FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Sensor Fusion for Mobile Robot Localization using UWB and ArUco Markers

Sílvia Dolores Nogueira Faria

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Paulo José Cerqueira Gomes da Costa

Second Supervisor: José Luís Sousa Magalhães Lima

July 27, 2021

Resumo

Uma das principais necessidades de um robô autónomo é este ser capaz de se localizar, em tempo real e no seu ambiente, ou seja, saber a sua posição e orientação. Para obter a localização de um robô é possível recorrer a diferentes metodologias, no entanto existem metodologias que apresentam problemas em diferentes circunstâncias. Um desses problemas é a incerteza na deteção do robô devido a ruído presente nos sensores utilizados. Assim, com o intuito de obter uma localização mais robusta do robô e mais tolerante a falhas é possível combinar diversos sistemas de localização, combinando assim as vantagens de cada um deles.

Neste trabalho, é utilizado o sistema Pozyx, uma solução de baixo custo que fornece informação de posicionamento, mais especificamente medidas de distância, com o auxílio da tecnologia Ultra-WideBand Time of Flight (UWB ToF). Também são utilizados marcadores ArUco dispersos pelo ambiente, que através da sua identificação por uma câmara incorporada no robô é também possível obter informação de posicionamento, incluindo medidas de distância e de ângulo. Estas duas soluções foram estudadas e implementadas num robô móvel diferencial, através de um esquema de localização baseado em marcadores. Primeiramente, foi realizada uma caracterização do erro de ambos os sistemas, uma vez que as medidas não são perfeitas, havendo sempre algum ruído nas medições. De seguida, as medidas fornecidas pelos sistemas, mais especificamente as medidas de distância fornecidas pelo sistema Pozyx e as medidas de ângulo e distância obtidas através dos marcadores ArUco, são fundidas com os valores da odometria do robô através da implementação de um Filtro de Kalman Estendido (EKF), obtendo assim uma estimação da pose do robô. Um sistema de localização adicional é também desenvolvido, com o intuito de obter o Ground-Truth do robô. Para tal, um marcador ArUco é colocado no topo do robô, marcador este que é detetado por uma câmara colocada no teto. A partir da deteção do marcador é depois possível determinar a pose do robô com o auxílio da biblioteca ArUco. Posteriormente, a pose estimada pelo Filtro de Kalman é comparada com a pose fornecida pelo sistema de Ground-Truth, determinando assim a precisão do algoritmo de localização desenvolvido. Neste trabalho, diversos testes de localização foram realizados em dois ambientes, o simulado, em que no caso, o simulador SimTwo foi utilizado, e o real.

O trabalho desenvolvido mostrou que com a utilização do sistema Pozyx e de marcadores ArUco é possível melhorar a localização do robô, o que significa que é uma solução adequada e eficaz para este fim.

Abstract

One of the main needs of a truly autonomous robot is to be able to locate itself, in real time, in its environment, that is, to know its position and orientation. To obtain the localization of a robot it is possible to use different methodologies, however there are methodologies that present problems in different circumstances.One of these problems is the uncertainty in the sensing of the robot due to noise present in the sensors used. In order to obtain a more robust localization of the robot and more fault tolerant it is possible to combine several localization systems, thus combining the advantages of each one.

In this work, the Pozyx system is used, a low-cost solution that provides positioning information, more specifically distance measurements, through Ultra-WideBand Time of Flight (UWB ToF) technology. ArUco markers scattered around the environment are also used, which through their identification by a camera built into the robot, it is also possible to obtain positioning information, including distance and angle measurements. These two solutions were studied and implemented on a differential mobile robot, using a beacon-based scheme. First, an error characterization of the two systems was performed, since the measurements are not perfect, and there is always some noise in the measurements. Next, the measurements provided by the systems, specifically the distance measurements provided by the Pozyx system and the angle and distance measurements obtained through the ArUco markers, are fused with the robot's odometry data through the implementation of an Extended Kalman Filter (EKF), thus obtaining an estimation of the robot's pose. An additional localization system is also developed, in order to obtain the Ground-Truth of the robot. For this purpose, an ArUco marker is placed on top of the robot, which is detected by a camera placed on the ceiling. Once the marker is detected, it is then possible to determine the robot's pose with the aid of the ArUco library. Afterwards, the pose estimated by the Kalman Filter is compared with the pose provided by the Ground-Truth system, thus determining the accuracy of the localization algorithm developed. In this work, several localization tests were performed in two environments, the simulated one, in which case the SimTwo simulator was used, and the real one.

The developed work showed that with the use of the Pozyx system and ArUco markers it is possible to improve the robot localization, meaning that it is an adequate and effective solution for this purpose.

Agradecimentos

Em primeiro lugar gostaria de agradecer aos meus orientadores Professor Paulo Costa e Professor José Lima por todo o apoio, orientação, paciência e tempo dispensado ao ajudarem-me ao longo de todo este semestre. Apesar da época atípica que vivemos, eles ajudaram-me sempre em tudo o que precisei sem qualquer hesitação, tendo sido muito importante para o sucesso desta dissertação.

Em segundo lugar gostaria de agradecer aos meus familiares, sobretudo aos meus pais e tios que sempre me apoiaram em tudo, tanto nos bons momentos como naqueles menos bons. Acredito que sem o apoio deles não teria conseguido chegar até aqui.

Por fim, mas não menos importante, queria agradecer ao meu namorado e respetivos pais, e às minhas amigas mais próximas por todo o amor, força e por todos os incríveis momentos que passamos juntos ao longo desta grande jornada. Adoro-vos!

Obrigado por tudo,

Sílvia Faria

“Success is not the key to happiness.
Happiness is the key to success.
If you love what you are doing, you will be successful.”

Albert Schweitzer

Contents

List of Figures

List of Tables

Abbreviations

AoA	Angle of Arrival
API	Application Programming Interface
AR	Augmented Reality
DoF	Degrees of Freedom
EKF	Extended Kalman Filter
FCC	Federal Communications Commission
FFD	Full Function Device
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IR	Infrared
IrDA	Infrared Data Association
KF	Kalman Filter
LOS	Line-of-Sight
LPS	Local Positioning Systems
MCL	Monte Carlo Localization
NLOS	Non-Line-of-Sight
ODE	Open Dynamics Engine
RF	Radio Frequency
RFD	Reduced Function Device
RFID	Radio Frequency Identification
ROS	Robot Operating System
RSS	Received Signal Strength
RSSI	Received Signal Strength Indicator
SLAM	Simultaneous Localization And Mapping
TDoA	Time Difference of Arrival
ToA	Time of Arrival
ToF	Time of Flight
UDP	User Datagram Protocol
UKF	Unscented Kalman Filter
UWB	Ultra-WideBand
WLAN	Wireless Local Area Network
WPANs	Wireless Personal Area Networks

Chapter 1

Theme Introduction

The ability to locate a robot is one of the main needs for it to be truly autonomous. Robot localization is the process of determining where it is located, its position and orientation, in its environment [16].

Since the first mobile robot was invented, the problem of pose determination has been investigated by many researches and developers around the world due to its complexity and the multitude of possible approaches.

Different methodologies can be used to determine robots location as accurately as possible, however presents several problems in some circumstances. One of these problems is the existence of uncertainty in the sensing of the robot. To solve this is necessary to combine the uncertain information properly [17]. In this way is possible to have a system that allows a more robust location of the robot, more tolerant to failures and disturbances.

This dissertation concerns the implementation of an accurate localization system, whose goal is to improve the pose estimation of mobile robots.

1.1 Context and Motivation

In a semi-structured environment a mobile robot needs to be able to locate itself without human intervention, i.e. autonomously. Being that if the robot knows his own pose (position and orientation) means that is truly autonomous to execute given tasks [18].

Robot localization answer to the following question: “Where am I?”. By answering this question and finding a reliable solution to this problem, location of a mobile robot, it is possible to answer the following questions: "Where am I going?" and "How should I get there?". These three questions, presented previously, summarize the navigation problem of a robot [19].

The localization methods can be classified in two great categories: relative methods and absolute methods [19, 20]. Relative localization methods gives the robot’s pose relative to the initial one and for this purpose dead reckoning methods such as odometry and inertial navigation are used. Odometry is a technique in which it is common to use encoders connected to the rotation axes of the robot wheels. The basic idea of odometry is the integration of information from the mobile robot during a certain period of time, which leads to the accumulation of errors with the distance traveled by the robot [20].

Absolute localization methods gives the global pose of the robot and do not need previously calculated poses. In these methods, data from odometry and data from external sensors are combined to estimate the robot's pose [19, 20].

In an indoor environment, a mobile robot needs to localize itself accurately because of the limited space and obstacles present in space. Today, the number of applications which relay on indoor localization is rapidly increasing and because of this, localization of a robot in an indoor environment it has become in an active research area.

The localization methods can also be classified according to the application context and are divided into two groups: indoor state estimation and outdoor state estimation [21]. Global Positioning System (GPS) is usually used at present in outdoor state estimation, nevertheless the use of GPS in indoor state estimation is difficult. Due to this, different alternative technologies has been proposed and developed to solve this. Among these technologies that has been proposed, Ultra-WideBand (UWB) is known for its low-cost, high precision and easy deployment [21]. An UWB based localization system can be characterized by an UWB tag present in the robot that can be located by measuring the distance till UWB anchors, whose position is known. Since UWB based localization systems cannot provide robot orientation information, another way to achieve this information is required. To solve this problem is possible to use odometry information to achieved the orientation of the robot. Another solutions to locate the robot are based on the visual field, whose aim is to find the correspondence between the real world and the image projection. To achieve this goal, methods based on binary squared fiducial markers have been widely utilized, such as ArUco markers [22]. So, all the information can be fused to combine the advantages of each system, and if one fails, the estimate localization can be improved through others. This fusion of information is accomplished through the use of a filter, where an Extended Kalman Filter (EKF) will be used, since a non-linear system is being handled. This method increases robustness, scalability and accuracy of location [23].

1.2 Objectives

Following the context and motivations addressed in the previous Section (Section 1.1), the main objectives of this dissertation are:

- Study the different localization systems and algorithms available;
- Study the different methodologies for sensor fusion;
- Development of the localization systems;
- Development of the sensor fusion algorithm;
- Implement the methodology developed in SimTwo simulation environment;

- Implement the methodology developed in real environment using a mobile robot provided, i.e a localization system based on odometry, UWB ToF and ArUco markers;
- Validation of the developed systems/algorithms through testing and analyze the results obtained.

1.3 Document Structure

The document is divided into seven chapters, to describe the work carried out throughout the dissertation. Following Chapter 1 (Theme Introduction), the document has been organized as follows:

- Chapter 2 that presents the Literature Review for indoor positioning technologies, with a special emphasis to the UWB and Image Based technologies, fusion algorithms and robotic simulators;
- Chapter 5 presents the system architecture used in this work, demonstrating its structure used to address the problem and presenting and describing the software, hardware and simulation environment;
- Chapter 4 describes the selected approaches to deal with the robot localization problem and explains their theoretical implementation;
- Chapter 5 describes the different performance and localization tests performed;
- Chapter 6 demonstrates and discusses the results obtained in the different tests;
- Chapter 7 lists some final conclusions and presents considerations regarding future work.

Chapter 2

Literature Review

This chapter presents the literature review on indoor localization technologies, fusion algorithms and robotic simulators. First, Section 2.1 have an introduction to Indoor Positioning Systems and then discusses the positioning algorithms, techniques and technologies, with a special emphasis to the UWB and Image Based technologies, since are the technologies used in the implementation. Section 2.2 analyses the algorithms that are used in fusion and estimations problems. Lastly, Section 2.3 presents some robotic simulators, including its features.

2.1 Indoor Positioning Systems

Positioning systems can be classified as: Global Positioning Systems (GPS - most popular) and Local Positioning Systems (LPS).

Global Positioning Systems refer to systems that provide localization information through the use of satellites. In outdoor environments it is possible to obtain the localization of the users with an accuracy of a few meters. However, it provides good operation for outdoor environments only, in an indoor environment and in harsh environments it fails because signals of GPS cannot penetrate through buildings and that are hugely affected by indoor surroundings, becoming impractical for many indoor environments. The indoor environment is more complex due to the existence of obstacles that influence the propagation of the signals and due to the interference added to the propagation signal by noise sources from other wireless networks [24].

With Local Positioning Systems it is possible to obtain localization information through base stations or anchors that can generate beacon signals. The coverage of this type of positioning systems is restricted and localization is obtained only inside the coverage area of the network. Local Positioning Systems can be classified according to different network criteria: computation, environment and medium for transmission. In particular to the environment criteria, Local Positioning Systems can be categorized into three categories: outdoor (outside buildings), indoor (inside buildings) and underwater [24].

Depending on the type of application, the type of positioning technologies is different in order to satisfy all the needs and constraints of the application. Focusing on the theme of the work in point, it is about applications on indoor environments and because of this, this study will focus on Indoor Positioning System (IPS) and technologies used, that have become very popular in recent years. The indoor positioning is very interesting and although there are already a number of studies on this area, it remains an active and attractive research area because there is a great market opportunity for technologies in this domain [24, 1].

An indoor positioning system determine the position of an object or a person in a physical indoor space continuously and in real-time, and has a lot of different applications, such as home, public buildings and medical applications [15]. IPSs use a various positioning approaches that vary considerably in terms of accuracy, cost, precision, technology, scalability, robustness and security. So, depending on the application, the requirements are different and because of this there are different performance metrics that should be evaluated. This metrics and their definition are presented in the Table 2.1

Table 2.1: Performance metrics of IPSs [15]

Metric	**Definition**
Accuracy	Closeness between the true and measured value of a given measure.
Availability	The availability of the positioning service in terms of percentage of time.
Coverage Area	The area covered by an indoor positioning system.
Cost	Can be measured in terms of: money, time, energy and space. Can be affected at different levels of the system: system installation and maintenance, infrastructure components and positioning devices.
Scalability	The degree to which the system ensures the normal positioning function when it scales in one of two dimensions: number of users and geography.
Privacy	Access control over how users' personal information is collected and used.

As already verified, different characteristics distinguish indoor positioning from outdoor positioning. Indoor environments may rely on Non-Line-of-Sight (NLOS) propagation, which due to obstacles in the environment, signals cannot travel directly in a straight line from a emitter to a receiver [2], as shown in Figure 2.1. Due to this, the receiver may have to deal with inconsistent time delays. The presence of obstacles causes high attenuation and signal dispersion.

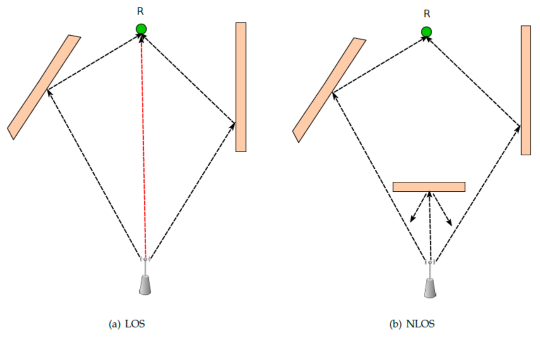


Figure 2.1: (a) Line-of-Sight and (b) Non-Line-of-Sight [1]

To deal with small areas and obstacles, which is common in indoor environments, indoor positioning requires higher accuracy compared to outdoor positioning [2].

2.1.1 Indoor Positioning Algorithms

Positioning algorithms indicate how to estimate the position of a given object. More specifically, this algorithms specify how the signal properties that are captured can be measured and then converted to a given position. Signal properties are geometrical parameters consisting of metrics such as angle and distance to measure an object's position using calculations [1]. There are four basic indoor positioning algorithms: Angle of Arrival (AoA), Time of Arrival (ToA), Time Difference of Arrival (TDoA) and Received Signal Strength Indication (RSSI), which will be presented below.

2.1.1.1 Angle of Arrival (AoA)

In the AoA algorithm, the estimation of the signal reception angles, from at least two sources (reference points), is compared with either the signal amplitude or carrier phase across multiple antennas. The location can be found using triangulation, where there is intersection of the angle line for each signal source, as can be seen in the Figure 2.2. In a 2D-dimension plane, AoA algorithm requires only two sources to estimate the position of a given object, while in a 3D-dimension plane requires three sources. However, to improve the accuracy of this algorithm more sources are used for position estimation [1, 15].

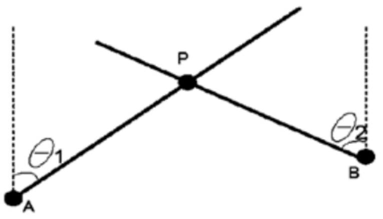


Figure 2.2: AoA algorithm [1]

The accuracy of this type of AoA algorithms tends to decrease when the distance between the emitter and receiver increases. These algorithms also have a great complexity when compared to others, having a hardware that tends to be complex and expensive [1, 15].

2.1.1.2 Time of Arrival (ToA)

ToA, which is also called Time of Flight (ToF) is relative to the circle intersection for multiple transmitters, and the radius of the circles is the distance between the transmitter and the receiver. The distance is determined by calculating the one-way propagation time between the transmitter and receiver. So, the measurement of ToA is mainly distance-based. It is important to note that for this algorithm it is necessary to synchronize the time of all transmitters. This algorithm provides high accuracy but at a cost of higher hardware complexity [1, 15].

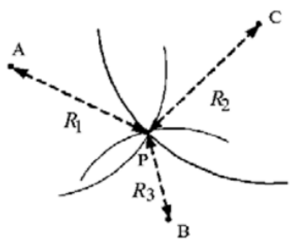


Figure 2.3: ToA algorithm [1]

A mobile device sends a time-stamped signal in the direction of the receiving beacons. When the signal is received the distance between the receiver and the transmitter is calculated using the speed and the measured time. Subsequently, the localization of the target object is estimated through triangulation [2].

2.1.1.3 Time Difference of Arrival (TDoA)

Similar to ToA, TDoA algorithm is also distance-based. In this algorithm the determination of the relative position of a mobile transmitter is done using the difference in the propagation time of the arrival of the transmitter and the multiple reference points (receivers). This means that TDoA measures the difference in ToA at two different reference points, thus eliminating the necessity of knowing when the signal was transmitted. Since TDoA eliminates the modification of the transmitter to the absolute time of arrival, it reduces its complexity. Furthermore, the TDoA also offers high precision [1, 15].

So, in a TDoA algorithm, the various receivers receive a transmission, whose start time is unknown. It should be noted that only the receivers require time synchronization. A hyperbolic curve in the localization space is thus generated through the difference in measurement of the arrival time. The possible localizations for the target object are determined from the intersection of the multiple hyperbolic curves, however only one of these intersections will represent the localization of the object. Therefore some previous knowledge is required to eliminate the position ambiguity [1, 15, 2]. This algorithm is illustrated in the Figure 2.4.

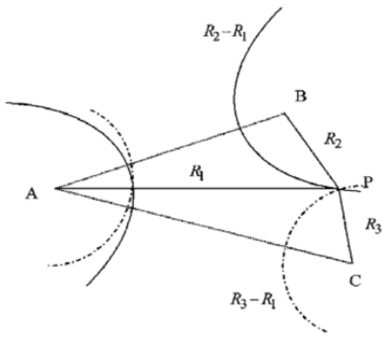


Figure 2.4: TDoA algorithm [1]

2.1.1.4 Received Signal Strength Indication (RSSI)

RSSI is a measure of power level of the Received Signal Strength (RSS) present in a radio infrastructure that can be used to estimate distance between the transmitters and receivers. More specifically, this algorithm takes measurements of the attenuation of the transmitted signal in order to calculate the decrease or loss of signal strength because of the propagation. Then the distance between transmitters and receivers can be estimated, and from this estimate, position information can be obtained [1, 15]. Figure 2.5 demonstrates this algorithm.

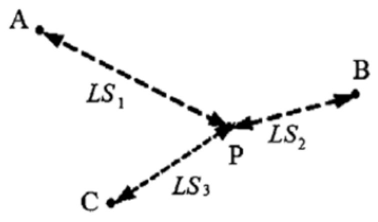


Figure 2.5: RSSI algorithm [1]

Since in indoor environments there are several obstacles, it is difficult to obtain Line-of-Sight. Therefore, RSSI and positioning can be subject to multi path and shadow effects, which leads to a decrease in accuracy, showing that this method is not good for IPS [1, 15].

2.1.2 Indoor Positioning Technologies

There are several indoor positioning technologies and this technologies can be used at the same time with the objective of reach the advantages of each one. An indoor environment may require different quality attributes, depending on the application in question. Because of this, the indoor positioning system to be used must be chosen properly [1, 2].

Over the years different researchers have been classifying indoor positioning technologies in many different ways. In the articles [1, 2] present these different ways of classifying the indoor positioning technologies and the respective researchers who proposed the classification. In addition to these classification proposals, the authors of this articles suggest a new classification for these technologies. This classification is made depending on the infrastructure of the system that uses them and can be seen in the Figure 2.6. The new classification divide the technologies into two main classes: building dependent and building independent. The first one refer to technologies that rely on the building in which they would operate and on existing technology in the building or on the map and structure of the building. The second refers to technologies that don’t need any specific hardware in a building, i.e. dead reckoning and image-based technologies. Building dependent indoor positioning technologies can be classified into two main classes, indoor positioning that required dedicated infrastructure and indoor positioning technologies that use the building’s infrastructure. Depending on the structure of the buildings, it is possible to understand whether the use of dedicated infrastructure is necessary. For example, most buildings have Wi-Fi, while very few have Radio Frequency Identification (RFID) [1, 2].

