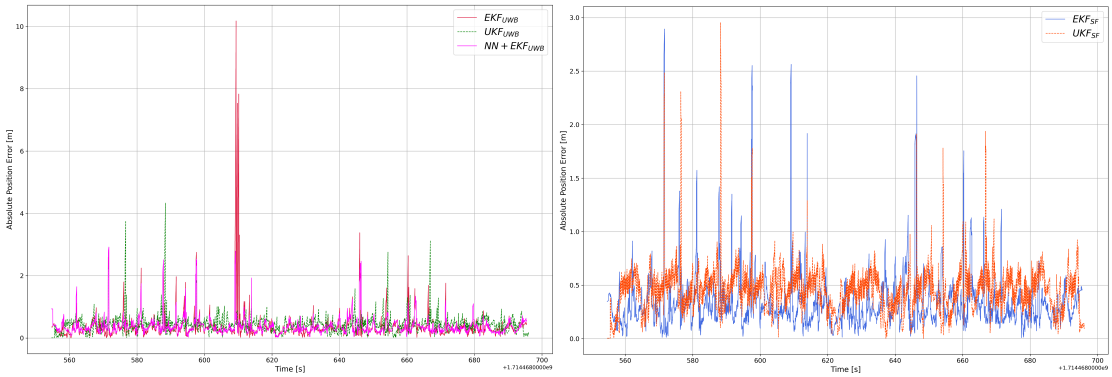


Figure 5.10: Partial trajectory on (x, y) plane - bag27

Figures below show the Absolute Position Error over the time. In particular, Figure 5.11a represents the APE for the UWB algorithms, while Figure 5.11b for the SF algorithms.



(a) **(b)**

Figure 5.11: (a) Absolute Position Error for UWB algorithms - bag1 (b) Absolute Position Error for SF algorithms - bag27.

The CDF of APE below shows that even though the standard approach, EKF_{UWB}, is quite good, it presents some very high peak values for the Absolute Position Error, clearly visible in Figure 5.11a too. All the other analyzed algorithms provide improvements with respect to EKF_{UWB}, about 90% of their error is below

40-50 centimeters, and show even smaller peak values. In particular, EKF_{SF}, $NN + EKF_{UWB}$ and UKF_{SF} seems to be the best in this case.

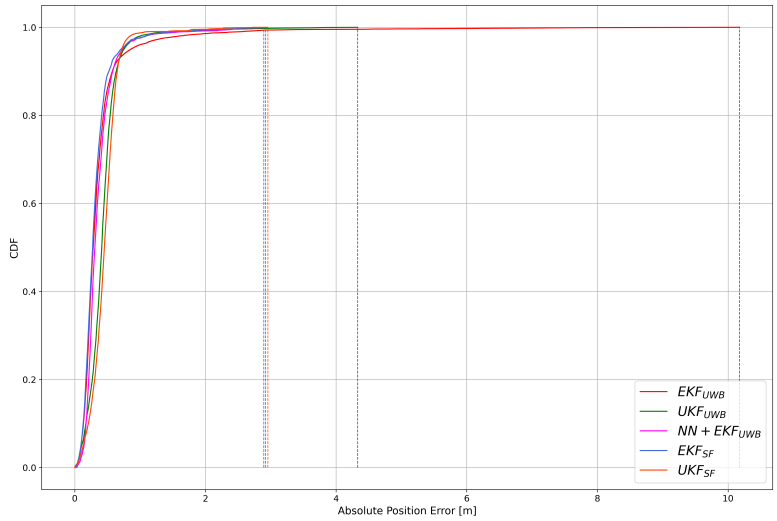


Figure 5.12: Cumulative Distribution Function (CDF) of Absolute Position Error - bag27

Experiment 4: Cardboard obstacle

The fourth experiment was conducted in the presence of a cardboard obstacle centered in the area delimited by the four anchors, as depicted in Figure 5.14a.

As expected, Table 5.4 shows significantly lower values compared to the experiment involving a metal obstacle, as cardboard causes less interference with radio waves. In particular, in this scenario, EKF_{UWB} and EKF_{SF} algorithms show the best performance, localizing the rover with an error smaller than 40 centimeters.

Algorithms	$RMSE_x[cm]$	$RMSE_y[cm]$	$RMSE_{pos}[cm]$	$MeanAPE[cm]$
EKF_{UWB}	28.596	27.343	39.658	36.446
UKF_{UWB}	35.626	29.068	45.986	40.635
$NN + EKF_{UWB}$	30.834	31.39	44.119	40.865
EKF_{SF}	28.401	27.801	39.841	37.045
UKF_{SF}	38.204	29.512	48.281	44.326

Table 5.4: Comparison of different positioning algorithms' performance in terms of RMSE, both along x and y, and on the position, and Mean APE. The reported values are obtained as averages from six experiments of the same type (Soft NLOS conditions - cardboard obstacle).

Among the six experiments conducted under similar conditions, a further analysis of one of them is provided below. In particular, the experiment labeled as bag32 is examined.
To enhance clarity, Figure 5.13 shows only the partial trajectory on (x, y) plane generated by the different algorithms, with the ground truth depicted in black. However, the obtained results refer to full trajectory, since the algorithms were executed on the full recording of the experiment. The complete trajectory can be observed in Figure 5.14b.