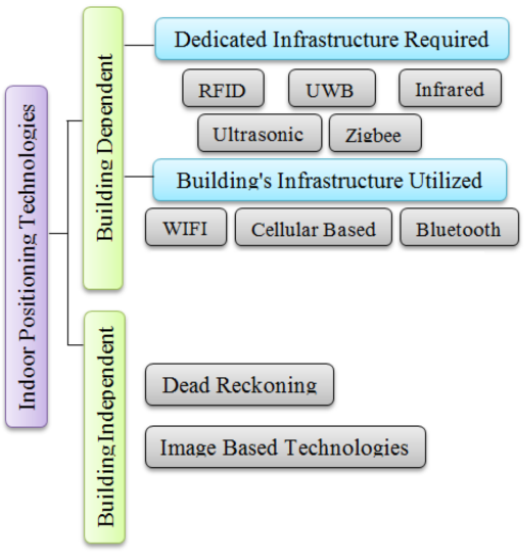


Figure 2.6: Classification of indoor positioning technologies [1, 2]

In the following subsections each of the technologies present in the diagram in Figure 2.6 will be introduced. In particular, more emphasis will be given to the UWB and Image based technologies as it will be the main focus of this work.

2.1.2.1 Ultra-WideBand (UWB)

Ultra-WideBand (UWB) is a recent and accurate technology that has emerged as a viable candidate for precise indoor positioning due to its distinctive characteristics. The Federal Communications Commission (FCC) defines UWB as a Radio Frequency signal occupying a portion of the frequency spectrum that is greater than 20 % of the center carrier frequency, or has a bandwidth greater than 500 MHz [15, 2]. UWB technology transmits very short pulses and uses techniques that cause radio energy to spread across a wide frequency band with a low power spectral density [15].

In UWB technology the use of time-based positioning algorithms such as TDoA and ToA are the most widely used, since they exploit the high temporal resolution provided by the large bandwidths. The AoA algorithm can be a challenge because of a number of distinct identified paths [15].

UWB technology can be used in several application areas, including communication, positioning and tracking. Regarding positioning, the use of UWB can provide real time precision tracking for various applications, including locators for emergency services and tracking of people or objects. So, UWB signals provide an accurate estimate of position and location for indoor environments [15, 25].

Advantages of using UWB

Regarding the work to be developed (localization of mobile robots), UWB can be a useful technology, due to its low cost and great time domain. Due to the larger bandwidth of the radio spectrum (greater than 500 MHz) and the sub-nanosecond duration pulses, it is not only robust to the interference of other types of RF signals, but can also discriminate the different multipath components of the received signal, thus being better than other technologies [26]. In addition, the lowest frequencies of the UWB spectrum can penetrate through a variety of materials (walls and objects), which makes it especially interesting for indoor applications. Typically, this technology can provide for the majority of applications an accuracy level of centimeter as opposed to the most traditional narrow band technologies such as Wi-Fi and Bluetooth, that only have a meter level of accuracy [15, 25, 27, 1].

Challenges of using UWB

Although UWB has several advantages, it also has some limitations and challenges in its use. The variation of time measurements in a Non-Line-of-Sight environments is typically much greater than in a Line-of-Sight environments. Indeed, the error based on the ToA range can be modeled as a Gaussian LOS distribution, however for the NLOS case, it adds an exponentially distributed random variable. Because of this difference, an accurate characterization of the system becomes more difficult [28]. In conclusion, clear LOS is the ideal scenario and is when the system has both the best range and the best accuracy [28].

As already mentioned, one of the advantages of using the UWB is that it can penetrate a variety of materials. However, when a signal goes against an obstacle, one part partially penetrates the object, another is absorbed, and the rest is reflected. The effect caused depends on the type of material the obstacle is made of. The type of materials can be divided into two categories: insulators and conductors. Insulators are very transparent to radio waves, which means that if the obstacle is made up of this type of material, the effect is usually insignificant. On the other hand, the conductors, whose most common are the metals, will reflect most of the radio waves, which will give rise to more difficulties, and the signal will have less power and therefore a reduced range, and the signal will spend more time trying to cross the obstacle, reducing accuracy [28].

Operational Principle

The UWB works through an omnidirectional radio wave that is sent from the transmitter to the reference points (also known as anchors) and measures the time between the emission and reception of the wave using one of the time ranging methods, such as ToF. As radio waves travel at the speed of light (c = 299792458 m/s), the distance traveled can be easily deduced by dividing the ToF by the speed of light. With this distance it is then possible to obtain an estimated position of a given object [3].

Radio waves travel very fast, and so there is a need for high temporal precision that can only be achieved in the presence of a very narrow pulse [3]. From the Heisenberg's uncertainty principle (Equation 2.1) it is possible to determine approximately the pulse width (Δt), given a certain bandwidth (Δf). From the analysis of Equation 2.1 it is possible to conclude that if it is desired to have a very narrow pulse, a large bandwidth is required [3].

$$\Delta t \Delta f = \frac{1}{4\pi} \tag{2.1}$$

The UWB signal has a bandwidth of about 500 MHz which results in impulses of 0.16 ns duration. Thanks to this time resolution, the receiver is able to distinguish various reflections from the signal (Figure 2.7), thus making it possible to have a precise range even in places where there are many reflectors, such as indoor environments.

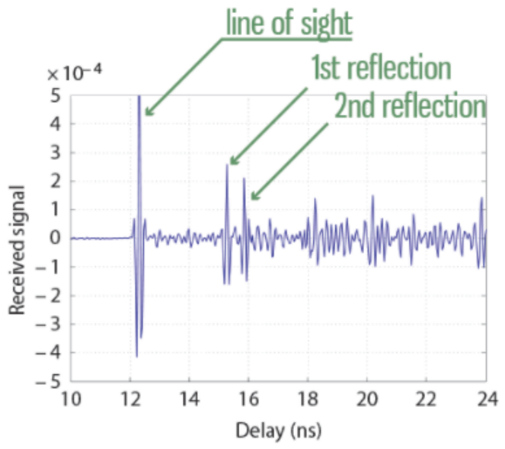


Figure 2.7: Several reflections of the signal [3]

The higher bandwidth comes at the expense of rigorous low-power regulations regarding UWB systems (power spectrum density must be below -41.3 dBm/MHz) (Figure 2.8). Consequently, a single pulse is not distinguishable from noise (pulse will probably be below noise level) when it reaches the receiver. To solve this problem a pulse train is sent by the transmitter instead of just one, which are accumulated. With sufficient pulses, the power of the accumulated pulse will increase above the noise level and reception is possible [3].

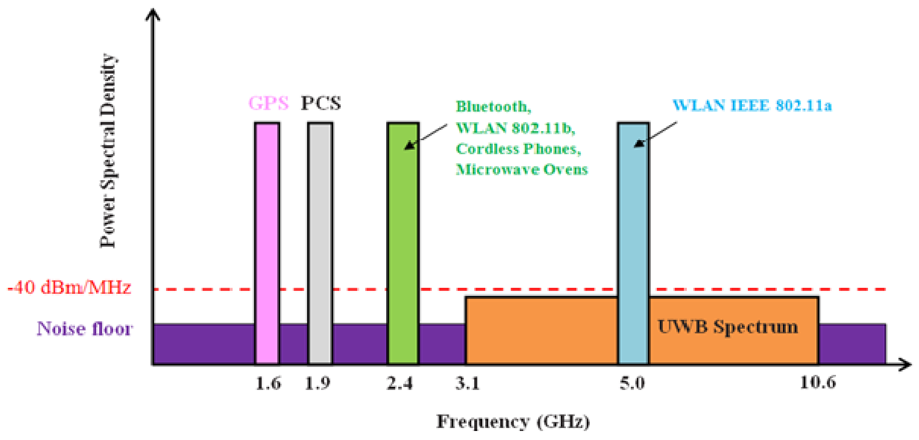


Figure 2.8: Regulated UWB spectrum [4]

UWB solutions in Market and Previous Work

In order to obtain an efficient UWB-based solution for indoor environments, several companies manufacture UWB chipsets and hardware. Ubisense that takes advantage of AoA and TDoA (Dimension 4), DecaWave that uses ToA and TDoA (DW 1000), Uniset with TDoA (Sequitir) and Zebra Technologies that uses TDoA (Dart UWB), are among the best known commercial implementations that use UWB and that allow to acquire accuracies from 10 to 30 cm [29, 30, 31, 32, 25].

UWB has been increasingly used in applications related to localization on indoor environments and therefore several studies have been conducted by various researchers. For example, the authors of [21] proposed a method that uses Particle Filter to fuse UWB range measurements with odometry to obtain a real-time posture estimation of a wheeled robot in an indoor environment. The firmware of the UWB devices and the trilateration algorithm were also improved in order to obtain greater measurement stability in the presence of obstacles. It was concluded that with the method used the estimator has sufficient performance for the robot to maintain a position for a period of time and to run along a simple path. The experiments also showed that the interferences caused by dynamic obstacles could be partially relieved.

The authors of the paper [17] use a Kalman filter to combine information from a mobile robot's odometry and Ultra-Wideband ToF distance modules, which lacks the orientation. The proposed system is used to localize the robot and make a 3 DoF scanning of magnetic field in a room. Is also implemented a pre filtering system composed by a median based gate allows to reduce the covariance. The implemented system allowed the localization error to be reduced.

The paper [23] proposed a method to accurately locate mobile robots through sensor fusion. The acceleration from an inertial measurement unit (IMU) and the 2-D coordinates received from Ultra-Wideband (UWB) anchors are fused through a Kalman Filter to obtain an accurate location estimate. The proposed method increases robustness, scalability and location accuracy. The measurement results, which were obtained using the proposed fusion, show considerable improvements in the accuracy of location estimation (the original error was reduced by 75 %). They also concluded that the fusion of different sensor and evaluation systems using Kalman filtering, therefore allows for real-time capability.

The paper [33] presents an accurate and easy-to-use method for UWB anchor self-localization, using the UWB ranging measurements and readings from a low-cost IMU. The locations of the anchors are automatically estimated by freely moving the tag in the environment. The method is inspired by the Simultaneous Localization and Mapping (SLAM) technique used by the robotics community. They use a tightly-coupled Error-State Kalman Filter to fuse UWB and inertial measurements, producing UWB anchor position estimates and Six Degrees of Freedom (6DoF) tag pose estimates. Simulated experiments show that the method used enables accurate and robust UWB anchor self-localization within a few seconds and is capable of drift-free localization of the moving tag.

The authors of the paper [34] propose a methodology to identify the LOS/NLOS conditions. They use an EKF to fuse the odometry information of a robot, IMU and UWB in order to estimate its pose in an unknown interior environment. An algorithm based on variance measurement technique of distance estimates along with power envelope of the received signal is proposed for NLOS identification. Further, adaptive adjustment of sensor noise covariance approach is devised to mitigate the NLOS effect. A novel approach based on dynamic adaption of noise covariance is incorporated for further improvement in localization accuracy. Experiments performed with the proposed fusion approach achieves $\sim 2X$ improvement in accuracy.

2.1.2.2 Image Based

Image based technologies include a camera and computer vision based technologies. For this technologies can be used different types of cameras, including omni-directional and three dimensional cameras. The performance of these cameras varies according to the amount of information that can be derived from their images [2, 15].

There are two main categories of image based positioning systems: ego-motion systems and static sensor systems [15]. The first one uses the camera's movement with respect to a rigid scene in order to estimate the current position of the camera. On the other hand, the second one locate moving objects in the images.

Since nowadays almost all newly developed mobile robots have a built-in camera system, it is possible to use this to determine the robot's pose in a given environment [6]. A visual system that include one camera should detect some markers in the environment by using computer vision tools.

2D planar fiducial marker systems

In the last few year, fiducial markers have become a popular and efficient tool to solve monocular localization and tracking problems at a very low cost in indoor environments [35]. Fiducial marker systems consists of patterns (artificial landmarks) that are placed in the environment and automatically detected in camera images using an accompanying detection algorithm. These systems are useful for Augmented Reality (AR), robot navigation and applications where the relative pose between camera and object is required, due to their high accuracy, robustness and speed [5].

2D fiducial markers can be circular, square or others depending on the system. Circular systems can provide only a single point (the center) with a high degree of accuracy, so multiple circular markers are needed for full pose estimation. On the other hand, square fiducial markers can provide a key point for each of the four corners, which are sufficient to perform the camera pose estimation. Several fiducial marker systems have been proposed in the literature such as ARSTudio, ARToolkit and ARTag, shown in Figure 2.9. These markers have a similar function, their performances differs in terms of computing time and detection rate. To clearly identify the markers, a visual localization system using 2D markers consists of a set of plane patterns and algorithms that recognize these patterns. In addiction, it is necessary to ensured that the pattern encoding the unique marker identifier cannot be created by rotation of another pattern used in the environment [6]. Currently, the most popular marker in academic literature is the ArUco which is based on ARTag and comes with a library OpenCV of functions for detecting and locating markers [36].

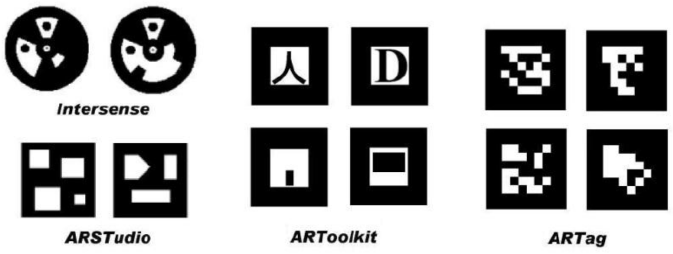


Figure 2.9: Examples of 2D marker systems [5]

The markers in ArUco system are defined as 7x7 square matrix. Each of these squares is either black or white, however the outer two rows and columns are black (Figure 2.10). The ArUco system comes with an OpenSource library developed by Rafael Muñoz and Sergio Garrido, wherein algorithms are implemented in functions for detecting, recognizing and locating the markers [37]. With use of digital coding theory, the system achieves a low level of confusion or detection of false markers. In general, the detection process of squared markers involves thresholding the scene, where a set of square regions, i.e candidate markers, are extracted from the background. Then, the interior of the regions is analyzed, discarding those that cannot be identified. Lastly, using the four corners of at least one marker, the camera position can be estimated. For efficient marker detection, the system assumes that the marker size and the camera calibration parameters are known. After a successful marker detection, ArUco library functions provide position information relative to optical center of the camera. For the rotation, the ArUco system provides standard transformation matrix or quaternion [6]. Moreover, several type of research have been successfully conducted based on this type of system: [22, 6, 38, 35, 39].

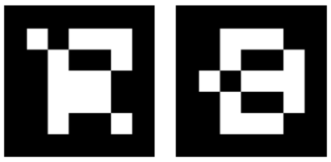


Figure 2.10: ArUco markers [6]

Camera Calibration

Some pinhole cameras (simple cameras, without a lens and with a single small aperture) introduce significant distortion into images. So, when using vision-based technologies it is important to calibrate the camera uses for a correct information extraction from the images. Camera calibration estimates the parameters of a lens and image sensor of an image or video camera. These parameters are used to correct the lens distortion, measure the size of an object in world units or determine the location of the camera in the environment.

There are two main kind of distortion: radial distortion and tangential distortion [7]. Radial distortion causes straight line appear curved and becomes greater the further the points are from the center of the image. The Figure 2.11 shows this kind of distortion well, where the two edges of the chessboard are marked with red lines. It can be seen that the edge of the chessboard is not a straight line and does not match the red lines, so the image shows distortion. Radial distortion can be represented by the Equations 2.2 and 2.3.

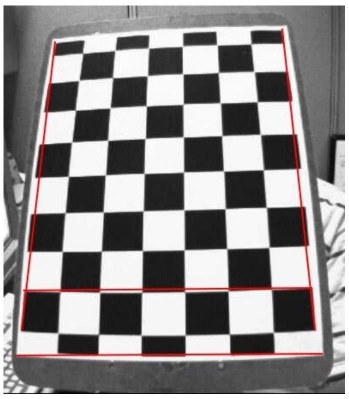


Figure 2.11: Radial Distortion in a chessboard [7]

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.2}$$

$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.3}$$

Tangential distortion occurs because the image pickup lens is not perfectly aligned with the image plane. Thus, some areas of the image may appear closer than expected. This kind of distortion can be represented by the Equations 2.4 and 2.5.

$$x_{distorted} = x + (2p_1 xy + p_2(r^2 + 2x^2)) \tag{2.4}$$

$$y_{distorted} = y + (p_1(r^2 + 2y^2) + 2p_2 xy) \tag{2.5}$$

In brief, it is necessary to find five parameters, known as distortion coefficients given by the matrix present in Equation 2.6.

$$Distortion\_Coefficients = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} \tag{2.6}$$

In addition, other parameters such as intrinsic and extrinsic parameters of the camera are needed to perform the camera calibration. The extrinsic parameters correspond to the rotation and translation vectors that translate the coordinates of a 3D point to a coordinate system. Intrinsic parameters are camera-specific parameters that include information such as focal length (f_x, f_y) and optical centers (c_x, c_y). The focal length and optical centers can be used to create a camera matrix, which can be used to remove distortion due to the lenses of a specific camera. This matrix is unique to a specific camera, so once calculated, it can be reused in other images taken by the same camera [7]. It is expressed by the matrix present in Equation 2.7.

$$Camera\_Matrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

To find all these nine parameters, it is necessary to provide some sample images of a well-defined pattern, for example a chessboard. By finding some specific points of which the relative positions are known, i.e. the square corners on the chessboard, and knowing the coordinates of these points in real world space (3D points) and the coordinates in the image (2D points), it is possible to find the distortion coefficients using correspondences [7].

2.1.2.3 Radio Frequency Identification (RFID)

Radio Frequency Identification (RFID) is a technology that uses radio waves to transmit the identify of a given object or a person wirelessly. This technology is very used to identify objects in large systems automatically [15, 2].

RFID technology consists of an exchange of different frequencies of radio signals between two components: RFID readers and RFID tags. The RFID reader reads the data emitted by the RFID tags, whereas the tags respond in conformity with the unique identification or information stored in them. Both components use a defined radio frequency and protocol in order to transmit and receive data [40].

RFID tags are fixed to objects that require tracking and consist of a micro ship and a radio antenna. The tags are categorized as either passive or active. The active RFID tag is a battery-powered transceiver, and therefore has a wider transmission range, thus reducing the number of tags needed for an installation. Passive RFID tag has a shorter transmission range since it does not use a battery and it gets its power from the reader's signal before it can respond with information. RFID reader consist of different components, including an antenna, transceiver, power supply, processor and interface, in order to connect to a server [1, 15]. Even though, different positioning methods can be used with RFID, proximity is the most used one and it senses the presence of RFID tags rather than the exact position [15].

2.1.2.4 Infrared (IR)

Infrared (IR) is a technology that is similar to the idea of infrared light, the only difference is the fact that it estimates its local position rather than estimating an object position. The use of the invisible spectrum of light just below the red edge of the visible spectrum by infrared wireless communication makes this technology less intrusive compared to indoor positioning which is based on visible light. There are several positioning methods that are frequently used with these technology, including Angle of Arrival, differential phase-shift and proximity [41, 2, 15].

Infrared technology can be used in two different ways: direct IR and diffuse IR. There is an example of direct IR that is Infrared Data Association (IrDA). IrDA uses a point-to-point Ad-hoc data transmission standard designed for very low-power communications and requires a Line-of-Sight communication between devices over a very short distance. Diffuse IR uses wide angle LEDs that emit signals in multiple directions, which allows one to many connections and doesn't need direct Line-of-Sight. Diffuse IR has a more powerful signal than direct IR and, for this reason, has a longer range [15, 2].

2.1.2.5 Ultrasonic

The ultrasound wave is a mechanical wave which is an oscillation of pressure that is transmitted through a medium, doesn't interfere with electromagnetic waves and has a relative short range. Since ultrasound is a mechanical wave, it depends on the state of the medium and its temperature [2, 41].

Ultrasonic technology estimates a given position by measurements of Time of Arrival (ToA), where the receiver can determine the distance traveled by the wave emitted by the emitter. An estimation of the emitter's coordinates is possible by multilateration from three or more ranges to some fixed receivers deployed at known locations [15].

2.1.2.6 Zigbee

Zigbee is based on the IEEE 802.15.4 standard that addresses the physical and MAC levels for low data rate, low cost and energy efficient personal area networks. This standard is a short distance and low rate wireless personal area network [2, 27].

Exists two distinct types of physical devices that are typically used for Zigbee nodes which are Full Function Device (FFD) and Reduced Function Device (RFD). Zigbee technology achieves positioning through coordination and communication with neighboring nodes and typically uses RSS values to estimate the distance between the nodes [15, 2].

2.1.2.7 Wireless Local Area Network (WLAN)

IEEE 802.11 WLAN, commonly known as Wi-Fi is another good wireless way of estimation, which relies on the strength of the signal received. A WLAN is a high-speed wireless network that by using high-frequency radio waves can connect and communicate between nodes and devices within a building or area. In this way, it is possible to assist the mobility of users within the coverage area [1].

With this technology it is possible to locate and determine the position of a certain object or person inside a coverage area using WLAN infrastructures [1]. Moreover, using Wi-Fi on indoor positioning and navigation systems depends on knowing a list of wireless routers that are available in an area in which the system operates [15].

The RSS, ToA and AoA techniques and any combination of them (i.e. hybrid methods) can be used to provide Wi-Fi based localization services. However, the most popular technique is RSS in which it is possible to achieve the accuracy of WLAN positioning systems around 3 to 30 m, with an update rate in the range of few seconds [15, 2].

2.1.2.8 Cellular Based

Global System for Mobile Communications (GSM) is accessible in the most countries that achieve WLAN coverage with the lowest positioning accuracy. GSM works in licensed bands that prevent interference from other devices that operate in the same frequency [15].

Indoor positioning based on mobile cellular network is only possible if the building is covered by several base stations or one base station with strong RSS received by indoor mobile clients. So, the most common method of GSM indoor positioning is fingerprinting which is based on the power level (RSS) [40].

2.1.2.9 Bluetooth

Bluetooth is a wireless standard for Wireless Personal Area Networks (WPANs). This technology involves the specifications of the physical and MAC layers to connect different fixed or moving wireless devices within a given personal space [27]. Bluetooth typically uses proximity and RSS methods for position estimation [15].

Bluetooth is a low-power technology in peer-to-peer communications. The advantage of using Bluetooth in positioning systems is its high security, small size and low cost. However, it cannot be used for real-time localization due to its comparative high delays with respect to the other techniques in this category [41]. Compared to WLAN, presented earlier, the raw bit rate is lower and the range is shorter [15].

2.1.2.10 Dead Reckoning

Using dead reckoning, an object can roughly determine its current position from its past position and the speed at which it is moving [15]. It is a cheap solution for determining the location of a given object and it is possible to obtain the location accurately in the short term and at high sampling rates. However, it should be noted that it is not an absolute form of localization.

Using dead reckoning it is possible to obtain accurate positioning information. However, this information is subject to error accumulation over a long period of time. With the aim of improving accuracy and reducing positioning error, other methods can be used to adjust the position after each interval [15].

2.1.2.11 Comparison between the technologies

As previously mentioned, depending on the application in question and its requirements, a given technology may be more or less appropriate. Therefore, the choice of the technology to be used for the location must be made carefully, taking into account the advantages, disadvantages and mode of operation of the technologies.

So, Table 2.2 presents the main advantages and disadvantages of each of the technologies previously presented. This table was suggested by the authors of the articles [15, 2].

2.1.3 Indoor Positioning Techniques

IPS can combine more than one of the positioning technologies discussed above and uses positioning techniques to locate objects and offer absolute, relative and proximity location information [42]. This techniques specify how to calculate the position of a target object, more specifically, translate a recorded signal properties into distances and angles and then computes the actual position of the object. The signal properties and the positioning techniques work together in order to determine the position of a given object [1]. According to the authors of the article [1] the main techniques used for positioning include trilateration, triangulation, proximity analysis and scene/fingerprint analysis. Where the triangulation and trilateration are the most common techniques.

2.1.3.1 Triangulation

To perform triangulation it is necessary to use the geometric properties of triangles to estimate the position of a given target object through the calculation of angular measurements with respect to two known reference points. In order to determine an object's position, the calculation of a transmitter's position based on angle and distance in relation to reference points is used [1]. So, the position is determined through the intersection of a pair of angle direction lines, including the distance between the lines. Triangulation uses the direction angle of arrival (AoA) as a foundation of measurement.

In general, two dimensional triangulation requires two angle measurements and one length measurement that is the distance between the reference points. In contrast, in three dimension requires one length measurement, one azimuth measurement and two angle measurements to specify a precise position [2].

When triangulation uses two or three reference points to determine a given position, the result is a simple and low-cost system. But when the coverage area is larger and with multiple reference points, the position determination can contain several errors which can result in a lower accuracy. In addition, the hardware requirement for a large coverage area tends to be complex and expensive [2].

2.1.3.2 Trilateration

Trilateration or lateration, like the triangulation, uses the geometric properties of triangles to estimate the position of a target object. But in trilateration, the determination of a position is accomplished using distance measurements with respect to three known reference points [1].

In two dimensions the calculation of the position requires distance measurements from three non-collinear points. In three dimensions requires distance measurements from four non-coplanar points [2]. In two dimensions case, each distance will be the radius of a circle and the intersection

Table 2.2: Comparison between the different indoor positioning technologies [15, 2]

Technology	**Advantages**	**Disadvantages**
RFID	Penetrate solid and non-metal objects. Does not require LOS between RF transmitters and receivers.	The antenna affects the RF signal, the positioning coverage is small, cannot be integrated easily with other systems, RF communication is not inherently secure and consumes more power than IR devices.
UWB	High accuracy positioning, even in the presence of severe multipath, effectively passes through walls, equipment and any other obstacles. UWB will not interfere with existing RF systems if properly designed.	Although UWB is less susceptible to interference relative to other technologies, it is still subject to interference caused by metallic materials.
Infrared	Since IR signals cannot penetrate through walls, it is suitable for sensitive communication because it will not be accessible outside a room or a building.	Does not penetrate walls, therefore it is typically used in small spaces such as a room. IR communication is blocked by obstacles that block light which includes almost everything solid. Requires LOS between sender and receiver when using direct IR.
Ultrasonic	Does not require LOS. Do not interfere with electromagnetic waves.	Does not penetrate solid walls. There may be loss of signal because of obstruction. There are false signals because of reflections and interference caused by high frequency sounds.
Zigbee	Its sensors require very little energy. It is a low cost technology.	ZigBee seems vulnerable to interference caused by a wide range of signal types (using the same frequency). This might disrupt radio communication. It is suitable for networks in which conversation between two devices takes some few milliseconds which allows the transceiver to switch to sleep mode quickly.
WLAN	Use existing communication networks that may cover more than one building. The majority of devices available nowadays are equipped with WLAN connectivity. WLANs exist approximately in the majority of buildings. LOS is not required.	A major drawback of WLAN fingerprinting systems is the recalculation of the predefined signal strength map in case of changes in the environment (e.g. open/closed doors and the moving of furniture in offices).
Cellular Based	No interference with devices that operate at the same frequency. The hardware of customary mobile phones can also be used.	Low reliability due to varying signal propagation conditions.
Bluetooth	Does not require LOS between communicating devices. It use a lighter standard and highly ubiquitous. It is also built into most smartphones, personal digital assistants, etc.	The greater the number of cells, the smaller the size of each cell and hence better accuracy, but more cells increase the cost. Requires some relatively expensive receiving cells. Requires a host computer to locate the Bluetooth radio. Because of the 2.4 GHz spectrum that Bluetooth is using is unlicensed, new uses for it are to be expected, and as the spectrum becomes more widely used radio interference is more likely to occur.
Dead Reckoning	Does not require additional hardware such as sensors.	The Dead Reckoning calculates only an approximate position.
Image Based Technologies	They are relatively cheap compared with other technologies such as ultrasound technologies.	Requires LOS and the coverage is limited.

of all circles will be a unique point, theoretically. But, in practice, measurements are prone to noise and uncertainties, and therefore the intersection may be a region of points rather than a unique point. In three dimensions the principle is the same, but instead circles the distances forms spheres [2].

To automatically measure the distance between the target object and the reference positions in IPSs is common use Time of Flight (ToF) approach, that measure the time it takes to travel between the object and the reference points at a known velocity [2]. Multilateration is another technique that is similar to trilateration and uses four or more reference points [1].

2.1.3.3 Proximity

In contrast to triangulation and trilateration presented earlier, proximity cannot provide an estimate of absolute or relative position, since it only gives position information. To provide the information a dense grid of antennas, that have a well-known positions is used to derive the position of a target object. When a mobile target is detected by a single antenna (closest antenna) it is considered to be collocated with it. However if more that one antenna detects the mobile target, it is considered to be collocated with the one that receives the strongest signal. This method is relatively simple to implement [1, 40].

Usually, in proximity technique, the position of a given object is specified using the RSS technique. The accuracy of the proximity technique is associated with the density of the detectors and the signal range of each one. When high accuracy and wider coverage area is required, is important to be careful with the concentration of detectors that can bring complexity and high cost [1].

This technique can be implemented over different types of physical media. In particular, the systems using IR, RFID and Bluetooth are often based on this [40].

2.1.3.4 Scene Analysis and Fingerprinting

To estimate the position through Scene Analysis it is not necessary to use angle or distance. Scene Analysis, also known as fingerprinting, refers to the type of techniques that begins by collecting information or characteristics (fingerprints) from a scene and later estimates the position of a given object through matching or comparing the information collected with that found within an available database. A fingerprint is a unique feature or signature that allows a distinction between scenes. In the scene analysis technique, the collected information that is later compared is usually the received signal strength (RSS) that depends on the location [1, 2, 40].

In order to collect the location fingerprints there are two phases: offline stage and online stage. In the offline stage, which is also called the training phase, an environment within a building is monitored and grid points are computed at different places in the building. At each grid point, a list of Received Signal Strength Indicator (RSSI) values for visible access points at the locations is listed. In addition the positioning and signal strength information for the different locations are collected for position estimation needs. During the online stage, which is also called the serving stage or run time stage, a positioning technique makes use of the currently observed signal strengths and previously collected information to discover an estimation of the location [1, 2, 40].

There are a few challenges related to techniques based on location fingerprinting: the RSS could be affected by diffraction, reflection and scattering in the propagation indoor environments. In addition, is a time-consuming process and have high computational cost [1, 2].

2.2 Fusion Algorithms

To determine the position of a robot it is possible to use any of the solutions presented above. However, there may be uncertainties in the results, due to, for example, environment dynamics, inaccurate models and limitations of the sensors used, such as resolution or noise in the measurements.

With the use of probabilistic algorithms it is possible increase the accuracy and robustness of the location systems, since it can handle with ambiguous measurements and noise from the sensors that compose the system, from the detection of unexpected disturbances from the environment. In addition, these algorithms can be used to fuse different technologies, which can increase the redundancy of the state estimation.

There are two techniques to perform estimation: recursive state estimation techniques and batch methods. Batch methods are less common in robotics since they cannot be used in real time, once they cannot employ future measurements to estimate past states [43]. On the other hand, recursive state estimation techniques are more used in robotics and are therefore the ones that will be addressed in this section. This last technique has a representation based on Bayes' theorem present on the Equation 2.8. The probability $p(x|y)$ is known as the posterior probability distribution over x [44]. This represents the confidence that the system is in state x given the measure y. It should be noted that in robotic localization systems state x is usually the robot's pose. It should also be noted that past and future data does not depend on knowledge of the current state, and because of this knowledge of the immediately previous state and current measurements is sufficient to predict the current state [44].

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \tag{2.8}$$

The probabilistic localization is a method which calculates/approximates the probability distribution of the robot localization at each instant. This method usually have two phases:

- Action/Prediction phase: uses proprioceptive sensors, such as encoders and gyroscopes. In other words robot sensors that measures position/movement without depending on anything external to the robot;

- Perception/Actualization phase: Uses exteroceptive sensors such as sonar, compasses and cameras. In other words robot sensors that measure the position/movement but rely on something external to the robot.

2.2.1 Gaussian Filters

In this section will be discussed the most used filters family to date in localization algorithms, the Gaussian filters.

Gaussian filters share the basic idea that beliefs are represented by well known unimodal distribution, the multivariate gaussian distribution and can be characterized by just a mean and a covariance [44]. The best known filters of this type are the Kalman Filters and their extended versions and for this reason it will be these that will be addressed below.

2.2.1.1 Kalman Filter

Kalman Filter (KF) is a widely investigated technique for implementing Bayes filters as mentioned in [44], because of its simplicity, optimality and robustness. This filter is a linear, discrete time, finite dimensional time-varying system that evaluates the state estimate that minimizes the mean-square error.

The dynamics of this filter results from the consecutive cycles of Prediction and Filtering/ Correction [45]. Prediction and Correction are the two different phases for follow the state estimation and its uncertainty. On the first phase, prediction phase, the new state is predicted by introducing the previous estimate and its uncertainty, along with the motion data into a state transition model. On the second phase, correction phase, the sensors data are integrated into the calculate state in the previous phase, this obtaining a final estimation and its uncertainty.

It is important to note that most robotics problems deal with nonlinear systems and therefore the use of the Kalman Filter is not a good option. In order to overcome this problem the Extended Kalman Filter (EKF) was created [45], which is an extension of the Kalman Filter that will be addressed below.

2.2.1.2 Extended Kalman Filter

The Extended Kalman Filter, as mentioned above, is an extension of Kalman Filter that was created with the intention of dealing with nonlinear systems. This type of filter assumes that the next state probability and the measurement probabilities are governed by nonlinear arbitrary functions [45].

The EKF estimates an approximation to the true belief and represents this approximation by a Gaussian. Thus, this filter is also represented as a Gaussian probability distribution over the variables, like the Kalman Filter, but this probability is just an approximation of the true belief, not exact as was the case with the Kalman Filter.

The key idea underlying the Extended Kalman Filter is called linearization. There are a multitude of linearization techniques for nonlinear functions, but EKF uses the (first-order) Taylor expansion method, which builds a linear approximation to a function f through the value of f and the slope. The first order Taylor Expansion is demonstrated on the Equation 2.9.

$$f(x) \approx f(a) + f'(a)(x-a) \tag{2.9}$$

The EKF is based on two important assumptions: the state estimation is relatively close to the actual value and nonlinear functions can be approximated accurately. If these functions are approximately linear, the EKF approximation may generally be a good and accurate one. However, in the same cases, the functions are not only nonlinear. These functions may also be multimodal or highly nonlinear, in which case linearization can be a poor approximation.

It also important to note that the use of EKF has various advantages. The major advantages are its simplicity and computational efficiency, because of the unimodal Gaussian distribution. In addition, the computational complexity just depends on the dimension of the measurement and state vectors. So the EKF is more efficient than Non Parametric Filters, whose algorithms uses more sophisticated representations and will be addressed on Section 2.2.2. Due to these advantages, this filter is arguably the more popular tool for state estimation in robotics and can be proven from several papers like this ones: [20, 18, 46].

2.2.1.3 Unscented Kalman Filter

As previously addressed, there are multiple linearization techniques for nonlinear functions, which Taylor expansion is one of this. However, there are other techniques that produce superior results. One of these is the Unscented Kalman Filter (UKF). This filter appears in order to addresed the error caused by the linearization of the Extended Kalman Filter.

UKF yields performance equivalent to the Kalman Filter for linear systems, yet generalizes elegantly to nonlinear systems without linearization steps required by Extended Kalman Filter [47]. This filter uses a deterministic sampling technique known as the unscented transformation to pick a minimal set of sample points, called sigma points, around the mean. The state is still approximated as a gaussian random variable, although is just linearized in a smaller set of sampled points (sigma points picked by the unscented transformation).

Although the computational complexity of this technique is similar to the EKF technique, the process of picking the sigma points involves a little more effort. Comparisons between EKF and UKF have been proposed in several contexts ranging spacecraft localization ([48]), mobile robot localization ([49, 50]) and others. In particular, in the last article mentioned it was compared the performance of an UWB-based localization system for a quadcopter on indoor and outdoor scenarios using the EKF and UKF. The authors concluded that for UWB-based localization systems the EKF is preferred because reveals a more consistent results. In contrast UKF reveals a sensitivity of the weights of sigma points to different anchor configurations. It is possible conclude that for measurement models that have large nonlinearities, UKF produces better performance, however, if measurement models don't have large nonlinearities and are reasonably stable, EKF is better, because is simpler and reaches a great levels of accuracy.

2.2.2 Non-parametric Filters

A common alternative to gaussian techniques are non-parametric filters. This type of filters, in contrast to gaussian filters, removes the fixed distribution form (gaussian distribution) of the beliefs representation. Non-parametric filters approximate posteriors by a finite number of values, each roughly corresponding to a region in state space [44].

Unlike gaussian filters, non-parametric filters are well-suited to represent complex multimodal beliefs. They can handle well with phases of global uncertainty and with data association problems. This type of filter makes no assumptions regarding the initial state, so in robot localization problems, the robot can start from any unknown position. The representative power of these techniques has a price: increase of the computational complexity that relies on the number of parameters used to approximate the posterior. It is important to note that the quality of the approximation varies according to the number of parameters used: the higher number of parameters the better will be the quality of the approximation. Therefore, a trade-off analysis must be made keeping in mind the complexity of the posterior.

There are two principal non-parametric approaches: histogram filter and particle filter. The first one will be addressed on Section 2.2.2.1 and the second one on Section 2.2.2.2.

2.2.2.1 Histogram Filter

Histogram filter splits the state space into finitely many regions and represents the posterior by a histogram. The histogram assigns a single cumulative probability to each of the regions generated by the filter. By choosing the sampled region with the highest probability it is possible to complete the estimation.

The histogram filter is used in a grid localization approach [44]. In this approach the environment where the robot is located is discretized in a grid map and each cell of the grid map represents the probability that the robot is in it. A critical variable of grid localization is the grid resolution. If the choice of grid resolution is well done, this technique can be a good solution for localization problems, so this decision should be made with caution. For example if a too high resolution is chosen for the grids discretization, the algorithm may become extremely slow and memory consumption extremely high. This approach, grid localization, has also various advantages, in particular its robustness and ability to converge to the correct value even in dense or unstructured environments.

2.2.2.2 Particle Filter

Another type of non-parametric filters is the particle filter, which concept is similar to histogram filter (the posterior is approximated by a finite number of parameters). This two type of filters differs in the way that these parameters are generated and in which they populate the state space. The key idea of the particle filter is to represent the posterior by a set of random state samples, called particles, drawn from this posterior. Instead of representing the distribution by a parametric form, particle filter represents a distribution through a set of samples taken from this distribution [44].

For global localization problems is very common the use of algorithm called Monte Carlo Localization (MCL), that use the particle filter, as can be verified by [51, 52]. In this algorithm each particle represents the probability of the robot being in a specific pose. It is a recursive algorithm based on Bayes rule, which propagates (Prediction phase - sampling) and updates (Update phase - resampling) the particles. If everything goes well, at the end, the region with more particles (denser region) is the region around the truth robot's pose. The number of particles influences the precision of the result, so it is important to be careful with the computational weight.

Monte Carlo Localization has problems associated with resampling. The particle filter may not be able to recover from a kidnapping of the robot or from a bad estimate of the localization because after a while, all the the particles will be in one area near real position of the robot. To avoid this some particles should be introduced randomly along the navigation space of the robot. This turns the algorithm more robust.

Article [51] presents also a series of advantages of using particle filter in comparison with other filters. By analyzing these advantages it is possible to conclude that particle filter reduces drastically the memory consumption compared to histogram filters. It is also more accurate than histogram filters with a fixed cell size. The advantages of using the particle filter have made this method very popular in mobile robotics applications, as can be seen from the articles [53, 54, 55].

2.3 Robotic Simulators

The validation of projects in the area of robotics is a complicated task. To support and facilitate this task there are several tools, such as 3D simulators.

Simulation softwares are a simple and more economical alternative for validation of complex systems, platforms or prototypes [8]. These are very useful to test the implemented code/algorithms before moving to the real system. Because of this, they make possible investigations that would not be possible without the use of this type of tools, due to limited resources or financial restrictions [8].

The use of simulators brings several advantages [8]:

- Allows the production and validation of the software, even without access to hardware, allowing a fast reconfigurability;
- Has a lower development cost, since initially only the intellectual cost exists, before investing in materials/hardware;
- It is a low risk environment, since it is a controlled experiment, which avoids accidents and damages;
- It can be flexible and dynamic, adapting specific sensors to improve results.

It should be noted that the more realistic the simulated model is greater the certain that it will work in the real robot, after work properly in the simulator. However, realism must be adapted to the application and the requirements of each application in order not to make the simulator unnecessarily complex and computationally heavy.

When choosing a simulator, the following features must be taken into account:

- Detail and realism of physical models (physics engine);
- Programming languages;
- Connection to other applications or external equipment;
- License type (open source, free, commercial or other);
- Included sensors and possibility of creating other sensors;
- Simplicity of installation and use;
- Computational resources;
- Operating Systems.

There are several 3D robotic simulators, among them Gazebo and V-Rep, which are the most used by the robotic community, according to [8]. And that's why they are the ones that will be analyzed in this section. Besides these two simulators mentioned above, Webots and SimTwo simulators will also be analyzed.

2.3.1 Gazebo

Gazebo is an open-source 3D robotic simulator for UNIX-like operating system, which allows its distribution without any costs. Gazebo software supports a wide variety of wheeled mobile robots or even humanoids [8]. The Figure 2.12 shows the Gazebo simulator software.