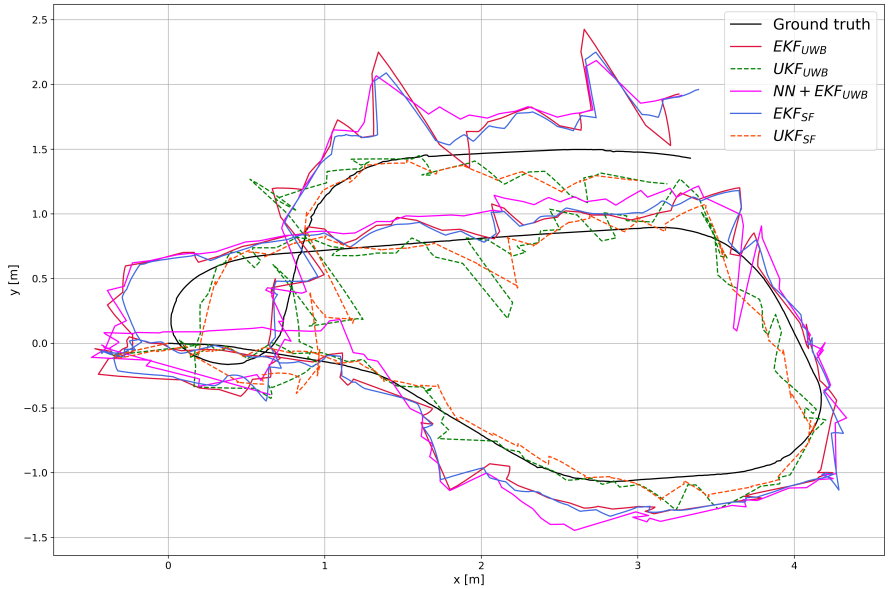
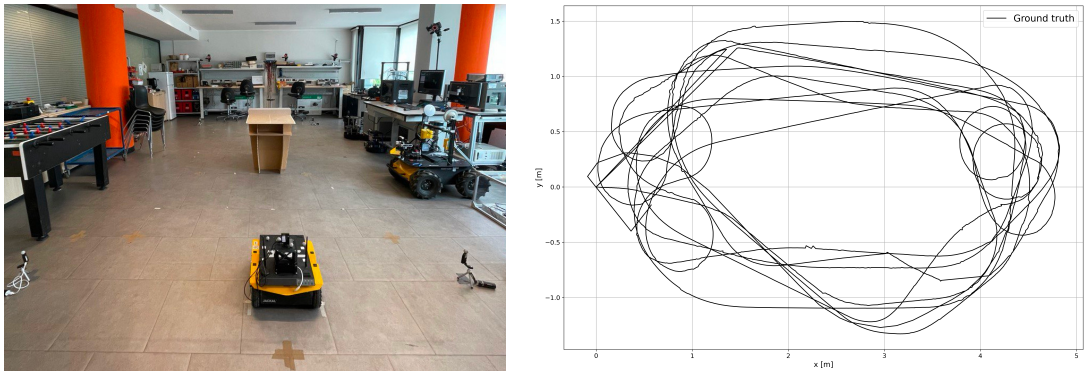


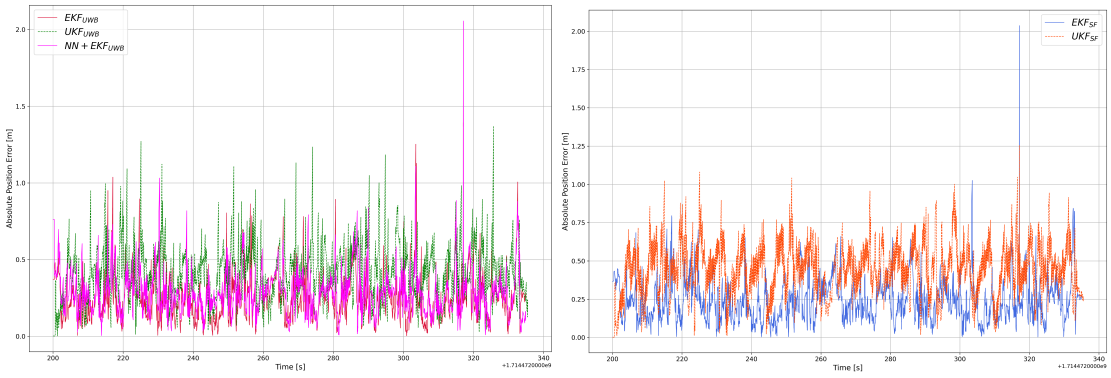
Figure 5.13: Partial trajectory on (x, y) plane - bag32



(a) **(b)**

Figure 5.14: (a) Soft NLOS scenario - cardboard obstacle centered (b) Full trajectory on (x, y) plane - bag32.

Figures below show the Absolute Position Error over the time. In particular, Figure 5.15a represents the APE for the UWB algorithms, while Figure 5.15b for the SF algorithms.



(a) **(b)**

Figure 5.15: (a) Absolute Position Error for UWB algorithms - bag1 (b) Absolute Position Error for SF algorithms - bag32.

The CDF of APE below confirms that, in this particular case, EKF_{SF} and EKF_{UWB} algorithms performs the best in localizing the rover.

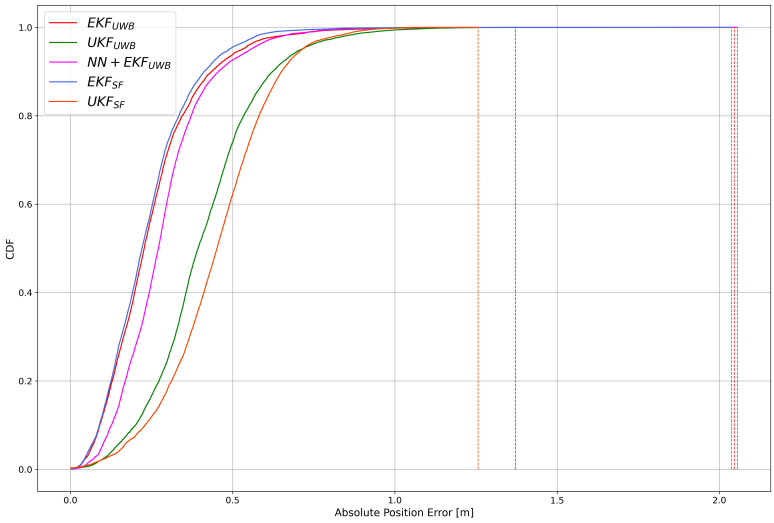


Figure 5.16: Cumulative Distribution Function (CDF) of Absolute Position Error - bag32

Experiment 5: Styrofoam obstacle

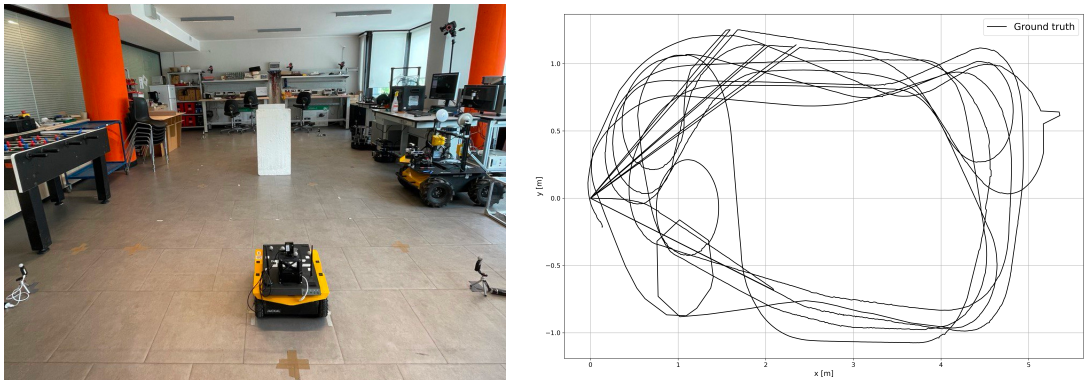
The fifth experiment involves a styrofoam obstacle centered in the area delimited by the four anchors, as depicted in Figure 5.17a.