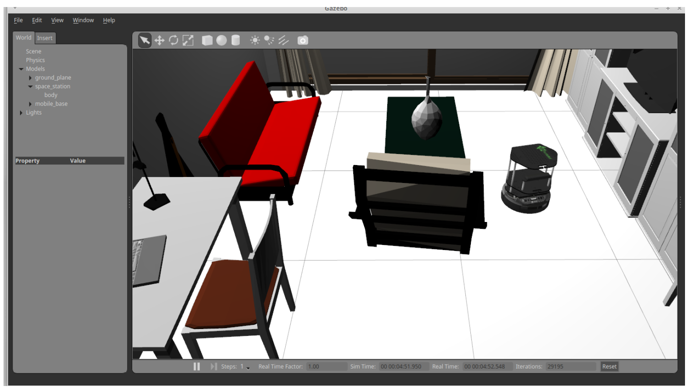


Figure 2.12: Gazebo simulator [8]

Gazebo is widely used by the Robot Operating System (ROS) community because of its ready integration with ROS. It also allows communication using sockets, facilitating integration with scripts in several languages that support this protocol [8]. Gazebo also have a good documentation on the Internet and a large library of scene elements. Gazebo website has tutorials and examples that teach how to use it and how to integrate it with ROS.

Regarding programming languages, Gazebo allows C/C++, Python and Ruby. Gazebo offers access multiple high-performance physics engines including Open Dynamics Engine (ODE), Bullet, Simbody and DART [56].

2.3.2 V-Rep

V-Rep is a non open-source 3D robotic simulator for UNIX-like operating system, Windows or Mac-OS. However presents a free version for educational applications. V-Rep software supports a variety of sensors, robotic arms, mobile robots and also accepts user-created import models on a graphical development platform [8]. The Figure 2.13 shows the V-Rep simulator software.

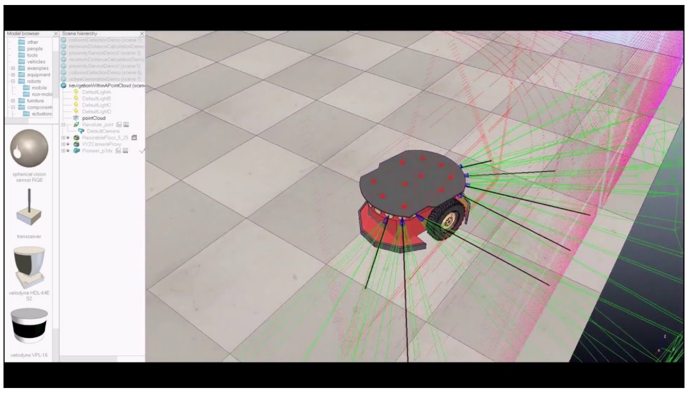


Figure 2.13: V-Rep simulator [9]

In [9] it is possible conclude that V-Rep was discontinued on November 2019 and substituted by CoppeliaSim that is totally compatible with V-Rep. However CoppeliaSim is fastest and have more features.

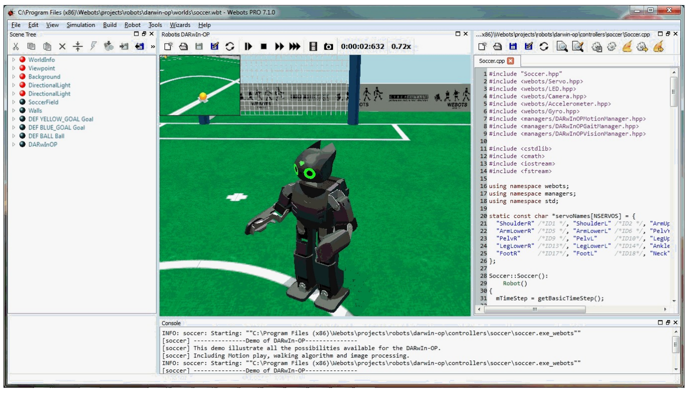
V-Rep/CoppeliaSim allows integration with ROS, such as Gazebo. Simulator and simulations are fully customizable, with six programming approaches that are mutually compatible and that can even work hand-in-hand (remote Application Programming Interface (API), ROS nodes, BlueZero nodes, plugins, embedded scripts and add-ons) [8]. Controllers can be written in C/C++, Python, Java, Lua, Matlab or Octave [9]. V-Rep offers access multiple high-performance physics engines including ODE, Bullet, Vortex and Newton [8].

2.3.3 Webots

Webots is an open source robotic simulator developed by Cyberobotics Ltd since 1998. It is a simulator that is very used in industry, education and research areas and it provides a complete development environment to model, program and simulate robots [10].

Webots has an asset library that includes various robots such as two-wheeled table robots, industrial arms and autonomous underwater vehicles, sensors, actuators, objects and materials. In addition, it is also possible to build new models from scratch or import them from 3D CAD software.

This simulator runs on Windows, Linux and Mac-OS and uses a ODE physic engine. The robots in this simulator may be programmed in multiple languages: C, C++, Java, Matlab or ROS with a simple API converting all the basic robotics needs. The Figure 2.13 shows the Webots simulator software.



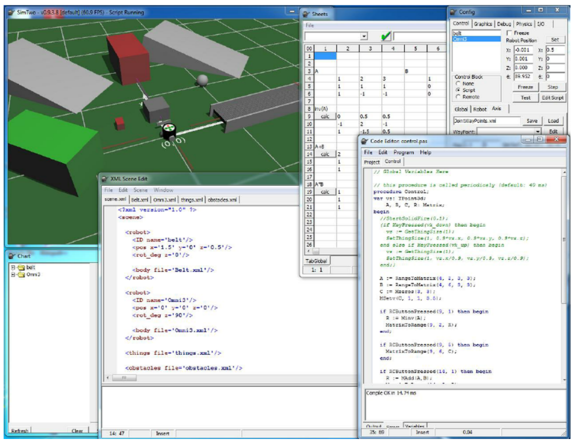
```
#include "Soccer.hpp"
#include <webots/Servo.hpp>
#include <webots/LED.hpp>
#include <webots/Camera.hpp>
#include <webots/Accelerometer.hpp>
#include <webots/Gyro.hpp>
#include <managers/DARwInOPMotionManager.hpp>
#include <managers/DARwInOPGaitManager.hpp>
#include <managers/DARwInOPVisionManager.hpp>

#include <cstdlib>
#include <cmath>
#include <iostream>
#include <fstream>

using namespace webots;
using namespace managers;
using namespace std;

static const char *servoNames[NSERVOS] = {
  "ShoulderR" /*ID1 */, "ShoulderL" /*ID2 */, "ArmUp
  "ArmLowerR" /*ID5 */, "ArmLowerL" /*ID6 */, "PelvY
  "PelvR"     /*ID9 */, "PelvL"     /*ID10*/, "LegUp
  "LegLowerR" /*ID13*/, "LegLowerL" /*ID14*/, "Ankle
  "FootR"     /*ID17*/, "FootL"     /*ID18*/, "Neck"
};

Soccer::Soccer():
    Robot()
{
  mTimeStep = getBasicTimeStep();
```

```
INFO: soccer: Starting: ""C:\Program Files (x86)\Webots\projects\robots\darwin-op\controllers\soccer\soccer.exe_webots""
[soccer] ---------------Demo of DARwIn-OP---------------
[soccer] This demo illustrate all the possibilities available for the DARwIn-OP.
[soccer] Including Motion play, walking algorithm and image processing.
INFO: soccer: Starting: ""C:\Program Files (x86)\Webots\projects\robots\darwin-op\controllers\soccer\soccer.exe_webots""
[soccer] ---------------Demo of DARwIn-OP---------------
```

Figure 2.14: Webots simulator [10]

2.3.4 SimTwo

SimTwo is a realistic 3D simulator with ODE physic engine. Is very simple to use and install and highly realistic concerning electric motors. SimTwo is very flexible and provides a great variety of sensors and actuators. This simulator is a free software that is not open source [11]. It is available for Windows operating system and its software can be visualized in the Figure 2.15.



```
<?xml version="1.0" ?>
<scene>

  <robot>
    <ID name='belt'/>
    <pos x='1.5' y='0' z='0.5'/>
    <rot_deg z='0'/>

    <body file='Belt.xml'/>
  </robot>

  <robot>
    <ID name='Omni3'/>
    <pos x='0' y='0' z='0'/>
    <rot_deg z='90'/>

    <body file='Omni3.xml'/>
  </robot>

  <things file='things.xml'/>

  <obstacles file='obstacles.xml'/>
```

```
// Global Variables Here

// this procedure is called periodicaly (default: 40 ms)
procedure Control;
var vs: TPoint3d;
    A, B, C, R: Matrix;
begin
  //StartSolidFire(0.1);
  {if KeyPressed(vk_down) then begin
    vs := GetThingSize(1);
    SetThingSize(1, 0.9*vs.x, 0.9*vs.y, 0.9*vs.z);
  end else if KeyPressed(vk_up) then begin
    vs := GetThingSize(1);
    SetThingSize(1, vs.x/0.9, vs.y/0.9, vs.z/0.9);
  end;}

  A := RangeToMatrix(4, 2, 3, 3);
  B := RangeToMatrix(4, 6, 3, 3);
  C := Mzeros(3, 3);
  MSetv(C, 1, 1, 3.5);

  if RCButtonPressed(9, 1) then begin
    R := Minv(A);
    MatrixToRange(9, 2, R);
  end;

  if RCButtonPressed(9, 5) then begin
    MatrixToRange(9, 6, C);
  end;

  if RCButtonPressed(14, 1) then begin
    R := MAdd(A,B);
```

```
Compile OK in 14.74 ms
```

Figure 2.15: SimTwo simulator [11]

Many types of robots can be implemented, such as wheeled mobile robots with different configurations, manipulators, legged mobile robots and underwater vehicles with thrusters [57]. The dynamics realism in SimTwo is obtained by decomposing a robot in rigid bodies hinge or prismatic joints, optionally actuated by electric motors. SimTwo allows the analysis of the interactions between the different bodies that are portrayed in the simulator in a very realistic way and in real time. This bodies have a set of parameters, such as shape, mass and moments of inertia, which allows to adjust mechanical properties and simulate their dynamics [11]. To enable the actuation of the robots, DC motors and controllers are also available, whose parameters of their models can also be adjusted, and their control can be done in the simulator itself or through a remote application that communicates via UDP or serial port.

It should also be noted that the SimTwo have a code editor that offers an Integrated Development Environment (IDE) for high-level programming based in Pascal language, that is the main tool in this simulator [11].

Table 2.3 provides a comparison of SimTwo simulator with the other simulators presented.

Table 2.3: Comparison of the different robotics simulators

Simulator	**Physics Engines**	**Programming languages**	**Connection to other applications**	**License**	**Operating Systems**
Gazebo	ODE, Bullet, Simbody, DART	C/C++, Python, Ruby	ROS, TCP, UDP, etc	Free	Linux
V-Rep	ODE, Bullet, Vortex, Newton	C/C++, Python, Java, Lua, Matlab, Octave	ROS, TCP, UDP, etc	Free only for non commercial applications	Linux, Windows, Mac-OS
Webots	ODE	C, C++, Java, Matlab	TCP, ROS	Free	Linux, Windows, Mac-OS
SimTwo	ODE	Free Pascal	UDP, RS232	Free	Windows

Chapter 3

System Architecture

This chapter describes the software and hardware components used in the proposed approach to get a precise robot localization in real-time. Figure 3.1 presents the system architecture proposed for this work and its data flow. This system has an easy switching between the real environment and the simulation environment, since it share the same communication protocol. The real robot used in this project was previously made by the supervisors for other projects that encompass mobile robotics.

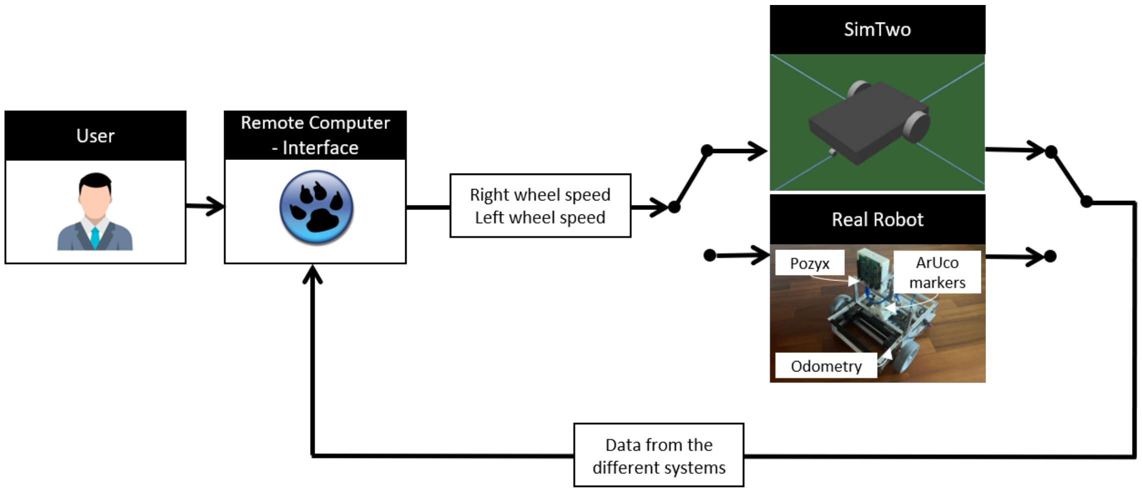


Figure 3.1: Architecture proposed for the interaction between the user and the robot

3.1 Remote Computer

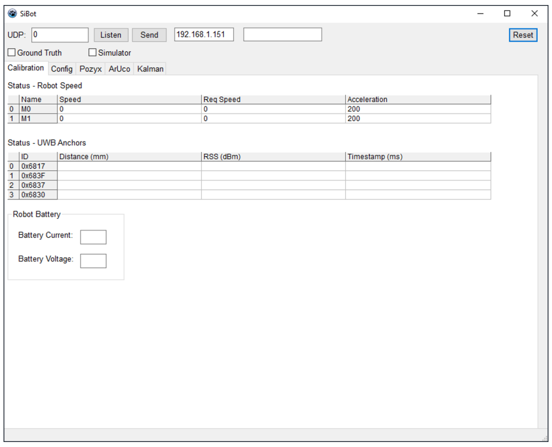
Beginning with the remote computer, this is where a Graphical User Interface (GUI) has been developed for remote access to the robot (real or simulated) and is where the center of all operations is located. The GUI deals with real and simulation environment. The idea was to create a control system that allows a user to perform commands on the computer through the use of buttons and other graphical elements, thus being able to interact with the robot. It is in the remote computer that the robot localization algorithm was implemented, based on the messages received by the robot.

The Graphical User Interface was developed in Lazarus that is a professional open-source cross platform IDE using the Free Pascal compiler. It has variety of components ready for use and a graphical form designer to easily create complex graphical user interfaces [58].

The protocol for the communication is User Datagram Protocol (UDP). Packets with information from odometry, UWB technology and ArUco markers is sent from the robot to the remote computer, whereas a packet containing the right and left speed wheels is sent from the remote computer to the robot. The Section 3.6 will describe this communication better.

The developed GUI presents a header and five main tabs. Through the header it is possible to establish connection with the real or simulated robot, through the "Send" button, which sends a packet to the desired IP, and the "Simulator" check-box, respectively. It is also possible to establish a connection to the Ground-Truth system through the corresponding check-box. This header can be seen in the upper part of Figure 3.2a.

In the first tab, represented in Figure 3.2a, it is possible to visualize different information received through the robot, such as speed and acceleration of each motor, corresponding to each wheel and data from the Pozyx system, i.e. distance, RSS and timestamp relative to each one of the anchors. In this tab it is also possible to check the robot's battery. The second tab, presented in Figure 3.2b, has the objective of setting different parameters of the robot, i.e. distance between wheels, wheel radius and steps to turn. It is also possible to indicate the position of each Pozyx anchor. Note that all these parameters are taken into account in the robot's pose estimation algorithm.



UDP: 0
192.168.1.151

☐ Ground Truth ☐ Simulator

Status - Robot Speed

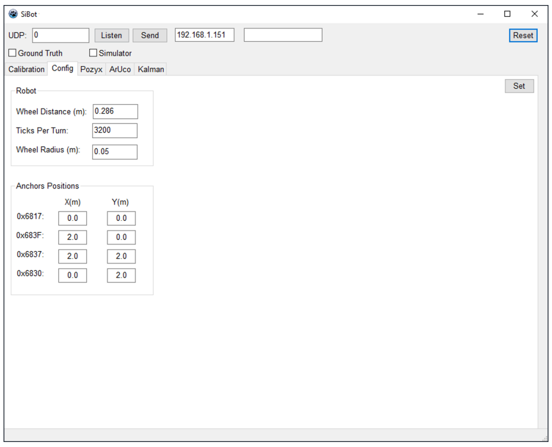
	Name	Speed	Req Speed	Acceleration
0	M0	0	0	200
1	M1	0	0	200

Status - UWB Anchors

	ID	Distance (mm)	RSS (dBm)	Timestamp (ms)
0	0x6817			
1	0x683F			
2	0x6837			
3	0x6830			

Robot Battery
Battery Current:
Battery Voltage:

(a) Received data from the robot



UDP: 0
192.168.1.151

☐ Ground Truth ☐ Simulator

Robot
Wheel Distance (m): 0.286
Ticks Per Turn: 3200
Wheel Radius (m): 0.05

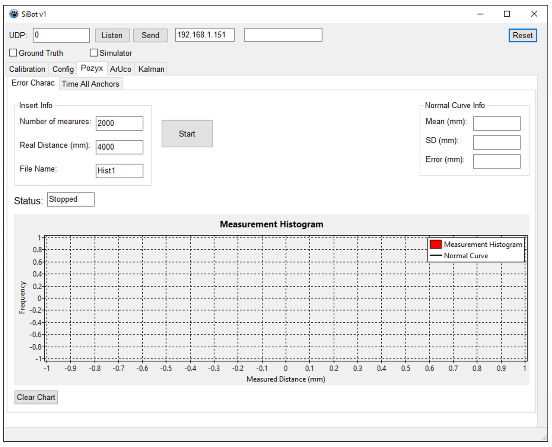
Anchors Positions

	X(m)	Y(m)
0x6817:	0.0	0.0
0x683F:	2.0	0.0
0x6837:	2.0	2.0
0x6830:	0.0	2.0

(b) General configurations tab

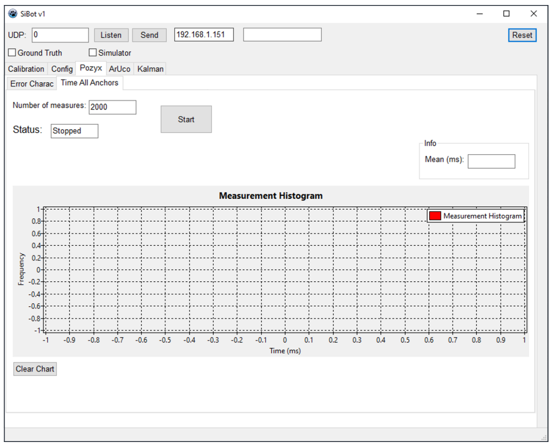
Figure 3.2: Tabs of robot configurations and data

The third tab is used to characterize the Pozyx system and consists of two sub-tabs. The first one (Figure 3.3a) is used to characterize the error of the Pozyx measurements, by placing the tag, incorporated in the robot, and an anchor separated by a given distance, and indicating a given number of measurements and the real distance measurement. After acquiring the number of measurements indicated in the respective text box, a histogram is generated with these measurements and the normal distribution that fits the histogram. The second sub-tab, represented in Figure 3.3b, is used to characterize the frequency of the measurements received from the Pozyx system, by indicating a given number of measurements. After this acquisition results a histogram and the average of the frequency with which the Pozyx measurements reach the remote computer.



UDP: 0 192.168.1.151
☐ Ground Truth ☐ Simulator
Number of mearures: 2000
Real Distance (mm): 4000
File Name: Hist1
Status: Stopped

(a) Pozyx distance characterization histogram

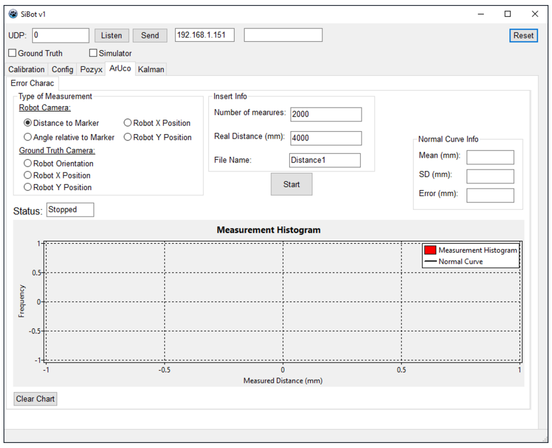


UDP: 0 192.168.1.151
☐ Ground Truth ☐ Simulator
Number of measures: 2000
Status: Stopped

(b) Pozyx time characterization histogram

Figure 3.3: Tabs of Pozyx characterization

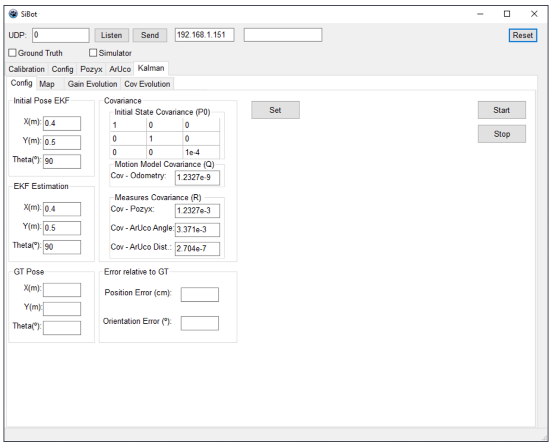
The fourth tab, presented in Figure 3.4, is used to characterize the different measures acquired through the ArUco markers. For this it is necessary to indicate the type of measure to be characterized, through the available check-boxes, the number of measures used for this characterization and the real measure. As a result a histogram is generated with all the measurements and the normal distribution that fits the histogram.



UDP: 0 192.168.1.151
☐ Ground Truth ☐ Simulator
◉ Distance to Marker ○ Robot X Position
○ Angle relative to Marker ○ Robot Y Position
○ Robot Orientation
○ Robot X Position
○ Robot Y Position
Number of mearures: 2000
Real Distance (mm): 4000
File Name: Distance1
Status: Stopped

Figure 3.4: Tab of ArUco measurements characterization

The last tab is dedicated to the Kalman Filter, used to estimate the pose of the robot and is composed of four different tabs. The first one, represented in Figure 3.5a, is used to define the initial values of the filter. In this tab it is also possible to see the current values of the pose estimated by the EKF, the pose obtained by the Ground-Truth system and the error present in the estimation. In the second tab dedicated to the Kalman Filter, and represented in Figure 3.5b, it is possible to observe in a map, the resulting positions of the EKF and the Ground-Truth system. The map also shows the position of all Pozyx anchors and ArUco markers, as well as the predefined path to be taken by the robot. Finally, the last two tabs, represented in Figures 3.5c and 3.5d, are useful to observe the evolution of the filter over time, namely the EKF gain matrix and the state estimation covariance matrix, respectively.



UDP: 0 192.168.1.151
☐ Ground Truth ☐ Simulator

Initial Pose EKF
X(m): 0.4
Y(m): 0.5
Theta(º): 90

EKF Estimation
X(m): 0.4
Y(m): 0.5
Theta(º): 90

GT Pose
X(m):
Y(m):
Theta(º):

Covariance
Initial State Covariance (P0)

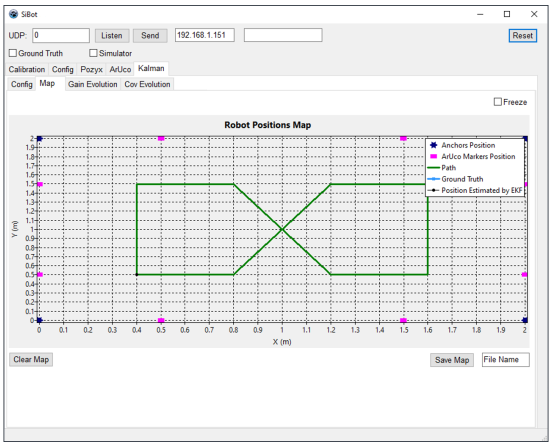
1	0	0
0	1	0
0	0	1e-4

Motion Model Covariance (Q)
Cov - Odometry: 1.2327e-9

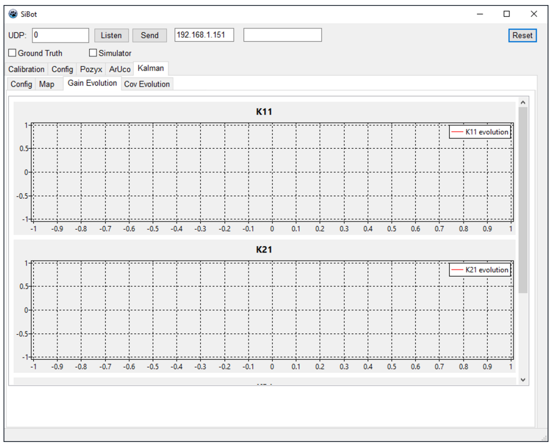
Measures Covariance (R)
Cov - Pozyx: 1.2327e-3
Cov - ArUco Angle: 3.371e-3
Cov - ArUco Dist.: 2.704e-7

Error relative to GT
Position Error (cm):
Orientation Error (º):

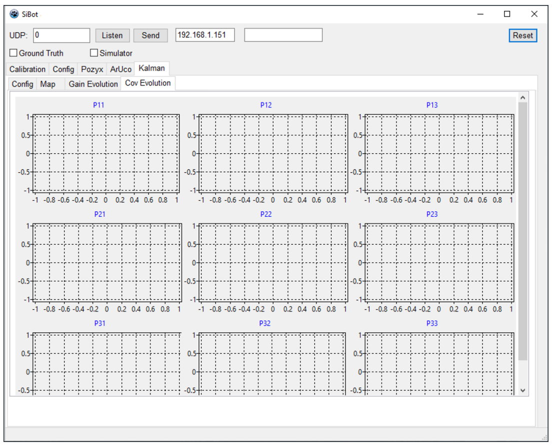
(a) Filter configurations



(b) Positions map



(c) Kalman Filter gains matrix evolution



(d) Covariance state estimation matrix evolution

Figure 3.5: Tabs relative to Kalman Filter

3.2 Real Robot Description

A mobile robot prototype presented in Figure 3.6 was provided by the supervisors of this dissertation. This section will describe the features of the real robot. First, the structural part that makes up the body of the robot will be presented. Then, it will be presented the robot's electronic schema. Lastly, it will be described the software that operates on the robot.

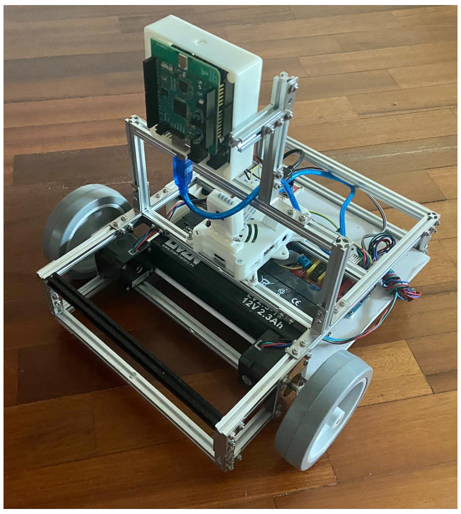


Figure 3.6: Real mobile robot

3.2.1 Structural Constitution of the Robot

The used robot has a differential wheel drive system. So, the robot has three degrees of freedom: $[x, y, \theta]^T$. The prototype has three wheels, two of which are coupled to independent stepper motors. The other is a castor wheel, which has the support function. Differential robots are widely used because of the simplicity of their control. The dimensions of the mobile robot's structure are shown in Table 3.1. This dimensions will be used for model the simulated robot.

Table 3.1: Dimensions of the robot

Description	**Dimension**
Length	0.31 m
Width	0.22 m
Wheel diameter	0.10 m
Wheel thickness	0.027 m
Wheel clearance	0.02 m
Distance from wheel to robot front	0.03 m
Castor wheel diameter	0.02 m
Castor wheel thickness	0.017 m
Robot mass	3.4 kg

3.2.2 Electronic Hardware

The mobile robot, in which the practical tests were carried out, has embedded in its base some electronic components, battery, sensors, Raspberry computer and Arduino microcontroller boards. A block diagram of the mobile platform architecture is presented in Figure 3.7. The following subsections will be devoted to describing the components that compose the robot.

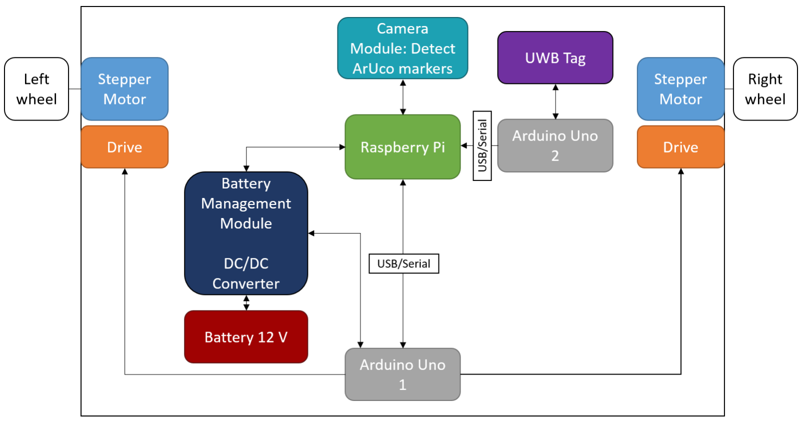


Figure 3.7: Mobile platform architecture

3.2.2.1 Raspberry Pi 3 Model B and Camera Module

The top level consists of a Raspberry Pi 3 Model B (Figure 3.8). This is a microcomputer that runs the raspbian operative system and has a low cost and small size. It has integrated Wi-Fi, avoiding the use of adapters.

The microcomputer is powered by a DC/DC step down converter which has a 12 V battery voltage input and provides 5 V to the Raspberry Pi. The Raspberry Pi runs an application that is responsible for the communication with the remote computer over Wi-Fi and communication with two Arduinos via USB, emulated serial port. It is responsible for receive the data of the Pozyx sensor obtained through the Arduino 2 and receives data to the Arduino 1 that corresponds to the speed and voltage of the motors and battery current. A Raspberry Pi Camera Module is also connected to the Raspberry and is tasked with acquiring positioning information from the ArUco markers placed in the environment. This information is acquired by a python script that runs on the Raspberry Pi and is sent to the remote computer via Wi-Fi.

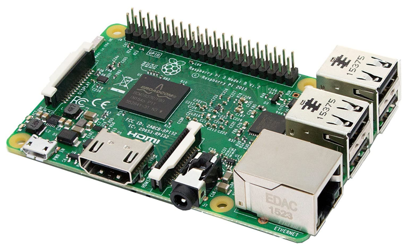


Figure 3.8: Raspberry Pi 3 Model B [12]

3.2.2.2 Arduino UNO

The Arduino Uno (Figure 3.9) is an open-source microcontroller board based on the Microchip ATmega328 microcontroller with a 5 V operating voltage. The robot has in its architecture two Arduinos Uno.

The Arduino 1, depicted in the Figure 3.7, is directly connected to the CNC V3 shield and the Allegro MicroSystems-A4988 chip that makes up the power drive and stepper motor control (discussed in the Subsection 3.2.2.3). The development board is also responsible for odometry calculations and has connection with shields to obtain current, voltage and power management values.

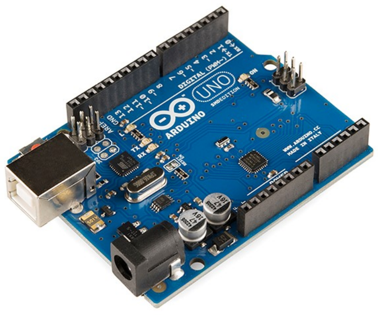


Figure 3.9: Arduino UNO [13]

The Arduino 2 is directly connected to the Pozyx Ultra-WideBand tag (Figure 3.10). Thus, the development board along with the Pozyx tag acquires the data of the four Pozyx anchors scattered around the environment where the robot is, obtaining the distance between the robot and each of the anchors.

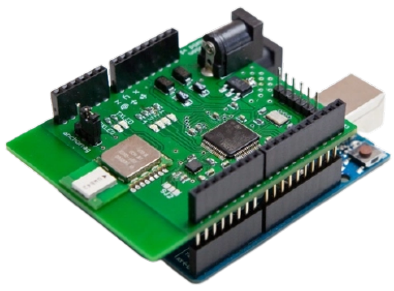


Figure 3.10: Connection between Pozyx tag and Arduino UNO [14]

3.2.2.3 Drive and Stepper Motors

The robot has two NEMA 17HS16 stepper motors, each one attached to each traction wheels. Using these stepper motors it is possible to easily control the speed of the robot's wheels through the steps of the motors. It uses a CNC shield V3 as an interface between the Arduino and the two Allegro MicroSystems-A4988 drivers, each one responsible for each of the motors. In this way, the stepper motor receives a sequence of voltage logic levels that will feed its coil, through which current will pass, resulting in an electromechanical transformation, moving the robot's wheels. Figure 3.11 demonstrate how the Arduino is coupled together with the CNC shield V3 and the Allegro MicroSystem-A4988 chip.

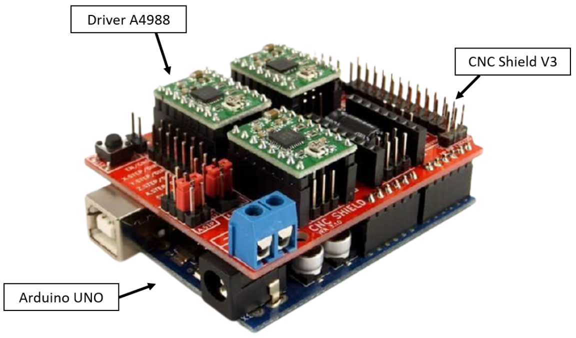


Figure 3.11: Connection between Arduino, CNC shield V3 and Allegro MicroSystems-A4988 chip

3.2.2.4 UWB Pozyx Tag

To obtain an accurate localization of the robot the Pozyx positioning system based on UWB technology will be used, as already mentioned. This system has a total of five components, of which four are immobile anchors scattered around the environment and one mobile tag that is on the robot. In this subsection only the tag will be presented, because it integrates the robot. The Pozyx system will be described in Subsection 3.3 in more detail.

The Pozyx Tag is an Arduino compatible shield that provides positioning and motion information. The Pozyx Tag connects to an Arduino board using long wire-wrap headers that extend through the shield. This keeps the pin layout intact and allows another shield to be stacked on top (Figure 3.12).

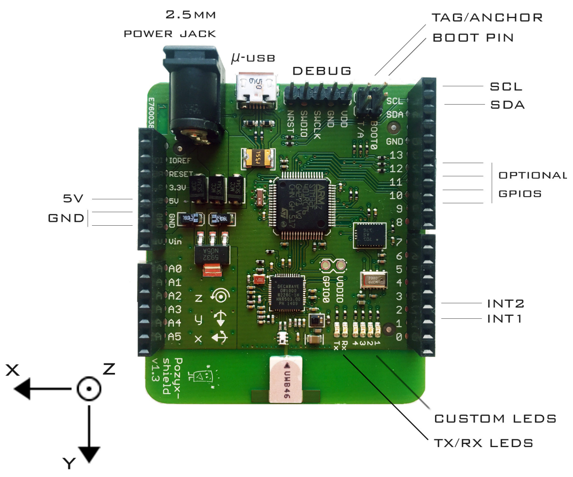


Figure 3.12: Pin-out of Pozyx Tag [14]

3.2.3 Robot Software

The software of the robot includes an application built in Lazarus that is responsible for several operations. This application runs on the Raspberry Pi integrated in the robot and manages communications with the two Arduinos as well as with the remote computer.

The application is responsible for receiving the data from the stepper motors and the data obtained from the Pozyx system. These data are directly forwarded by UDP to the remote computer which performs the sensory fusion through the Kalman Filter, thus estimating the pose of the robot in its environment. From the estimated pose the velocities that each of the wheels must have to perform a given trajectory are calculated. The remote computer then returns the velocity information back to the application running on the Raspberry Pi, which then transmits it to the stepper motors, thus actuating the robot's wheels.

In addition to the application developed in Lazarus, the Raspberry Pi also runs a Python script that acquires frames from the camera connected to the Raspberry Pi. After acquiring a frame, the script processes it in order to detect ArUco markers present in it and then takes information from the detected markers, such as angle and distance in relation to the camera. This information is sent directly to the remote computer via Wi-Fi. In Figure 3.13 it is possible to see a diagram that summarizes everything that has been explained about robot software, including also the communication protocol used for communication between modules.

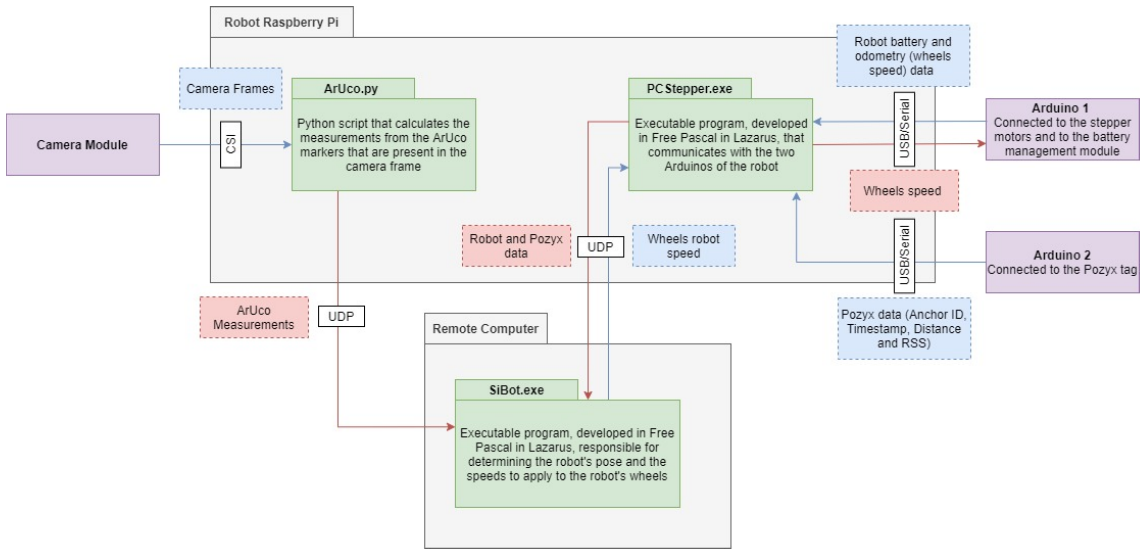

Robot Raspberry Pi
Camera Frames
ArUco.py
Python script that calculates the measurements from the ArUco markers that are present in the camera frame
Camera Module
CSI
PCStepper.exe
Executable program, developed in Free Pascal in Lazarus, that communicates with the two Arduinos of the robot
Robot battery and odometry (wheels speed) data
USB/Serial
Arduino 1
Connected to the stepper motors and to the battery management module
Wheels speed
Robot and Pozyx data
UDP
Wheels robot speed
USB/Serial
Arduino 2
Connected to the Pozyx tag
ArUco Measurements
UDP
Remote Computer
Pozyx data (Anchor ID, Timestamp, Distance and RSS)
SiBot.exe
Executable program, developed in Free Pascal in Lazarus, responsible for determining the robot's pose and the speeds to apply to the robot's wheels

Figure 3.13: Explanatory diagram of the robot software

3.3 Ultra-WideBand Time of Flight System

For the implementation of this work, namely the UWB ranging framework, Pozyx technology [14] was selected, specifically the "Creator System" version. It is a hardware and software RTLS solution that provides robust and accurate positioning, easy to use and easy to integrate. The "Creator System" consists of five modules, a tag that works with an Arduino UNO and four anchors

(Figure 3.14). Each module of the Pozyx system is uniquely and consistently identified, so the problem of data association is immediately solved. The five modules together can provide raw range measurements, position estimates and orientation estimates. Orientation is achieved by an embedded IMU chip that collects additional data using sensors, such as an accelerometer, a gyroscope, a magnetometer, all sensors within the IMU, and a pressure sensor. Concerning the procedure for obtaining the range, it will be used and is based on the two-way ToF technique. However it can also output the RSS of the signals received by the tag.

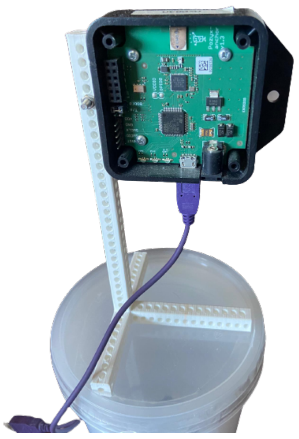


Figure 3.14: Pozyx anchor

The architecture of a Pozyx tag can be seen in Figure 3.15.

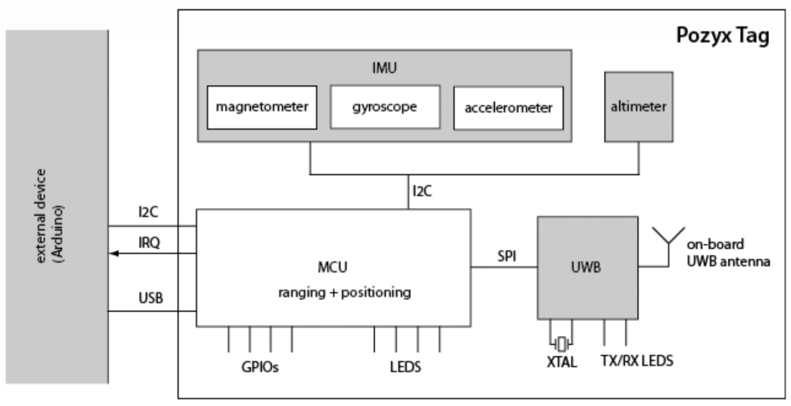


Figure 3.15: Architecture of a Pozyx tag [14]

3.4 Ground-Truth System

To determine the Ground-Truth of the real robot, a camera (Playstation Eye, shown in Figure 3.16) was placed on the center of the ceiling, pointing to the square where the robot moves. This camera is connected to a Raspberry Pi 4 that contains a Python script that obtains the robot's pose.