Table 5.5 shows values that are very similar to those obtained in the experiment with the cardboard obstacle, as styrofoam has minimal effect on the radio waves too. EKF_{SF} is the only algorithm that slightly improves the results obtained with the standard EKF_{UWB}. So, as in the fourth experiment, these two algorithms are the ones that perform the best.

$Algorithms$	$RMSE_x[cm]$	$RMSE_y[cm]$	$RMSE_{pos}[cm]$	$MeanAPE[cm]$
EKF_{UWB}	34.538	30.575	46.189	40.292
UKF_{UWB}	38.774	28.694	48.25	41.931
$NN + EKF_{UWB}$	36.434	33.167	49.325	44.042
EKF_{SF}	34.108	30.361	45.722	40.684
UKF_{SF}	41.529	30.149	51.332	46.141

Table 5.5: Comparison of different positioning algorithms' performance in terms of RMSE, both along x and y, and on the position, and Mean APE. The reported values are obtained as averages from six experiments of the same type (Soft NLOS conditions - styrofoam obstacle).

Among the six experiments conducted under similar conditions, a further analysis of one of them is provided below. In particular, the experiment labeled as bag39 is examined.



(a) **(b)**

Figure 5.17: (a) Soft NLOS scenario - styrofoam obstacle centered (b) Full trajectory on (x, y) plane - bag39.

To enhance clarity, Figure 5.18 shows only the partial trajectory on (x, y) plane generated by the different algorithms, with the ground truth depicted in black. However, the obtained results refer to full trajectory, which can be observed in Figure 5.17b.

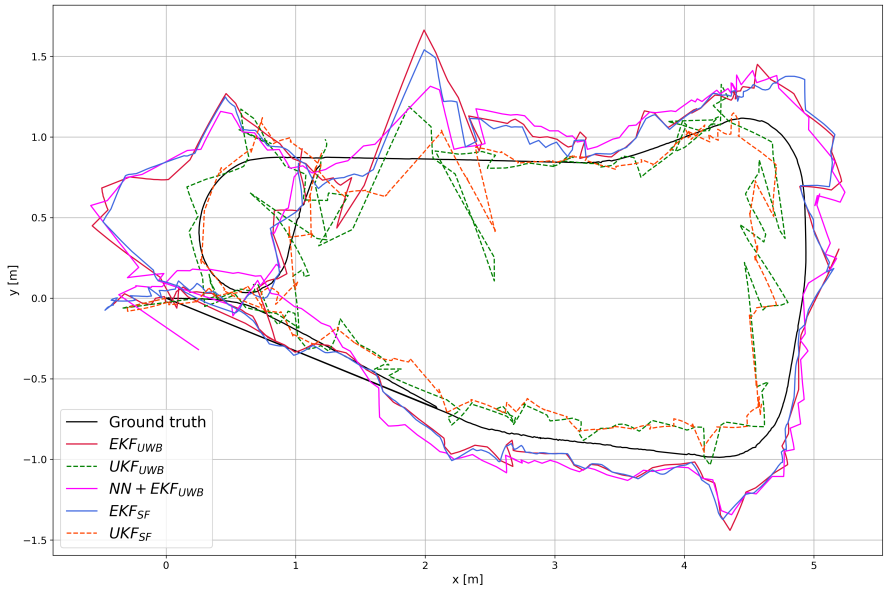
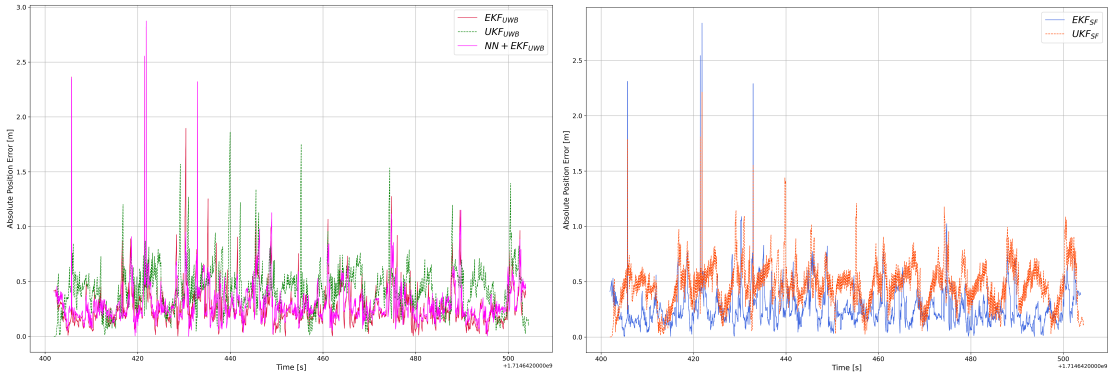


Figure 5.18: Partial trajectory on (x, y) plane - bag39

Figures below show the Absolute Position Error over the time. In particular, Figure 5.19a represents the APE for the UWB algorithms, while Figure 5.19b for the SF algorithms.



(a) **(b)**

Figure 5.19: (a) Absolute Position Error for UWB algorithms - bag1 (b) Absolute Position Error for SF algorithms - bag39.

The CDF of APE below confirms that, in this particular case, EKF_{SF} and EKF_{UWB} algorithms performs the best in localizing the rover.

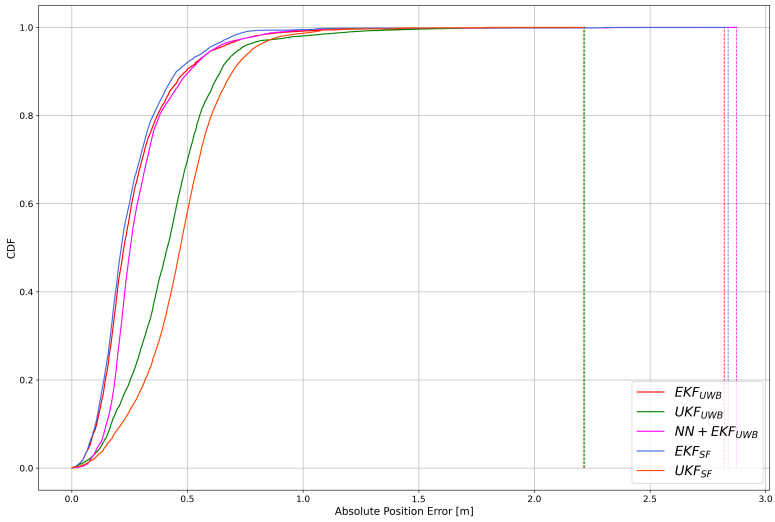


Figure 5.20: Cumulative Distribution Function (CDF) of Absolute Position Error - bag39

Experiment 6: Wooden obstacle

The sixth experiment was conducted in the presence of a wooden obstacle, which is expected to interfere with the radio waves a bit more than cardboard and styrofoam, but less than metal. Various scenarios were recreated by placing the obstacle in different positions, including centering it within the area delimited by the four anchors or placing it directly in front of one of the anchors, as depicted in Figure 5.22a.