Figure 3.16: Playstation Eye

In addition, a 13 cm ArUco marker was placed on top of the robot, placed on its center of rotation, as shown in Figure 3.17. The pose regarding the Ground-Truth is useful for determining the accuracy of the developed algorithms by its comparison. The methodology used to determine the robot's pose will be explained in Section 4.3.

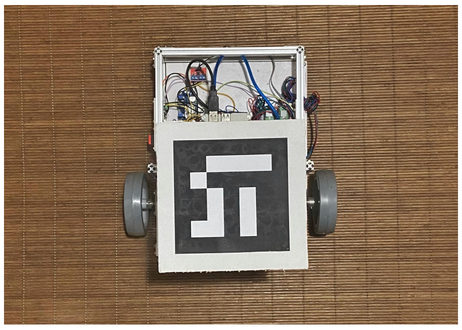


Figure 3.17: Robot with an ArUco Marker

3.5 Simulation Model Description

To test some of the algorithms used to obtain an accurate localization of the mobile robot, the SimTwo simulator was used, a simulator mentioned in the literature review (Section 2.3.4). The choice of this simulator took into account the fact that the real robot software and the user's interface are based on Lazarus (Free Pascal). Since SimTwo is also based on Free Pascal, the code developed can be used in both the real robot and the simulation environment. In this way the exchange between simulation and real environment is simplified. By using this simulator it was possible to validate the approach in a virtual 3D environment before performing the tests for practical validation. Note that the simulator was only used to validate the algorithms that use the Pozyx system and robot's odometry. This section will predefined the model performed for the simulation describing the simulated robot, the scenario and methods used to insert uncertainty into the Pozyx system.

The modeled and simulated prototype is shown in Figure 3.6. The dimensions of the simulated robot are faithful to those of the real robot, having the dimensions described in Table 3.1. The entire structure of the simulated robot was defined using boxes, cylinders and joints (connections between bodies and wheels), which are arranged to form a structure similar to that of the real robot. The simulated robot is represented in Figure 3.18.

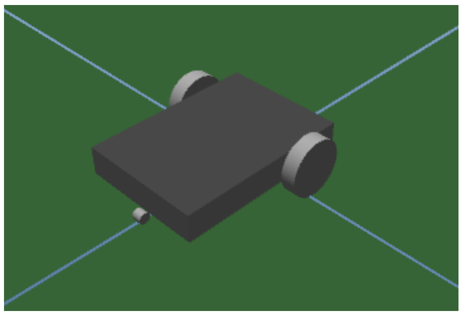


Figure 3.18: Simulated robot

SimTwo has a tab called "Code Editor" that represents the IDE where a Pascal based code is implemented, referring to the actions that the robot will perform inside the simulator.In this tab the UDP communication between the simulator and the interface is also implemented. The odometry information can be sent to the remote computer in the same way as the real robot, this information being the speed of the two drive wheels of the robot.

Real localization systems are subject to errors and uncertainties due to the real sensors. Therefore, to emulate the behavior of the Pozyx system it was necessary to perform some previous tests (Section 5.2.1) on the real robot in order to obtain information of the errors and noise associated with this system. With this information it is possible to develop a model that simulates the behavior of the Pozyx system. Thus, the simulator calculates the distances from the robot to the anchors and then increments a determined uncertainty. Just like in the real robot, this information is then sent to the remote computer. It should be noted that some tests performed and mentioned in Section 5.2.1 allowed understanding how often Pozyx measurements were sent to the remote computer. Thus, in SimTwo this behavior was also simulated so that the performance of the simulator would be as similar as possible to the real performance.

Finally, in the simulator, the robot's Ground-Truth can be obtained directly from the simulator's own functions. So it can directly get x, y and θ from the robot and send this information to the remote computer.

3.6 Communication Layer

Communication is what allows the connection and exchange of messages between the blocks that compose the system. In order to obtain a more robust communication it was used a communication protocol using its own structure for the exchanged messages. The communication between the robot, more specifically the Raspberry Pi, and the remote computer is done wirelessly. However, it should be noted that the Raspeberry Pi on board the robot does not only communicate with the remote computer, but also communicates with two Arduinos, discussed earlier, via USB, emulating serial port. Depending on the type of messages received by the Raspberry Pi, it must treat them in a certain way. When a message is received from the remote computer and it is related to the speeds to be applied to the wheels motors, it is forwarded via serial port to the Arduino which performs this function (Arduino 1). In case one of the Arduinos sends a message to the Raspberry Pi, being this message related to the Pozyx data or to the wheels motors, after being received it is sent back to the remote computer. The data extracted from the ArUco markers, after being acquired, is also sent back to the remote computer.

For the Ground-Truth system the same type of communication was used, but in this case only information corresponding to the robot's pose is sent to the remote computer.

As for the structure of the messages, they are composed of two components. The first is called Channel and is represented by a non-hexadecimal character that indicates what type of message is being sent. Then, eight hexadecimal characters represent the message being sent. The set of these characters is called the value. As a result, each message exchanged between modules has a corresponding size of two bytes. Figure 3.19 presents all the messages sent from each module.

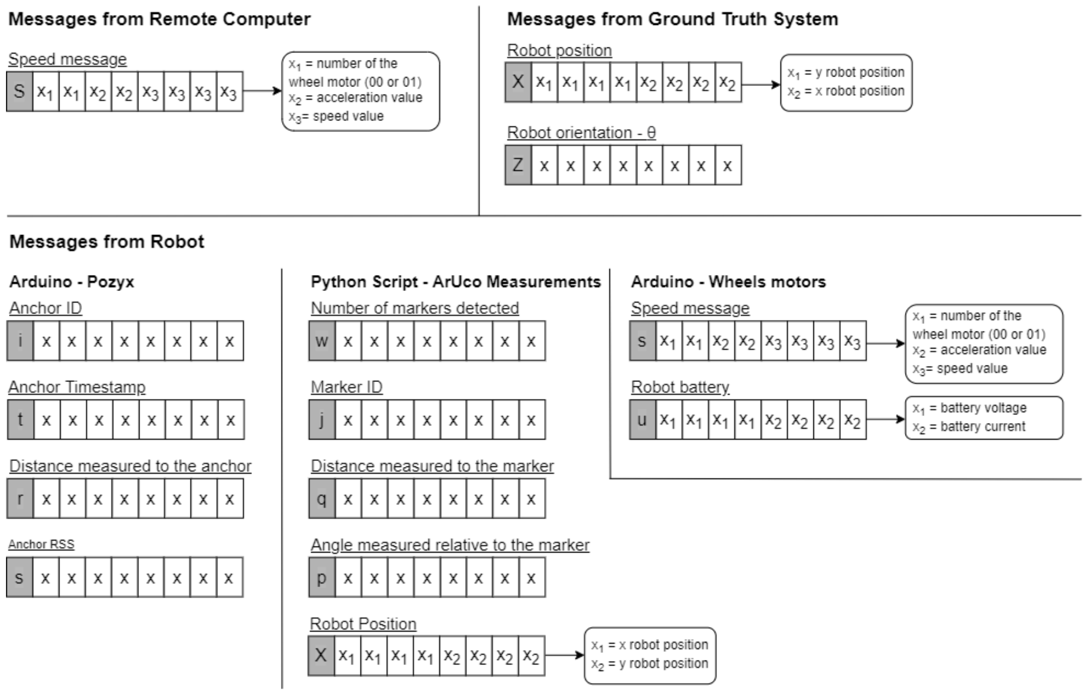


Figure 3.19: Messages exchange between modules

Finally, it is also important to mention that the messages sent and received by the simulator follow the same structure.

Chapter 4

Problem Formulation

In this chapter, the selected and implemented approaches will be presented and their theoretical background explained. First, the Beacon-Based localization formulation will be presented. Next, the kinematics of the robot used will be described. Then, the methodology for determining the Ground-Truth of the robot from a camera will be explained. Then, the approach followed to extract information from the ArUco markers present in the environment will be discussed. Finally, a detailed analysis of the Extended Kalman Filter based localization algorithm used will be conducted.

4.1 2D Beacon-Based Localization

The problem of Robot Beacon-Based Localization, as shown in Figure 4.1, can be defined as the estimate of the Robot's pose $X = \begin{bmatrix} x_r & y_r & \theta_r \end{bmatrix}$ by measuring the distances and/or angles to a given number of beacons, N_B, placed in the environment at known positions $M_{B,i} = \begin{bmatrix} x_{B,i} & y_{B,i} \end{bmatrix}, i \in 1...N_B$. X and $M_{B,i}$ are both specified in the global referential frame $W_X W_Y$.

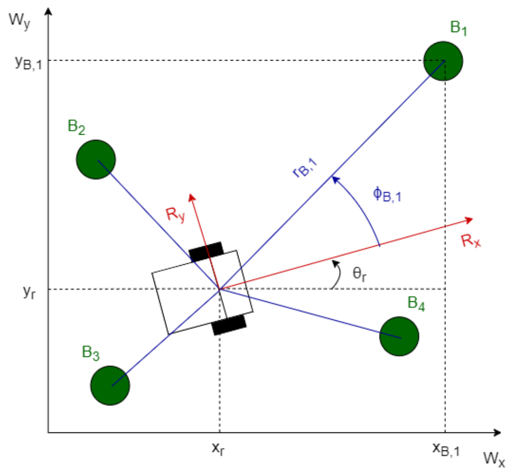


Figure 4.1: 2D localization from measurements to beacons

For the specific problem, the following assumptions were made:

- Since the robot only navigates on the ground without any inclinations, it is assumed that the robot navigates on a two-dimensional plane;
- Beacons positions, $M_{B,i}$, are stationary and previously known;
- Every 40 ms the odometry values are available. This values will be used to provide the system input $u_k = \begin{bmatrix} v_k & \omega_k \end{bmatrix}^T$ and refers to the odometry variation between k and $k-1$ instants;
- $Z_{B,i}(k)$ identifies the measurements relative to Pozyx anchors or ArUco markers i at the instant k.

4.2 Kinematics of Differential Mobile Robots

The robot's advanced kinematics define the odometry equations that calculate the relative motion of the robot. The Equations in 4.1 and 4.2 present the calculation of the angular and linear velocity at instant k, respectively, of both wheels of the robot given a number of *steps* referring to each, and where R_e represents the resolution of the stepper motor. The Equations in 4.3 and 4.4 present the calculation of the linear and angular velocity of the robot, respectively, given the linear velocities of each of its wheels and where b represents the distance between the points of contact of the wheels with the ground.

$$\omega_{R/L,k} = steps \cdot StepsToRad = steps \cdot \frac{2\pi}{R_e} \tag{4.1}$$

$$v_{R/L,k} = \omega_{R/L,k} \cdot WheelRadius \tag{4.2}$$

$$v_k = \frac{v_{R,k} + v_{L,k}}{2} \tag{4.3}$$

$$\omega_k = \frac{v_{R,k} - v_{L,k}}{b} \tag{4.4}$$

The inverse kinematics is also needed to control the robot along a given path. This requires providing the robot with velocities for each of the wheels (left and right) at a given instant k, and which can be calculated as shown in Equations 4.5 and 4.6.

$$v_{R,k} = v_k + \frac{\omega_k \cdot b}{2} \tag{4.5}$$

$$v_{L,k} = v_k - \frac{\omega_k \cdot b}{2} \tag{4.6}$$

4.3 Ground-Truth Explanation

As already mentioned in Section 3.4 a camera is placed on the ceiling and looking down to be able to detect the marker present on the robot, as shown in Figure 4.2. The marker, camera, robot and world referential frames present in the Figure 4.2 are represented by the letters *m*, *c*, *r* and *w* respectively.

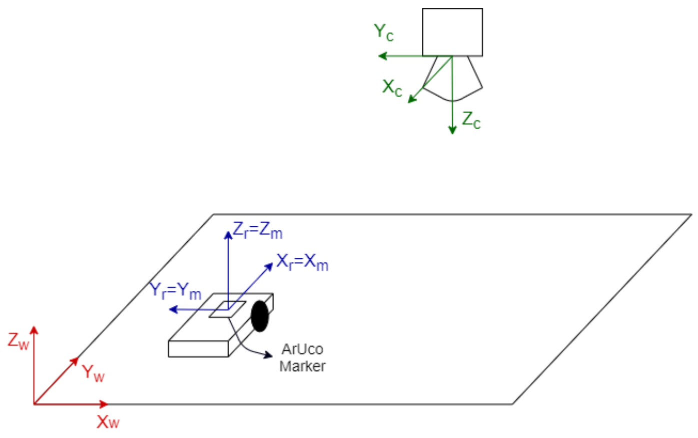


Figure 4.2: Referential frames

A Python script was developed to detect the marker and estimate its pose. This was done using the OpenCV ArUco Marker library [37], and following the [59] tutorial. It is important to emphasize that for the extracted measurements to be reliable, it is necessary to calibrate the camera with the help of a chessboard, as discussed in the literature review, in Section 2.1.2.2. The calibration was performed from scripts developed for this purpose and present in [59].

The estimation of the pose of the marker, i.e. the robot (coincident referential frame), includes the estimation of the marker's position and attitude relative to the camera. For marker detection, each captured frame is analyzed and the integrated OpenCV detection function, $detectMarkers()$ is used for marker detection. The detection procedure is divided into two parts:

1. The image frame is first analyzed by segmentation and thresholding to select the square-shaped objects that were candidates for marker;

2. After the marker candidates are selected, the inner coding is extracted by transformation and Otsu thresholding, so that it is possible to confirm that the marker belongs to the given library, in this case Original ArUco, and to its specific identifier. After the detection is finished is possible to obtain the coordinates of the four corners of the marker and its id.

To estimate the position and attitude of the marker relative to the camera frame the function $estimatePoseSingleMarkers()$ is used. This function has as input parameters the camera matrix, the distortion coefficients, both obtained in the camera calibration, the coordinates of the corners of the previously detected marker and the real size of the marker. As output parameters this function provides the rotation and translation vector that transforms points from the ArUco marker coordinate system to the camera coordinate system. The translation vector contains the position of the marker relative to the camera. To obtain the attitude of the marker relative to the camera, it was used the OpenCV function *Rodrigues* which converts the rotation vector to a rotation matrix, which contains the roll, pitch and yaw angles of the marker relative to the camera.

Then, it is still necessary to convert these measures so that they are referred to the world referential frame. To do this a homogeneous transformation is applied to the translation vector corresponding to the position of the marker referenced to the camera. The matrix that represents the homogeneous transformation between the camera referential frame, (X_c, Y_c, Z_c), and the world referential frame, (X_w, Y_w, Z_w), is present in Equation 4.7, where R_c^w represents the rotation and d_c^w the translation from the camera referential frame to the world, respectively. More specifically, the origin of the world frame is taken as the point $(x, y, z) = (0, 0, 0)$ and the camera is at a distance of 100 cm with respect to x and y and at a height z of 214 cm. The vector corresponding to the position of the robot/marker in the world referential frame is calculated using Equation 4.8, of which only the values of x_m^w and y_m^w are needed. In order to determine the robot's attitude in relation to the world referential frame it is only necessary to apply a rotation to the rotation matrix, R_m^c, given by the function *Rodrigues*, obtained previously, through Equation 4.9. From the resulting matrix it is only necessary to take the yaw angle (θ), that is, the rotation in the z axis, since the robot only has rotation in z.

$$H_c^w = \begin{bmatrix} R_c^w & d_c^w \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 100 \\ -1 & 0 & 0 & 100 \\ 0 & 0 & -1 & 214 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

$$t_m^w = H_c^w \cdot t_m^c = \begin{bmatrix} x_m^w \\ y_m^w \\ z_m^w \\ 1 \end{bmatrix} \tag{4.8}$$

$$R_m^w = R_c^w \cdot R_m^c \tag{4.9}$$

In this way it is then possible to have the localization of the robot (x, y, θ) which will then be compared with the pose extracted from the localization algorithm developed.

4.4 ArUco Markers in the Environment

As already mentioned the system developed is capable of extracting different type of measurements from ArUco markers arranged in the environment (Figure 4.3), i.e angle (corresponding to ϕ angle in Figure 4.1 and Figure 4.4), distance (corresponding to r in Figure 4.1 and d in Figure

4.4) of the markers relative to the referential frame of the robot, and *pitch* angle corresponding to the rotation of the Y_m axis with respect to the Z_r axis.

From the measurements mentioned above and knowing the position and orientation in the world of ArUco markers, (x_m, y_m, θ_m), it is possible to determine directly the (x_r, y_r) position of the robot. However, after some tests, presented in Section 5.2.2, it was found that these measurements (x_r and y_r) are not as reliable as the measurements of distance d and angle ϕ, i.e. they present greater error. For this reason these measurements will not be taken into account in the proposed algorithm.

Each of the markers has an associated identification number that can be extracted when the marker is detected in an image, which distinguishes each reference point.

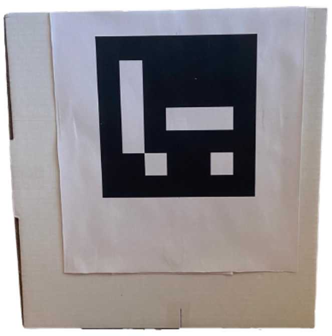


Figure 4.3: ArUco marker in the environment

Using the OpenCV ArUco Marker library provided by the authors [37], it is possible to obtain the rotation and translation of the ArUco marker referential frame to the camera referential frame, through the same procedure mentioned in Section 4.3. Since it is necessary to have the measures in the referential of the robot and not of the camera, and these referential are not coincident, as verified in Figure 4.4 it is necessary to apply a homogeneous transformation in order to obtain the measures of the markers relative to the frame referential of the robot. The matrix that represents this transformation is shown in Equation 4.10, where R_c^r represents the rotation and d_c^r the translation from the camera frame referential to the robot referential frame, respectively. The vector corresponding to the position of the markers in the robot referential is calculated using Equation 4.11. Since the movement of the robot is only in 2D, just the values of x_m^r and y_m^r are needed to calculate the ϕ angle and distance d of the marker relative to the robot.

To determine the *pitch* angle it is only necessary to use the *Rodrigues* function, mentioned in Section 4.3, converting the rotation vector referring to the ArUco marker into a rotation matrix. From the rotation matrix it can be extract the *pitch* angle.

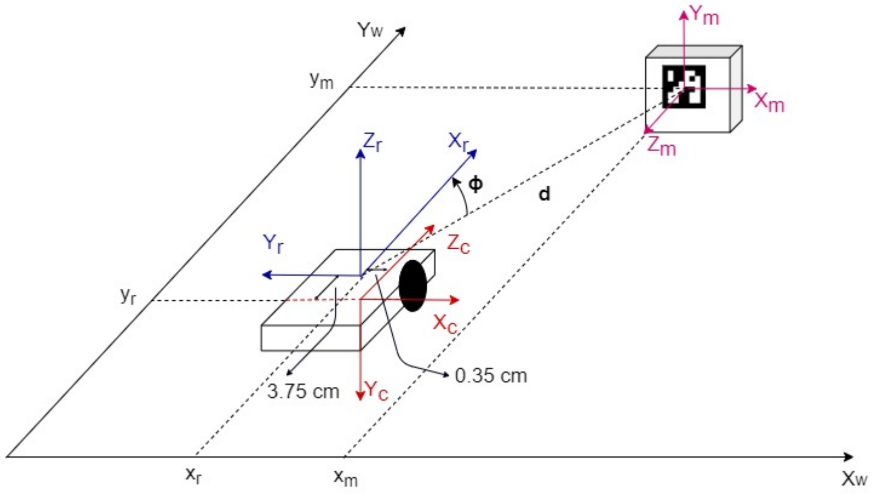


Figure 4.4: ArUco measurements

$$H_c^r = \begin{bmatrix} R_c^r & d_c^r \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & -3.75 \\ -1 & 0 & 0 & -0.35 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

$$t_m^r = H_c^r \cdot t_m^c = \begin{bmatrix} x_m^r \\ y_m^r \\ z_m^r \\ 1 \end{bmatrix} \tag{4.11}$$

The angle ϕ and the distance d from the marker to the robot can be extracted by Equations 4.12 and 4.13, respectively.

$$\phi = atan2(\frac{y_m^r}{x_m^r}) \tag{4.12}$$

$$d = \|(y_m^r, x_m^r)\|_2 \tag{4.13}$$

The (x_r, y_r) position of the robot can be determined from Equations 4.14 and 4.15, in which θ_m represents the orientation of the ArUco marker relative to the world referential frame.

$$x_r = x_m + d \cdot cos(pitch + \phi + \theta_m) \tag{4.14}$$

$$y_r = y_m + d \cdot sin(pitch + \phi + \theta_m) \tag{4.15}$$

Lastly, it is important to emphasize that, as with the Playstation Eye camera, the robot's camera also needs to be calibrated so that the measurements taken are reliable.