Algorithms	$RMSE_x[cm]$	$RMSE_y[cm]$	$RMSE_{pos}[cm]$	$MeanAPE[cm]$
EKF_{UWB}	43.653	34.952	56.523	43.555
UKF_{UWB}	35.685	28.786	45.902	39.219
$NN + EKF_{UWB}$	37.349	36.023	52.007	45.975
EKF_{SF}	35.718	31.984	48.113	42.261
UKF_{SF}	38.093	29.794	48.41	43.296

Table 5.6: Comparison of different positioning algorithms' performance in terms of RMSE, both along x and y, and on the position, and Mean APE. The reported values are obtained as averages from six experiments of the same type (Soft NLOS conditions - wood obstacle).

Table 5.6 shows that, in terms of $RMSE_{pos}$, all the analyzed algorithms provide considerable improvements with respect to EKF_{UWB}, specifically the UKF_{UWB}. In terms of Mean APE, instead, only the $NN + EKF_{UWB}$ algorithm does not provide any improvements, leading to a worsening of the results, probably due to a more variability in the y-direction accuracy. In contrast, the UKF_{UWB} algorithm also demonstrates strong performance in terms of Mean APE.

Among the six experiments conducted under similar conditions, a further analysis of one of them is provided below. In particular, the experiment labeled as bag53 is examined.

To enhance clarity, Figure 5.21 shows only the partial trajectory on (x, y) plane generated by the different algorithms, with the ground truth depicted in black. However, the obtained results refer to full trajectory, since the algorithms were executed on the full recording of the experiment. The complete trajectory can be observed in Figure 5.22b.

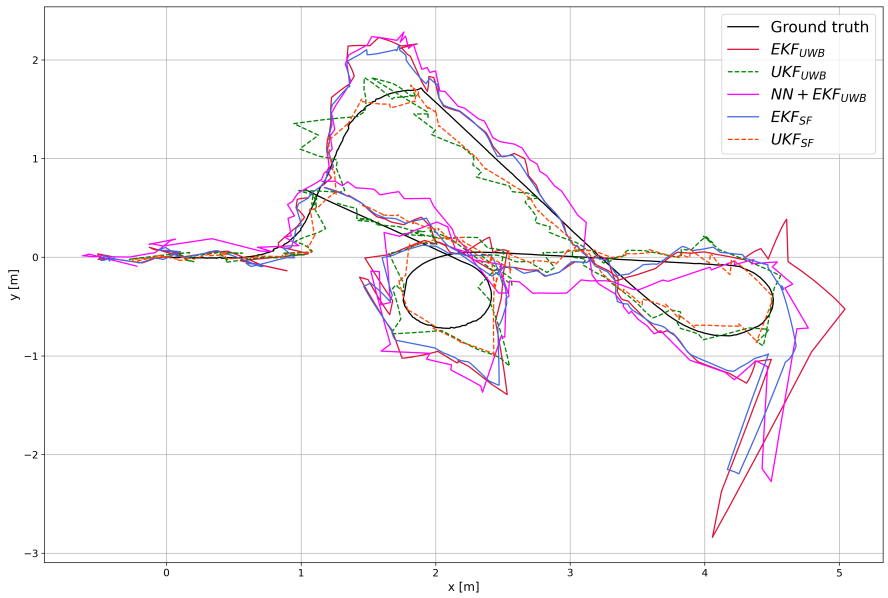
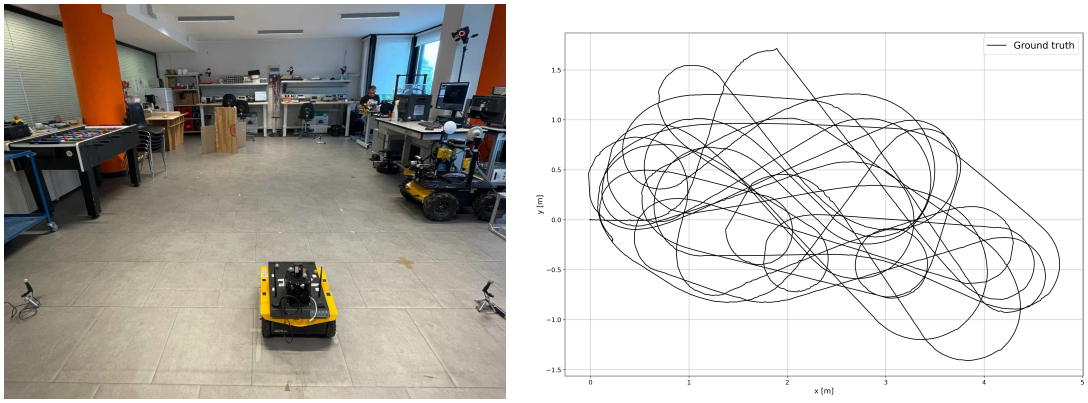


Figure 5.21: Partial trajectory on (x, y) plane - bag53

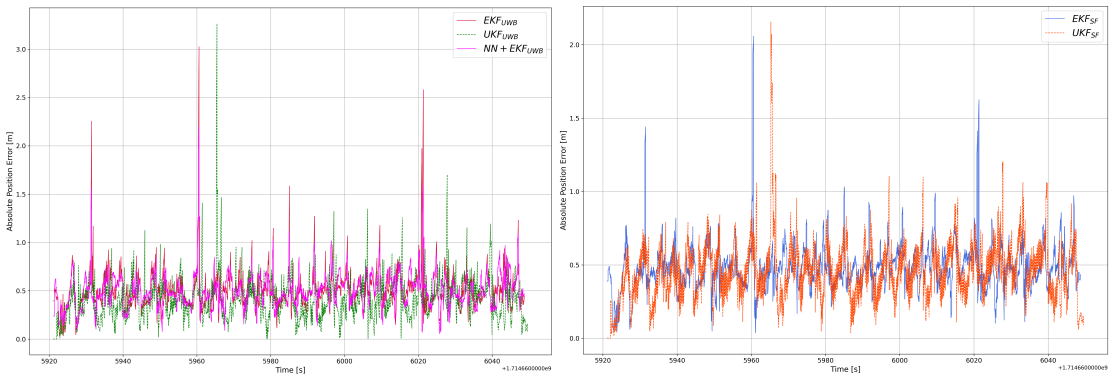
Regarding Figure 5.21, as for the previous experiments, only in some areas of the graph the filters significantly deviate from the ground truth, due to the surrounding environment. The UKF_{UWB} and UKF_{SF} are the algorithms that most closely follow the ground truth.