4.5 Extended Kalman Filter Localization

Through the distance measurements from the Pozyx tag to the anchors or the measurements regarding the angles and distances of ArUco markers it is not possible to obtain directly the pose of the robot, i.e. (x_r, y_r, θ_r). Therefore, it was necessary to choose a localization algorithm to implement. It will be used the known EKF, which is widely used in this type of problems. It is important to note that in the case of mobile robotics, the normal Kalman Filter could not be applied because these are nonlinear functions, its extended model is required. The EKF is relatively simple to implement and has a low computational cost, thus allowing to obtain localization information in real time. In addition to this, it is possible to add more data from other sensors or technologies as it is a flexible and expandable fusion algorithm.

In this work, the odometry data is fused with distance measurements to each Pozyx anchor and ϕ angle and distance measurements regarding ArUco markers. Through odometry it is possible to obtain information about the robot's motion at high frequency. The remaining information obtained from the other systems, i.e. Pozyx and ArUco, is absolute, independent of the past, and the errors are not cumulative over time, unlike with odometry.

The EKF models the robot's state as a Gaussian Distribution $X_k \sim N(\mu_{X_k}, \sum_{X_k})$, in which μ_{X_k} represents the mean pose value of the robot and $\sum_{X_k}$ the corresponding covariance matrix. The fusion between the odometry model and the Pozyx and ArUco measurements is divided into two phases. The Prediction phase, that estimates the actual pose from the State Transition Model, presented in Section 4.5.1, and the Correction phase that corrects the prediction state input with the Pozyx distance and ArUco ϕ angle and distance measurements in the Observation model, presented in Section 4.5.2.

4.5.1 State Transition Model

The state transition model, present in Equation 4.16, is a probabilistic model that describes the state distribution according to the inputs (u_k), in this case the linear (v_k) and angular (ω_k) velocity of the robot. In a robot localization problem, this model refers to the kinematic model, as it defines the probability of a given state in relation to the relative motion of the robot. In addition to the transition function $f(X_k, u_k)$, presented in the Equation 4.17, the noise model N is represented by a Gaussian distribution with zero mean and covariance Q (4.18), which is assumed constant. The noise model N is intended to represent the uncertainty associated with the odometry readings.

$$X_{K+1} = f(X_k, u_k) + N(0, Q) \tag{4.16}$$

$$f(X_k, u_k) = X_k + \begin{bmatrix} v_k \cdot \Delta t \cdot cos(\theta_k + \frac{\omega_k \cdot \Delta t}{2}) \\ v_k \cdot \Delta t \cdot sin(\theta_k + \frac{\omega_k \cdot \Delta t}{2}) \\ \omega_k \cdot \Delta t \end{bmatrix} \tag{4.17}$$

$$Q = \begin{bmatrix} \sigma_{v_k}^2 & 0 \\ 0 & \sigma_{\omega_k}^2 \end{bmatrix} \tag{4.18}$$

To linearize the process EKF uses the Taylor expansion, which involves the computation of the Jacobian of f in relation to X_k (4.19) and in relation to u_k (4.20).

$$\nabla f_x = \frac{\partial f}{\partial X_k} = \begin{bmatrix} 1 & 0 & -v_k \cdot \Delta t \cdot sin(\theta_k + \frac{\omega_k \cdot \Delta t}{2}) \\ 0 & 1 & v_k \cdot \Delta t \cdot cos(\theta_k + \frac{\omega_k \cdot \Delta t}{2}) \\ 0 & 0 & 1 \end{bmatrix} \tag{4.19}$$

$$\nabla f_u = \frac{\partial f}{\partial u_k} = \begin{bmatrix} \Delta t \cdot cos(\theta_k + \frac{\omega_k \cdot \Delta t}{2}) & -\frac{1}{2} \cdot v_k \cdot \Delta t^2 \cdot sin(\theta_k + \frac{\omega_k \cdot \Delta t}{2}) \\ \Delta t \cdot sin(\theta_k + \frac{\omega_k \cdot \Delta t}{2}) & \frac{1}{2} \cdot v_k \cdot \Delta t^2 \cdot cos(\theta_k + \frac{\omega_k \cdot \Delta t}{2}) \\ 0 & \Delta t \end{bmatrix} \tag{4.20}$$

4.5.2 Observation Model

Similarly, the correction phase of the EKF will need a model to handle with the input measurements, the Observation Model. The filter can have three types of information inputs to the filter: distance measurement between the Pozyx tag and each anchor and ϕ angle and distance measurement of ArUco markers to the robot referential frame. Therefore, it was decided to consider three different observation models:

- One for the measurement of the distance between the Pozyx tag and each anchor;
- Another to consider only the measurement of the ϕ angle of ArUco markers to the robot's referential frame;
- The other one to consider the measurement of ϕ angle and distance from ArUco markers to the robot's referential frame.

Both models will have in common the covariance matrix of the state estimate of the current state of the filter.

The model corresponding to the Pozyx measurements is presented in Equation 4.21. This model characterizes the euclidean distance between the estimated position of the robot and the position of the anchor i, as shown in Equation 4.22. As with the state transition model, the process is affected by an additive Gaussian noise $N(0,R)$ with zero mean and covariance R, a parameter that must be changed according to the error characteristics of the sensor.

$$Z_{B,i} = r_{B,i} = h(M_{B,i}, X_k) + N(0,R) \qquad R = [\sigma_r^2] \tag{4.21}$$

$$\hat{Z}_{B,i} = h(M_{B,i}, \hat{X}_k) = \sqrt{(x_{B,i} - \hat{x}_{r,k})^2 + (y_{B,i} - \hat{y}_{r,k})^2} \tag{4.22}$$

It is still necessary to calculate the Jacobian of h with respect to X_k for linearization of the system (Equation 4.23).

$$\nabla h_x = \frac{\partial h}{\partial X_k} = \begin{bmatrix} -\frac{x_{B,i}-\hat{x}_{r,k}}{\sqrt{(x_{B,i}-\hat{x}_{r,k})^2+(y_{B,i}-\hat{y}_{r,k})^2}} & -\frac{y_{B,i}-\hat{y}_{r,k}}{\sqrt{(x_{B,i}-\hat{x}_{r,k})^2+(y_{B,i}-\hat{y}_{r,k})^2}} & 0 \end{bmatrix} \tag{4.23}$$

The model corresponding to the ArUco ϕ angle measurements, on the other hand, is presented in Equation 4.24 and characterizes the angle between the robot's referential and the ArUco marker i (Equation 4.25). One more time, the process is affected by an additive Gaussian noise $N(0,R)$ with zero mean and covariance R, which must be adjusted according to the error obtained by the camera.

$$Z_{B,i} = \phi_{B,i} = h(M_{B,i}, X_k) + N(0,R) \qquad R = [\sigma_\phi^2] \tag{4.24}$$

$$\hat{Z}_{B,i} = h(M_{B,i}, \hat{X}_k) = atan2(y_{B,i} - \hat{y}_{r,k}, x_{B,i} - \hat{x}_{r,k}) - \hat{\theta}_{r,k} \tag{4.25}$$

In this case the Jacobian of h with respect to X_k is calculated from Equation 4.26.

$$\nabla h_x = \frac{\partial h}{\partial X_k} = \begin{bmatrix} \frac{y_{B,i}-\hat{y}_{r,k}}{(x_{B,i}-\hat{x}_{r,k})^2+(y_{B,i}-\hat{y}_{r,k})^2} & -\frac{x_{B,i}-\hat{x}_{r,k}}{(x_{B,i}-\hat{x}_{r,k})^2+(y_{B,i}-\hat{y}_{r,k})^2} & -1 \end{bmatrix} \tag{4.26}$$

Finally, the model corresponding to the ϕ angle and distance measurements obtained from ArUco markers is present in Equation 4.27 and characterizes the distance and angle between the robot's referential frame and the ArUco marker i (Equation 4.28). The process, as with the other models, is affected by an additive Gaussian noise with zero mean and covariance R, which is adjusted according to the error obtained in the measurements taken by the camera.

$$Z_{B,i} = \begin{bmatrix} r_{B,i} \\ \phi_{B,i} \end{bmatrix} = h(M_{B,i}, X_k) + N(0,R) \qquad R = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix} \tag{4.27}$$

$$\hat{Z}_{B,i} = h(M_{B,i}, \hat{X}_k) = \begin{bmatrix} \sqrt{(x_{B,i}-\hat{x}_{r,k})^2+(y_{B,i}-\hat{y}_{r,k})^2} \\ atan2(y_{B,i} - \hat{y}_{r,k}, x_{B,i} - \hat{x}_{r,k}) - \hat{\theta}_{r,k} \end{bmatrix} \tag{4.28}$$

In this case the Jacobian of h with respect to X_k is calculated from Equation 4.29.

$$\nabla h_x = \frac{\partial h}{\partial X_k} = \begin{bmatrix} -\frac{x_{B,i}-\hat{x}_{r,k}}{\sqrt{(x_{B,i}-\hat{x}_{r,k})^2+(y_{B,i}-\hat{y}_{r,k})^2}} & -\frac{y_{B,i}-\hat{y}_{r,k}}{\sqrt{(x_{B,i}-\hat{x}_{r,k})^2+(y_{B,i}-\hat{y}_{r,k})^2}} & 0 \\ \frac{y_{B,i}-\hat{y}_{r,k}}{(x_{B,i}-\hat{x}_{r,k})^2+(y_{B,i}-\hat{y}_{r,k})^2} & -\frac{x_{B,i}-\hat{x}_{r,k}}{(x_{B,i}-\hat{x}_{r,k})^2+(y_{B,i}-\hat{y}_{r,k})^2} & -1 \end{bmatrix} \tag{4.29}$$

4.5.3 Implementation

After presenting the theoretical and mathematical concepts outlined above, it is possible to describe the localization algorithm based on the EKF used (Algorithm 1).

The algorithm begins with the prediction phase, in which the State Transition Model is applied and linearized (lines 1:3). The prediction of the new state covariance P_{k+1} is calculated from the current uncertainty P_k and the propagation of the state and motion data around in the current state (line 4). If there were no state correction from any of the measurements from the Pozyx or ArUco systems, the filter accumulates the odometry errors, and P_{k+1} would gradually increase. However, whenever a new measurement from one of these systems is received, the location uncertainty is decreased.

When new data is received for either Pozyx or ArUco system, the algorithm cycles through each detected measurement (lines 5:15). In other words, in case new measurements relative to the Pozyx system are received, N_B cycles are performed, each cycle referring to each Pozyx anchor. If new measurements on the ArUco system are received, N_B cycles are also performed, but in this case N_B represents the number of markers detected. Depending on the system that is being treated the equations for $\hat{Z}_{B,i}$ and ∇h_x are different and equal to those presented in Section 4.5.2.

In lines 10:11 it is possible to verify that the update of the position estimate is only performed in case the measurement is not considered an outlier. In the case that the measurement is not considered an outlier, the state is updated relative to the Kalman gain $K_{k,i}$ (lines 12:14). It should be noted that the concept of outlier and how they are determined is explained in Section 4.5.4.

Finally, if the new data refer to measurements from the Pozyx system, an offset correction is made (line 7), which is explained in Section 4.5.5.

Algorithm 1: EKF Localization Algorithm

```
   Input  : $\hat{X}_k, Z_{B,i}, u_k, P_k, R, Q$
   Output : $\hat{X}_{k+1}, P_{k+1}$

   PREDICTION
1  $\hat{X}_{k+1} = f(\hat{X}_k, u_k)$
2  $\nabla f_x = \frac{\partial f}{\partial X_k}$
3  $\nabla f_u = \frac{\partial f}{\partial u_k}$
4  $P_{k+1} = \nabla f_x \cdot P_k \cdot \nabla f_x^T + \nabla f_u \cdot Q \cdot \nabla f_u^T$

   CORRECTION
5  for $i = 1$ to $N_B$ do
6      $\hat{Z}_{B,i} = h(M_{B,i}, \hat{X}_k)$
7      $\hat{Z}_{B,i} = CorrectOffset(\hat{Z}_{B,i})$ // only for Pozyx measurements
8      $\nabla h_x = \frac{\partial h}{\partial X_k}$
9      $S_{k,i} = \nabla h_x \cdot P_k \cdot \nabla h_x^T + R$
10     $OutlierDetected = OutlierDetection(S_{k,i}, \hat{Z}_{B,i}, Z_{B,i})$
11     if not $OutlierDetected$ then
12         $K_{k,i} = P_{k+1} \cdot \nabla h_x^T \cdot S_{k,i}^{-1}$
13         $\hat{X}_{k+1} = \hat{X}_k + K_{k,i} \cdot [Z_{B,i} - \hat{Z}_{B,i}]$
14         $P_{k+1} = [I - K_{k,i} \cdot \nabla h_x] \cdot P_{k+1}$
15     end
16 end
```

4.5.4 Outliers Detection

Normally, when taking measurements with sensors, they are prone to errors that can be generated due to hardware failures or unexpected changes in the environment. For proper use of the measurements obtained, the systems must be able to detect and deal with the less reliable measurements, i.e., those that deviate from the normal pattern and called outlier. Although there are several approaches to solving this problem, in the presence of multivariate data, one of the common methods is to use the Mahalanobis Distance (MD) (Equation 4.30) as a statistical measure of the probability of some observation belonging to a given dataset. In this approach, outliers are removed as long as they are outside a specific ellipse that symbolizes a specific probability in the distribution (Algorithm 2). Consequently, different ellipses represent different MD's and therefore different probability thresholds. In [60, 61, 26], it can be seen that this approach has been used successfully in different applications, consisting of calculating the normalized distance between a point and the entire population. It would be possible to use Euclidean Distance, however the use of MD considers not only the distance to the mean but also the direction, i.e. the covariance. Indeed, MD is only the same as the Euclidean distance if the covariance matrix is the identity, which means that all variables are independent and have the same variance.

$$MD(x) = \sqrt{(x-\mu)^T \cdot S^{-1} \cdot (x-\mu)} \tag{4.30}$$

In this case $\hat{Z}_{B,i}$ represents the mean population μ and S is a factor representing a combination of the state uncertainty and the actual sensor measurement, present in line 9 of Algorithm 1. The choice of the threshold value can be made by a simple analysis of the data by the determination of some probabilistic statistical parameter. In this case the threshold was established according to the χ^2 probability table, in a way that observations with less than 5 % probability were cut off in the case of the Pozyx system. For the ArUco system, more specifically for the ϕ angle measurements, a threshold was chosen, in order to cut off the observations with less than 2.5 % probability. These values were chosen through several iterations of the algorithm, and the values, which the algorithm behaved best for, were chosen.

Algorithm 2: Outliers Detection Algorithm

```
Input  : $S_{k,i}, \hat{Z}_{B,i}, Z_{B,i}$
Output : OutlierDetected
1 $MD(Z_{B,i}) = \sqrt{(Z_{B,i} - \hat{Z}_{B,i})^T \cdot S_{k,i}^{-1} \cdot (Z_{B,i} - \hat{Z}_{B,i})}$
2 if $MD(Z_{B,i}) > Threshold$ then
3     OutlierDetected = True
4 else
5     OutlierDetected = False
6 end
```

4.5.5 Pozyx Offset Correction

The Pozyx system distance measurements suffer from different offsets, i.e. difference between the actual value and the measured value, depending on how far the Pozyx tag is from a given anchor. These offsets also vary depending on the anchor at which the distance is measured, as verified in Section 6.1. In order to correct this offset, the Algorithm 3 was implemented. After calculating the estimated distance from the anchor to the robot tag, it is verified between which predefined values this distance is, and then the correction is made through a linear interpolation. The variable *RealDist* is an array with a set of predefined distances for which the offsets have been determined. The variable $Offset$ is a $4 \times length(RealDist)$ matrix containing the offsets corresponding to each distance for each of the four anchors placed in the environment.

Algorithm 3: Pozyx Offset Correction Algorithm

```
Input  : $\hat{Z}_{B,i}$
Output : $\hat{Z}_{B,i}$

1  if $\hat{Z}_{B,i} <= RealDist[0]$ then
2      $\hat{Z}_{B,i} = \hat{Z}_{B,i} + Offset[i,0]$
3  else if $\hat{Z}_{B,i} >= RealDist[7]$ then
4      $\hat{Z}_{B,i} = \hat{Z}_{B,i} + Offset[i,7]$
5  else
6      for $j = 0$ to $length(RealDist) - 1$ do
7          if $\hat{Z}_{B,i} >= RealDist[j]$ and $\hat{Z}_{B,i} <= RealDist[j+1]$ then
8              $OffsetLin = Offset[i,j] + \frac{Offset[i,j+1] - Offset[i,j]}{RealDist[j+1] - RealDist[j]} \cdot (\hat{Z}_{B,i} - RealDist[j])$
9              $\hat{Z}_{B,i} = \hat{Z}_{B,i} + OffsetLin$
10         end
11     end
12 end
```

Chapter 5

Experimental Methodology

In this chapter the different types of tests performed will be described. First, the developed trajectory control will be presented in order to later be able to perform the different localization tests. Then, the tests performed in the simulation and real environment are presented.

5.1 Trajectory Control

In order to test the developed algorithms it was necessary make the robot perform a given trajectory. This trajectory can be defined by a set of points (x, y) organized in a vector of the type: $[x_1, y_1, x_2, y_2, ..., x_n, y_n]$, where n is the number of points that define the trajectory. Each pair of consecutive points will define a line segment with a given start point (x_i, y_i) and an end point (x_f, y_f). These points are then used in the Follow Line procedure shown in Figure 5.1, which makes the robot move along the line segment defined by the points. When the robot reaches the first end point, it moves forward in the vector to the next pair of consecutive points.

To calculate the distance the robot is from the line segment defined by a given pair of points, the Equation 5.1 is used. This equation requires the prior calculation of u_x and u_y, which form the $\hat{u} = (u_x, u_y)$ vector corresponding to the direction of the line segment. The calculation of these two components is in Equations 5.2 and 5.3. The Equations 5.4 and 5.5 correspond to the errors of orientation and distance of the current position of the robot in relation to the desired end point, respectively. The Follow Line procedure still requires the use of several constants: LIN_VEL_NOM, W_NOM, LIN_VEL_DA, $GAIN\_DIST$, $GAIN\_FWD$, which are used to adjust the linear and angular velocity at which the robot must move. In addition to the constants mentioned above, three more are used: $DIST\_DA$, $TOL\_FINDIST$ and $DIST\_NEWPOSE$, which are used to make the state machine represented in Figure 5.1, advance from one given state to another, more specifically they represent tolerances of the robot's distance error.

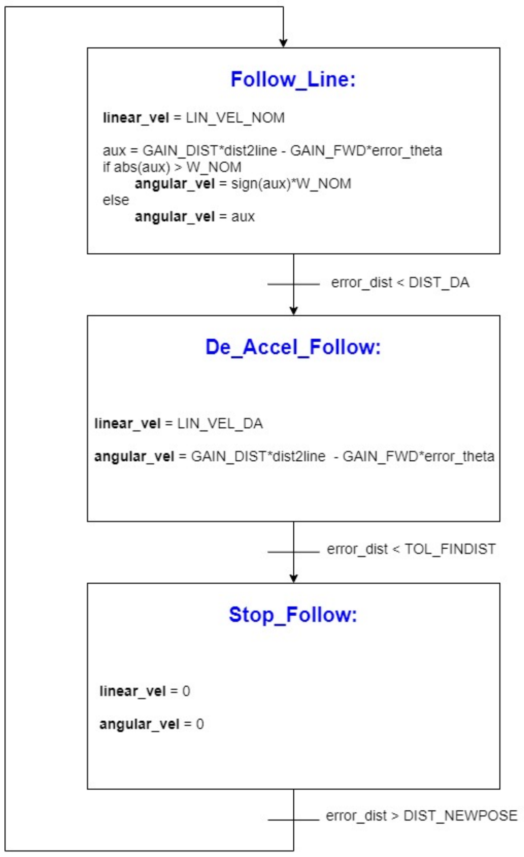
$$dist2line = \frac{(\hat{x}_{r,k} - x_i) \cdot u_y + (y_i - \hat{y}_{r,k}) \cdot u_x}{u_x^2 + u_y^2} \tag{5.1}$$

$$u_x = \frac{x_f - x_i}{\sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}} \tag{5.2}$$

$$u_y = \frac{y_f - y_i}{\sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}} \tag{5.3}$$

$$error\_theta = \hat{\theta}_{r,k} - atan2(y_f - \hat{y}_{r,k}, x_f - \hat{x}_{r,k}) \tag{5.4}$$

$$error\_dist = \sqrt{(\hat{x}_{r,k} - x_f)^2 + (\hat{y}_{r,k} - y_f)^2} \tag{5.5}$$



```
Follow_Line:
linear_vel = LIN_VEL_NOM
aux = GAIN_DIST*dist2line - GAIN_FWD*error_theta
if abs(aux) > W_NOM
      angular_vel = sign(aux)*W_NOM
else
      angular_vel = aux

De_Accel_Follow:
linear_vel = LIN_VEL_DA
angular_vel = GAIN_DIST*dist2line  - GAIN_FWD*error_theta

Stop_Follow:
linear_vel = 0
angular_vel = 0
```

Figure 5.1: Follow line procedure

5.2 Experimental Tests

In order to characterize the entire developed system and evaluate its performance, several tests were performed.