(a) **(b)**

Figure 5.22: (a) Soft NLOS scenario - wood obstacle centered (b) Full trajectory on (x, y) plane - bag53.

Figures below show the Absolute Position Error over the time. In particular, Figure 5.23a represents the APE for the UWB algorithms, while Figure 5.23b for the SF algorithms.



(a) **(b)**

Figure 5.23: (a) Absolute Position Error for UWB algorithms - bag1 (b) Absolute Position Error for SF algorithms - bag53

The CDF of APE below shows that, in this particular case, UKF_{UWB}, EKF_{SF} and UKF_{SF} algorithms perform the best.

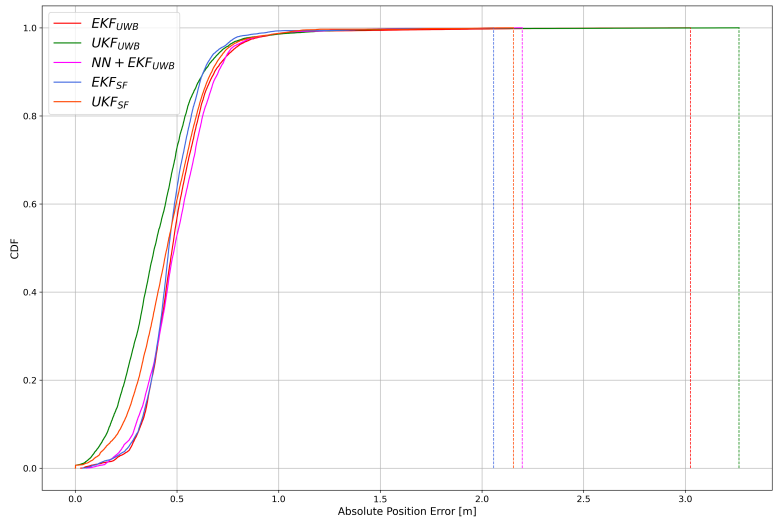


Figure 5.24: Cumulative Distribution Function (CDF) of Absolute Position Error - bag53

Hard NLOS conditions

The third and last set of experiments was conducted under hard NLOS conditions, thus multiple metal obstacles were involved. In particular, a total of six experiments were performed, two for each obstacle configuration. Due to the significant presence of metal in the environment, a further worsening of the results is expected.

Experiment 7: Multiple metal obstacles

Algorithms	$RMSE_x[cm]$	$RMSE_y[cm]$	$RMSE_{pos}[cm]$	$MeanAPE[cm]$
EKF_{UWB}	61.299	61.353	87.505	76.783
UKF_{UWB}	52.185	52.717	74.48	62.141
$NN + EKF_{UWB}$	62.385	63.567	89.75	79.361
EKF_{SF}	59.896	56.692	83.218	74.734
UKF_{SF}	51.749	52.151	73.867	61.375

Table 5.7: Comparison of different positioning algorithms' performance in terms of RMSE, both along x and y, and on the position, and Mean APE. The reported values are obtained as averages from six experiments of the same type (Hard NLOS conditions - multiple metal obstacles).

Table 5.7 shows that most of the algorithms provide considerable improvements with respect to EKF_{UWB}. In particular, UKF_{UWB} and UKF_{SF} show the lowest

values for all the considered metrics, decreasing both the $RMSE_{pos}$ and Mean APE of around 15 centimeters. Although the improvement is minimal (around 1–2 cm), sensor fusion still enhances accuracy, especially in more challenging scenarios like this one. $NN + EKF_{UWB}$ algorithm, instead, is the only one that, for this particular case, does not seem to provide any improvements.

Among the six experiments conducted under similar conditions, a further analysis of one of them is provided below. In particular, the experiment labeled as bag59 is examined.

To enhance clarity, Figure 5.25 shows only the partial trajectory on (x, y) plane generated by the different algorithms, with the ground truth depicted in black. However, the obtained results refer to full trajectory, since the algorithms were executed on the full recording of the experiment. The complete trajectory can be observed in Figure 5.26b.
The significant divergence between the estimated trajectories and the ground truth highlights the impact of the four obstacles present in the environment, which heavily influence the localization performance, leading to a worsening of the results. This suggests that UWB-based localization struggles in this specific setup, depicted in Figure 5.26a, due to the signal interference caused by the obstacles.

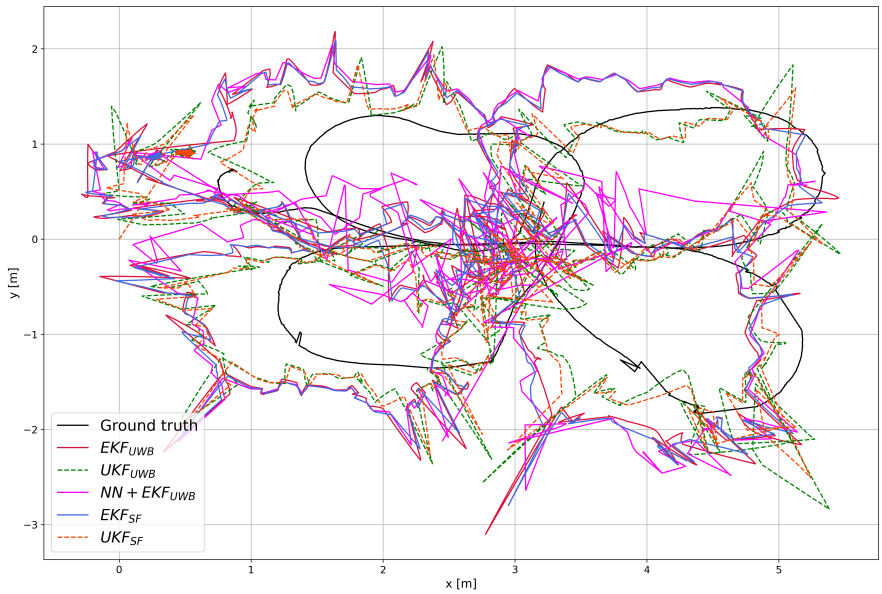
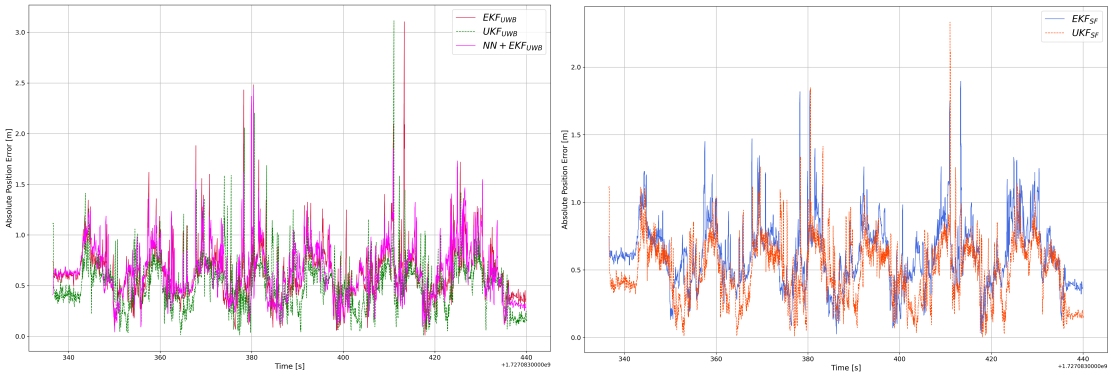


Figure 5.25: Partial trajectory on (x, y) plane - bag59

(a) **(b)**

Figure 5.26: (a) Hard NLOS scenario - multiple metal obstacles (b) Full trajectory on (x, y) plane - bag59

Figures below show the Absolute Position Error over the time. In particular, Figure 5.27a represents the APE for the UWB algorithms, while Figure 5.27b for the SF algorithms. As expected, the APEs are much higher compared to the previous experiments, with the UKF_{UWB} and UKF_{SF} showing the lowest ones.



(a) **(b)**

Figure 5.27: (a) Absolute Position Error for UWB algorithms - bag1 (b) Absolute Position Error for SF algorithms - bag59

The CDF of APE below shows that the EKF_{SF} slightly improves the overall performance compared to the EKF_{UWB}. In contrast, both the UKF_{UWB} and UKF_{SF}, present a much better curve, significantly shifted to the left. With these two algorithms, about 90% of the errors is below 60-65 centimeters, whereas for the other algorithms, about 90% of the errors falls below 1 meter.

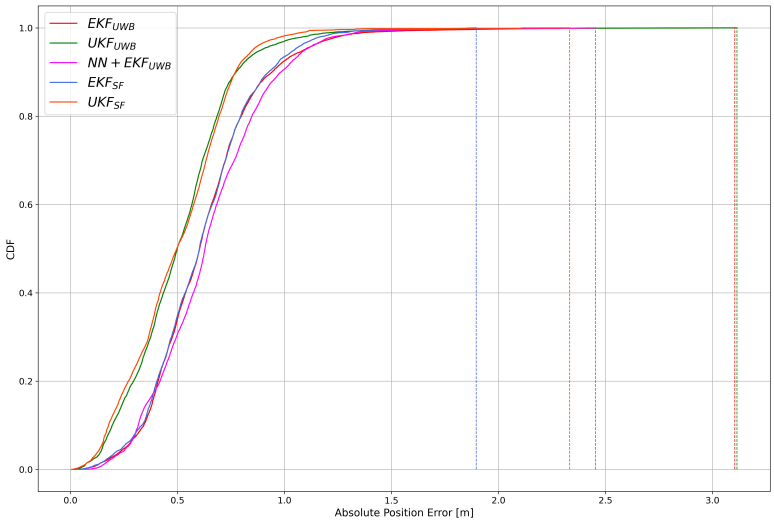


Figure 5.28: Cumulative Distribution Function (CDF) of Absolute Position Error - bag59

Chapter 6

Conclusions and Future Works

This work aimed to implement and analyze different localization algorithms for precisely tracking a UGV by exploiting the UWB technology, including standard approaches as EKF_{UWB} and UKF_{UWB} algorithms, sensor fusion-based variants (EKF_{SF} and UKF_{SF}), and an innovative approach ($NN + EKF_{UWB}$) for a more adaptive localization.

The results obtained with the experimental tests demonstrate that, for the EKF, sensor fusion can significantly enhance localization performance. Indeed, in all the performed experiments, it achieved better results compared to the EKF_{UWB}. In particular, under LOS conditions and in the presence of a cardboard or styrofoam obstacle, that slightly influences radio waves, it provides an improvement of a few centimeters. While in cases with metal or wooden obstacles the improvement in $RMSE_{pos}$ was around 10 centimeters, reaching up to 40 centimeters in the worst-case scenario. For the UKF, instead, sensor fusion was only effective in more complex situations, with multiple obstacles, showing the benefits of integrating additional sensors to mitigate the errors caused by the UWB signal reflections and distortions. In particular, for the last and most complex experiment, the UKF_{SF} algorithm performs even better than the EKF_{SF}, making it the optimal choice for this case. However, one drawback of the UKF is that, as it is implemented, it generates a lot of numerical errors, in particular regarding the covariance matrix P that easily becomes non-positive definite, limiting its use. The last and most innovative approach, the $NN + EKF_{UWB}$ algorithm, also provides some improvements compared to EKF_{UWB} in several experiments, but not as much as the EKF_{SF}. Moreover, due to its even higher computational complexity, it turns out not to be the best choice. However, this approach should not be excluded,

as it may offer valuable insights with further testing and optimization in future works. In particular, in this work, the anchors were placed on the floor, so raising them on pedestals and increasing their number could reduce the interference from ground-level obstacles and lead to more precise measurements. Consequently, this could provide cleaner data for the training of the NN, improving the model's accuracy and the precision of the resulting errors.

Appendix A

Jackal UGV

The Clearpath Jackal UGV is a four wheeled, skid-steering mobile robot (SSMR). In this kind of robots there is no explicit steering mechanism. Indeed, each pair of wheels is actuated by a single motor, in order to control the movement and the steering by changing the velocity and the direction of rotation of each pair of wheels. For example, if the wheels on the left side turn more slowly with respect to the wheels on the right side, the robot will turn left, while if the wheels on the left side move forward and the wheels on the right side backward, the robot will rotate in place.
For SSMRs, slippage is essential. When a pair of wheels moves more slowly or in opposite direction with respect to the other one, the wheels on the slower or opposite side will slip, or skid, on the surface, leading to a small lateral movement. The combination of versatile movement, adaptability to different surfaces, and robust design makes skid-steering mobile robots well-suited for all-terrain applications.

A.1 Kinematic Model

In this section the mathematical model of a skid-steering mobile robot is provided, assuming it moves on a planar surface.
Two reference frames (RFs) are required: a fixed reference frame (X_g, Y_g, Z_g) and a body reference frame (x_l, y_l, z_l) with the origin located at the center of mass (COM) of the robot itself. The coordinates of the COM in the fixed RF are (X, Y, Z), where Z is a constant since the robot only moves on the plane. Therefore, the linear and angular velocities are expressed as follows: $\mathbf{v}=[v_x, v_y, 0]^T$, $\mathbf{w}=[0, 0, w]^T$. If $\mathbf{q}=[X, Y, \theta]^T$ denotes the vector of generalized coordinates, $\dot{\mathbf{q}}=[\dot{X}, \dot{Y}, \dot{\theta}]^T$ denotes the vector of generalized velocities.
The following equation represents the relationship between $\dot{X}$ and $\dot{Y}$, along with

the local velocity components v_x and v_y.

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{A.1}$$

Figure A.1 shows the body diagram of the robot, depicting the same relationship. Moreover, in the planar case, $\dot{\theta}$=w.

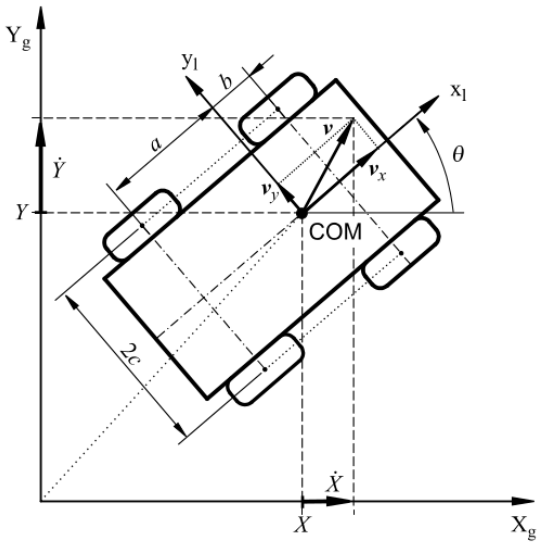


Figure A.1: Free body diagram. [26]

Equation A.1 describes the free-body kinematics. No restrictions on the SSMR plane movement are imposed, so the relationship between wheel velocities and local velocities should be better analyzed.
Each wheel rotates with an angular velocity $w_i(t)$, where i=1, 2, ..., 4, and has a negligible thickness and a single contact point, P_i, with the plane.
Despite the other wheeled vehicles, the lateral velocity of the SSMR is generally nonzero, due to the slippage phenomenon typical of these robots. However, for simplicity, a negligible slip between the wheels and the surface is considered.
So, the following equation can be derived:

$$v_{ix} = r_i w_i \tag{A.2}$$

where v_{ix} is the x component of the velocity vector of the i-th wheel in the local frame, and r_i the rolling radius of that wheel. In the figure below, the radius vectors $d_i = [d_{ix} \;\; d_{iy}]^T$ and $d_C = [d_{Cx} \;\; d_{Cy}]^T$ are represented in the local reference frame from the instantaneous center of rotation (ICR).

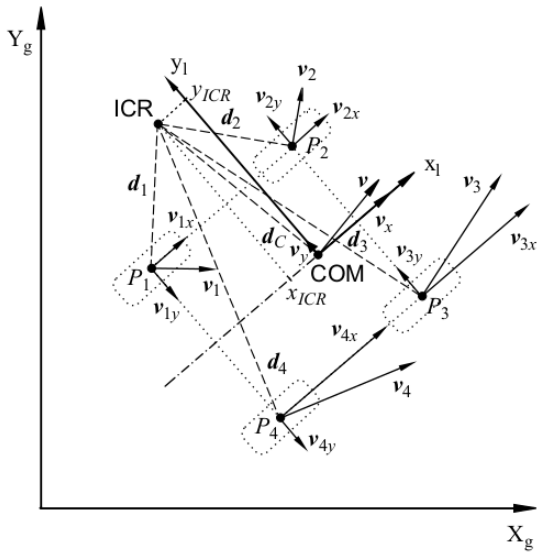


Figure A.2: Wheel velocities. [26]

By using geometry, the following equations have been derived:

$$\frac{\|v_i\|}{\|d_i\|} = \frac{\|v\|}{\|d_C\|} = |w| \tag{A.3}$$

that, written in a more detailed form, becomes:

$$\frac{v_{ix}}{-d_{iy}} = \frac{v_x}{-d_{Cy}} = \frac{v_{iy}}{d_{ix}} = \frac{v_y}{d_{Cx}} = w \tag{A.4}$$

Since the coordinates of the ICR are $(-d_{xC}, -d_{yC})$, equation A.4 can be rewritten as:

$$\frac{v_x}{y_{ICR}} = -\frac{v_y}{x_{ICR}} = w \tag{A.5}$$

Moreover, from Figure A.2 the following relationships arise:

$$\begin{aligned} d_{1y} &= d_{2y} = d_{Cy} + c \\ d_{3y} &= d_{4y} = d_{Cy} - c \\ d_{1x} &= d_{4x} = d_{Cx} - a \\ d_{2x} &= d_{3x} = d_{Cx} + b \end{aligned} \tag{A.6}$$

"where a,b and c are positive kinematic parameters", as written in [26]. Combining A.4 and A.6 the following equalities can be derived:

$$\begin{aligned} v_L &= v_{1x} = v_{2c} \\ v_R &= v_{3x} = v_{4x} \\ v_F &= v_{2y} = v_{3y} \\ v_B &= v_{1y} = v_{4y} \end{aligned} \tag{A.7}$$

"where v_L and v_R denote the longitudinal coordinates of the left and right wheel velocities, v_F and v_B are the lateral coordinates of the velocities of the front and rear wheels, respectively", as stated in [26].
Additional relationships can be obtained by combining A.4 and A.7:

$$\begin{bmatrix} v_L \\ v_R \\ v_F \\ v_B \end{bmatrix} = \begin{bmatrix} 1 & -c \\ 1 & c \\ 0 & -x_{ICR} + b \\ 0 & -x_{ICR} - a \end{bmatrix} \begin{bmatrix} v_x \\ w \end{bmatrix} \tag{A.8}$$

and assuming $r_i = r$ for each wheel:

$$w_w = \begin{bmatrix} w_L \\ w_R \end{bmatrix} = \frac{1}{r} \begin{bmatrix} v_L \\ v_R \end{bmatrix} \tag{A.9}$$

"where w_L and w_R are the angular velocities of the left and right wheels, respectively" [26].
By using A.8 and A.9, the relationship between the angular velocities of the wheels and the velocities of the robot, is the following one:

$$\eta = \begin{bmatrix} v_x \\ w \end{bmatrix} = r \begin{bmatrix} \frac{w_L + w_R}{2} \\ \frac{-w_L + w_R}{2c} \end{bmatrix} \tag{A.10}$$

The accuracy of A.10 highly depends on the slippage phenomenon, decreasing as long as the slippage increases. To complete the kinematic model of the SSMR, the following velocity constraint is necessary:

$$v_y + x_{ICR}\dot{\theta} = 0 \tag{A.11}$$

since in not integrable, it represents a nonholonomic constraint. In the Pfaffian form becomes:

$$\begin{bmatrix} -\sin\theta & \cos\theta & x_{ICR} \end{bmatrix} \begin{bmatrix} \dot{X} & \dot{Y} & \dot{\theta} \end{bmatrix}^T = \mathbf{A}(\mathbf{q})\dot{\mathbf{q}} = \mathbf{0} \tag{A.12}$$

Since $\dot{\mathbf{q}}$ is in the null space of $\mathbf{A}$, the kinematic model of a SSMR is:

$$\dot{\mathbf{q}} = \mathbf{S}(\mathbf{q})\eta \tag{A.13}$$

where $\mathbf{S^T}(\mathbf{q})\mathbf{A^T}(\mathbf{q}) = \mathbf{0}$ and $\mathbf{S(q)}$ is defined as follows:

$$\mathbf{S}(\mathbf{q}) = \begin{bmatrix} \cos\theta & x_{ICR}\sin\theta \\ \sin\theta & -x_{ICR}\cos\theta \\ 0 & 1 \end{bmatrix} \tag{A.14}$$

For in depth analysis, including the dynamical model and the drive model of the SSMR, see [26].

Bibliography

[1] H. Nikookar and R. Prasad. *Introduction to Ultra Wideband for Wireless Communications*. Berlin: Springer, 2009, p. 188. ISBN: 9781402066320 (cit. on pp. 4, 7, 9, 10).

[2] T. Zwick, W. Wiesbeck, J. Timmermann, and G. Adamiuk. *Ultra-wideband RF System Engineering*. Cambridge: Cambridge University Press, 2013, p. 185. ISBN: 9781107015555 (cit. on pp. 5, 7–10).

[3] D. N. Malayeri. «Review of UWB technology specifications and benefits as a powerful technology to develop power distribution grid automation techniques». In: *16th Electrical Power Distribution Conference*. Bandar Abbas, Iran, 2011, pp. 1–7 (cit. on p. 6).

[4] N. C. Zhi H. Arslan and M.-G. Di Benedetto. *Ultra Wideband Wireless Communication*. 2006, p. 500. ISBN: 9786468600 (cit. on p. 6).

[5] Paolo Dabove, Vincenzo Di Pietra, Marco Piras, Ansar Abdul Jabbar, and Syed Ali Kazim. «Indoor positioning using Ultra-wide band (UWB) technologies: Positioning accuracies and sensors' performances». In: *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. 2018, pp. 175–184. DOI: 10.1109/PLANS.2018.8373379 (cit. on p. 7).

[6] Lian Sang C., Adams M., Hörmann T., Hesse M., Porrmann M., and Rückert U. «Numerical and Experimental Evaluation of Error Estimation for Two-Way Ranging Methods». In: *Sensors (Basel, Switzerland)* 19.3 (Feb. 2019). DOI: 10.3390/s19030616 (cit. on p. 9).

[7] Ershad Ali. «Global Positioning System (GPS): Definition, Principles, Errors, Applications DGPS». In: (Apr. 2020) (cit. on p. 11).

[8] *GEOTAB blog*. URL: https://www.geotab.com/blog/what-is-gps/ (cit. on p. 11).

[9] *Official site of Pozyx*. URL: https://www.pozyx.io/newsroom/uwb-versus-other-technologies#uwb-vs-gps (cit. on pp. 11–13).

[10] *Official site of LINK LABS*. URL: https://www.link-labs.com/blog/bluetooth-vs-bluetooth-low-energy (cit. on p. 12).

[11] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005, p. 492. ISBN: 0262201623 (cit. on p. 15).

[12] J.G. Webster, S. Huang, and G. Dissanayake. «Robot Localization: An Introduction». In: *Wiley Encyclopedia of Electrical and Electronics Engineering*. 2016. DOI: 10.1002/047134608X.W8318 (cit. on p. 15).

[13] Qiang Li, Ranyang Li, Kaifan Ji, and Wei Dai. «Kalman Filter and Its Application». In: *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*. Tianjin, China, 2015, pp. 74–77. DOI: 10.1109/ICINIS.2015.35 (cit. on p. 16).

[14] Roger R. Labbe Jr. *Kalman and Bayesian Filters in Python*. GitHub (2020); eBook (Creative Commons Licensed), 2020, p. 506 (cit. on pp. 17, 18, 20).

[15] Alex Becker. *Kalman Filter from the Ground Up*. 2023. ISBN: 978-965-93120-1-6 (cit. on p. 17).

[16] G. Calafiore. *Neural Networks*. Politecnico di Torino: Lecture slides, 2024 (cit. on pp. 21–23).

[17] Nikhil Bhargav. *Neurons in Neural Networks*. Mar. 2024. URL: https://www.baeldung.com/cs/neural-networks-neurons (cit. on p. 23).

[18] Grossi Enzo and Buscema Massimo. «Introduction to artificial neural networks». In: *European journal of gastroenterology hepatology* 19 (Jan. 2008). DOI: 10.1097/MEG.0b013e3282f198a0 (cit. on p. 23).

[19] Umberto Albertin, Alessandro Navone, Mauro Martini, and Marcello Chiaberge. «Semi-Supervised Novelty Detection for Precise Ultra-Wideband Error Signal Prediction». In: (Apr. 2024) (cit. on pp. 23–25).

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016 (cit. on p. 24).

[21] *Official site of Clearpath Robotics - Jackal datasheet*. URL: https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/ (cit. on pp. 27, 28).

[22] *Official site of Qorvo*. URL: https://www.qorvo.com/products/p/DWM1001C (cit. on pp. 29, 30).

[23] G. Galati. *Robotics - ROS*. Politecnico di Torino: Lecture slides, 2023 (cit. on p. 32).

[24] John D'Errico. *Nearest SPD*. https://www.mathworks.com/matlabcentral/fileexchange/42885-nearestspd. [Accessed: 3-Oct-2024]. 2013 (cit. on p. 39).

[25] N. J. Higham. «Computing a nearest symmetric positive semidefinite matrix». In: *Numerische Mathematik* 52.3 (1988). https://doi.org/10.1016/0024-3795(88)90223-6, pp. 399–408. DOI: 10.1016/0024-3795(88)90223-6 (cit. on p. 39).

[26] Krzysztof Kozłowski and Dariusz Pazderski. «Modeling and control of a 4-wheel skid-steering mobile robot». In: *International Journal of Applied Mathematics and Computer Science*. Jan 2004 (cit. on pp. 63–66).