In characterizing system performance, it is worth noting that an error can have different sources and can be of different types. The experiments performed in the characterization of the systems

Pozyx and ArUco will focus mainly on the analysis of accuracy and precision, which are typically associated with systematic and random errors respectively. In the case of this particular work the characterization of sensor accuracy is particularly important, since it will be used in the EKF to characterize the noise measurement. Furthermore, accuracy measures how close the observations are to real values, thus characterizing the existing offset. So, first, some experiments were performed in order to characterize the Pozyx system and the error and noise involved in the measurements. Next, a set of experiments were also performed in order to characterize the ArUco system, the error and noise in the measurements. Similarly, the developed Ground-Truth system was characterized in order to conclude whether it obtained reliable measurements.

Finally, a set of localization tests were performed, using the results obtained from the previous experiments. The GUI developed and presented in Section 3.1 was used to observe all the results obtained.

5.2.1 Pozyx Error Characterization

In order to fully characterize the Pozyx system, several experiments were performed in an indoor environment, the same one where the localization tests were performed. For this purpose, distance measurements were performed between a static anchor and the Pozyx tag embedded in the robot, as shown in Figure 5.2. The distances between tag and anchor for which the experiments were performed were as follows: $[0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2]$ meters. These experiments were repeated for the four different anchors used in the work. It should also be noted that both the anchors and the tag are at the same height from the ground and that the actual distance (Ground-Truth) was measured with an appropriate laser sensor, a laser that has an accuracy of 1 mm. With these experiments is possible to conclude on the error/offset and standard deviation of the measurements and verify their evolution as a function of the distance between the anchor and the tag.

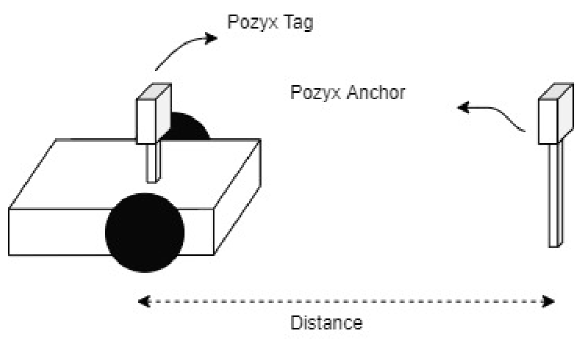


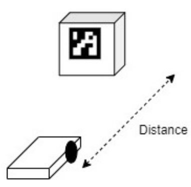
Figure 5.2: Pozyx distance measurement experiment

In addition to the above experiments, another experiment was performed to verify how often the Pozyx measurements reached the remote computer. To perform this experiment, a set of Pozyx measurements were considered, and then the average of the time interval with which these measurements reached the remote computer was calculated. By determining this frequency, along with the noise determined in previous experiments, it is then possible to emulate the behavior of the Pozyx system in the SimTwo simulator.

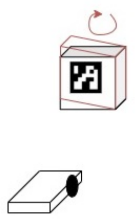
5.2.2 ArUco Error Characterization

For the characterization of the ArUco system a similar procedure was followed as for the experiments performed of the Pozyx system. However, for the ArUco system was necessary to characterize five different types of measurements, the distance between the referential frame of the robot and the ArUco marker, the ϕ angle, the $pitch$ angle and the position of the robot (x_r, y_r), measurements that are discussed in Section 4.4.

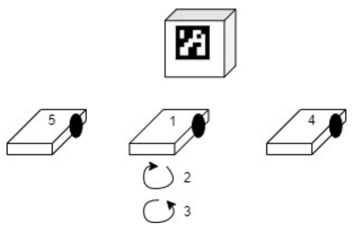
For the experiments related to distance, the robot was placed at the following distances from a given ArUco marker: $[0.5, 1, 1.5, 2]$ meters, as shown in Figure 5.3a. Concerning the ϕ angle, five different experiments were performed, demonstrated in Figure 5.3c: robot perpendicular and just in front of the marker (1), in front of the marker but rotated with a given angle to the right (2), in front of the marker but rotated with a given angle to the left (3), perpendicular to the marker but translated to the right (4) and finally, perpendicular to the marker but translated to the left (5). Regarding the $pitch$ angle, four different experiments were performed. The first one with the marker in front of the robot, that is, with a $pitch$ angle of $0°$. For the next three, the marker was tilted to obtain a rotation of the Y_m axis of the marker of $20°$, $30°$ and $40°$, respectively. These experiments are demonstrated in Figure 5.3b. Note that these experiments with the $pitch$ angle are useful, because this angle can be used to calculate the position of the robot (x_r, y_r), along with other measurements, i.e distance and ϕ angle. Therefore, it is important to understand if these measurements are accurate.



(a) ArUco distance experiments



(b) ArUco $pitch$ angle experiments



(c) ArUco ϕ angle experiments

Figure 5.3: ArUco measurements experiments

Finally, for the measurements of the robot position (x_r, y_r), four different experiments were performed. In these experiments the robot was placed in four different positions, represented in Figure 5.4, in which only certain markers were visible by the robot's built-in camera.

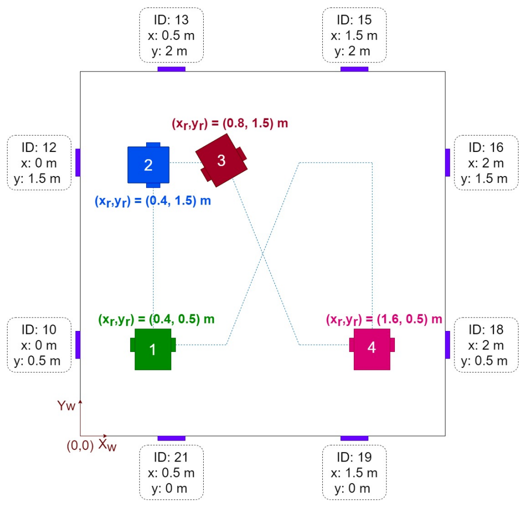


Figure 5.4: ArUco x_r, y_r measurements experiments

With these experiments it is possible to conclude about the noise, characterized by the standard deviation, and error in obtaining the measurements discussed above. Note that the center of the camera and the ArUco markers are at the same height from the ground. In addition, it should be noted that the real value (Ground-Truth) of these measurements was obtained with the same laser sensor used in the characterization of the Pozyx system.

5.2.3 Ground-Truth Characterization

In order to verify if the Ground-Truth system developed obtained accurate values, a characterization of it was performed. For this, several experiments were performed where the robot was placed in several poses in the environment, as shown in Figure 5.5. With these experiments it is possible to conclude about the error of this system.

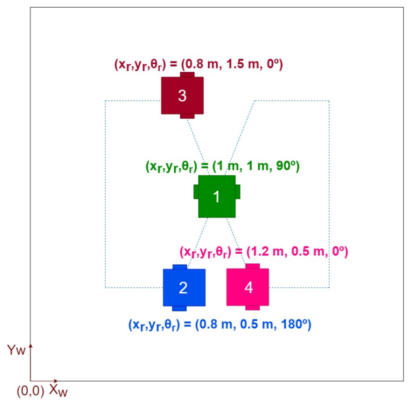

(xr,yr,θr) = (0.8 m, 1.5 m, 0º)
3
(xr,yr,θr) = (1 m, 1 m, 90º)
1
(xr,yr,θr) = (1.2 m, 0.5 m, 0º)
2
4
(xr,yr,θr) = (0.8 m, 0.5 m, 180º)
Yw
(0,0) Xw

Figure 5.5: Ground-Truth x_r, y_r, θ_r measurements experiments

5.2.4 Localization Tests

With the experiments for noise and error characterization of the Pozyx and ArUco systems already completed, it is possible to proceed to a set of tests to verify the behavior of the various variants of the developed EKF based localization system.

The developed localization system was tested on the robot in a 2 m $\times$ 2 m area (Figure 5.6). Pozyx anchors were placed at each of the corners of the square and ArUco markers at the following (x, y, θ) poses (with x and y in meters and θ in degrees): $(0, 0.5, 0)$, $(0, 1.5, 0)$, $(0.5, 2, -90)$, $(1.5, 2, -90)$, $(2, 1.5, 180)$, $(2, 0.5, 180)$, $(1.5, 0, 90)$ and $(0.5, 0, 90)$. The initial pose (x, y, θ) (with x and y in meters and θ in degrees) of the robot is $(0.4, 0.5, 90)$ and it moves clockwise. Different tests were performed in the two available environments, the simulated environment with the SimTwo simulator and the real environment.

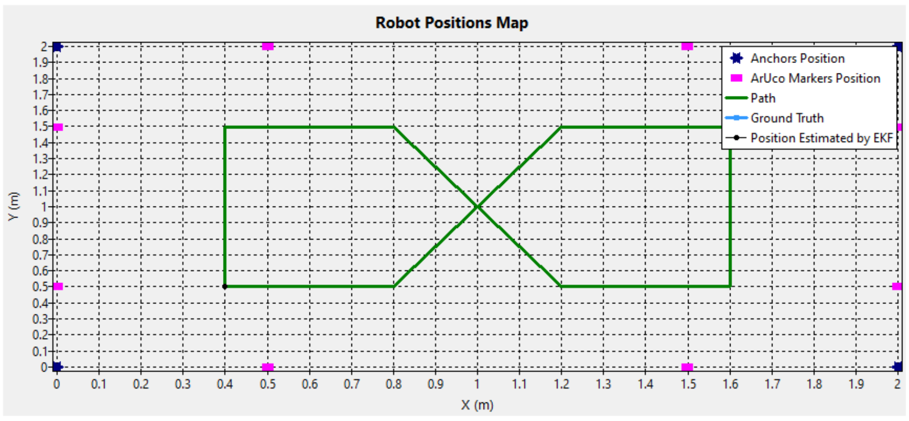


Figure 5.6: Robot positions map

The real environment can be observed in Figure 5.7.

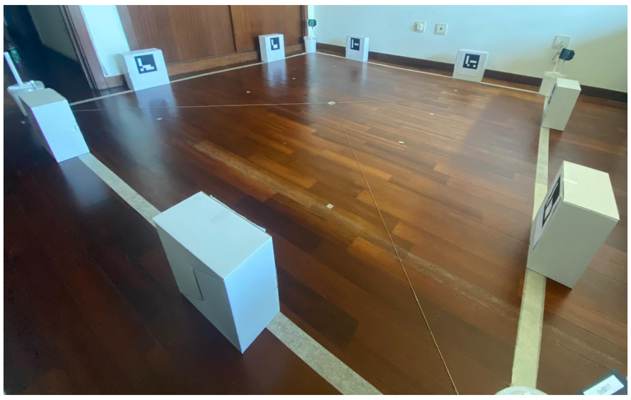


Figure 5.7: Real environment

5.2.4.1 Simulator Localization Tests

As already mentioned in Section 3.5, in the simulated environment only tests related to odometry and the Pozyx system were performed. The tests performed and explained below allowed to test various parts of the developed EKF. Consequently, the following sequence of tests was performed:

1. **Test with the robot stopped at the initial pose and with the Pozyx system on:** This test aims to verify if the filter was well designed, initializing it with a different position from the real one, and then verifying if it converges to the correct one. The test also allows to verify if the Pozyx system modeled in SimTwo is working properly and if the standard deviation used for noise fits the system.

2. **Test with the robot in motion and with the Pozyx system turned off:** The purpose of this test is to test the prediction phase of the EKF by being able to verify the results obtained when using only the robot's odometry for the localization.

3. **Test with the robot in motion and the Pozyx system on:** Having already concluded that the various parts of the filter are working properly, the idea of this test is to verify how the EKF behaves when it performs the prediction and correction phase. This way, it is possible to later compare the robot localization results considering the fusion of odometry and Pozyx data with the results obtained considering only the robot's odometry. It can also be verified if with the fusion of the odometry data with the distance data from the Pozyx system it is possible to obtain an accurate localization of the robot.

To verify the accuracy achieved by the various tests performed, the robot localization results obtained by these tests are compared with the true pose of the robot (Ground-Truth), a pose that is determined directly by the SimTwo simulator and sent to the remote computer for later comparison.

5.2.4.2 Real Robot Localization Tests

For the real environment all the systems used, i.e. odometry, Pozyx and ArUco, have already been considered. The different tests performed allowed to test the piecewise filter and to verify the results obtained when using several different combinations of inputs to the EKF. Consequently, the following tests were performed:

1. **Test with the robot stopped at the initial pose and with only the Pozyx system on:** This test is the same performed in the simulator and aims to verify if the filter was well designed, initializing it with a different position from the real one, and then verifying if it converges to the correct one. It also verifies if the real Pozyx system is working well and if the standard deviation used for the noise fits the system.

2. **Test with the robot stopped in the initial pose and with only the ArUco system on (considering that the distance and ϕ angle of the markers relative to the robot is obtained):**

This test is similar to the previous one, however in this case the aim is to verify if the ArUco system is working properly and if the standard deviations used for noise of the distance and ϕ angle measurements are adequate.

3. **Test with the robot in motion and with the Pozyx and ArUco systems off:** As in the same test performed in the simulator, the objective is to test the EKF prediction phase, being able to verify the results obtained by using only the robot's odometry to localize it.

4. **Test with the robot in motion, with the Pozyx system on and with the ArUco system off:** The idea of this test is to verify the results obtained considering odometry data from the robot and the distance data obtained by the Pozyx system. In this case, the EKF performs the correction phase whenever new measurements are obtained from the Pozyx system.

5. **Test with the robot in motion, with the ArUco system on (considering only the ϕ angle measurements of the markers relative to the robot) and with the Pozyx system off:** The purpose of this test is to verify the results obtained considering the robot's odometry data and the data related to the ϕ angle measurements of the ArUco markers in relation to the robot's referential frame. In this case the EKF performs the correction phase whenever new ϕ angle measurements are obtained from the ArUco system.

6. **Test with the robot in motion, with the ArUco system on (considering the ϕ angle and distance measurements of the markers relative to the robot) and with the Pozyx system off:** The aim of this test is to verify the results obtained considering the robot's odometry data and the data related to the ϕ angle and distance measurements of the ArUco markers in relation to the robot's referential frame. In this case the EKF performs the correction phase whenever new ϕ angle and distance measurements are obtained from the ArUco system.

7. **Test with the robot in motion, with the ArUco system on (considering only the ϕ angle measurements of the markers relative to the robot) and with the Pozyx system on:** This test allows verifying the results obtained when considering the robot odometry data, the ArUco marker ϕ angle measurements relative to the robot referential frame and the distance measurements obtained by the Pozyx system. In this case the EKF performs the correction phase whenever new ϕ angle measurements are obtained from the ArUco system or new distance measurements from the Pozyx system.

8. **Test with the robot in motion, with the ArUco system on (considering the ϕ angle and distance measurements of the markers in relation to the robot) and with the Pozyx system on:** This test allows verifying the results obtained when considering the robot odometry data, the ArUco marker ϕ angle and distance measurements relative to the robot referential frame and the distance measurements obtained by the Pozyx system. In this case the EKF performs the correction phase whenever new ϕ angle and distance measurements are obtained from the ArUco system or new distance measurements from the Pozyx system.

The localization results obtained by the various tests are compared with the developed Ground-Truth system (presented in Section 3.4), in order to verify the accuracy achieved when using the different variants of the developed EKF algorithm. By combining different inputs in the Kalman filter, it is later possible to conclude with which one obtains a more accurate robot localization.

Chapter 6

Results and Discussion

In this chapter, the results obtained in the experiments and tests discussed in the previous chapter will be presented. Thus, the chapter will start by presenting the results concerning the characterization of the Pozyx system, followed by the results concerning the characterization of the different measurements acquired through the ArUco markers. Next, the results of the experiments performed with the developed Ground-Truth system are presented. Finally, the results of each localization test presented in the previous chapter will be demonstrated and discussed. A comparison between them will also be performed.

6.1 Pozyx Characterization Results

As mentioned in Section 5.2.1, several experiments were performed in order to characterize the Pozyx system. These experiments involved determining the standard deviation and offset present in the measurements of the four anchors used. Figures 6.1, 6.2, 6.3 and 6.4 show the histograms resulting from the different experiments performed for each of the anchors, anchors that were distanced from the Pozyx tag by a given distance.

By observing the resulting histograms it is possible to conclude that the Pozyx system measurements follow an approximately Gaussian distribution, which enforces the use of the EKF Gaussian process.

Tables 6.1, 6.2, 6.3 and 6.4 summarize the results obtained, i.e. mean, standard deviation and offset, for each anchor for all of the distances performed. By analyzing these tables it is possible to conclude that the standard deviation is approximately constant, i.e. it belongs to the same range of values. Therefore, it is concluded that the standard deviation, used for the Kalman filter noise variable, is independent of distance and anchor. However, as future work, it could be interesting to check whether over larger areas the standard deviation remains approximately constant. To minimize possible outliers, caused by a larger standard deviation, it is useful to use an outlier detection filter, described in Section 4.5.4. Regarding the offset, it was concluded that it is very variable, not following a concrete pattern. However, in an attempt to correct the offset in the estimation of the Pozyx measurements, an algorithm was developed and presented in Section

4.5.5. The results obtained for the offset in the different Pozyx anchors and distance were used in this algorithm.

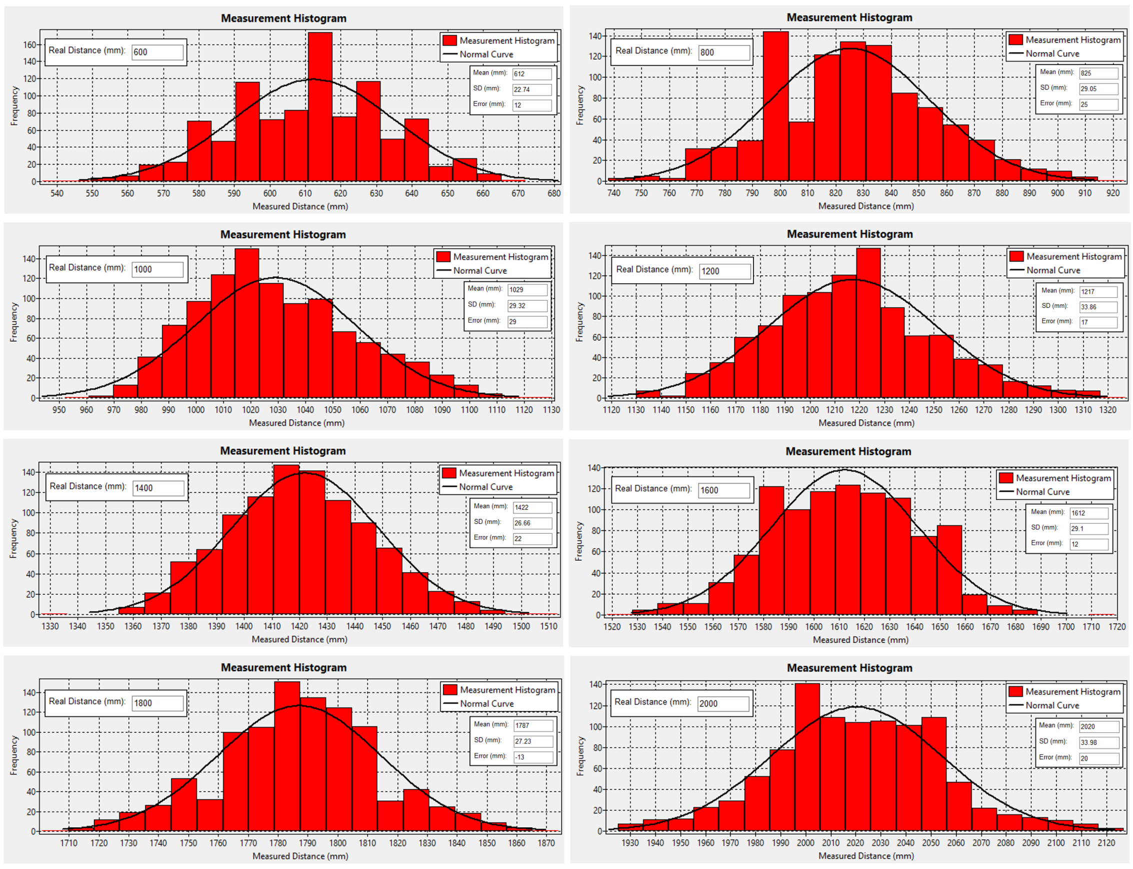


Figure 6.1: Distribution of Pozyx measurements - Anchor 1

Table 6.1: Offset and SD as a function of measurement Pozyx distance - Anchor 1

Real Distance (mm)	Mean (mm)	Offset (mm)	SD (mm)
600	612	12	22.7
800	825	25	29.1
1000	1029	29	29.3
1200	1217	17	33.9
1400	1422	22	26.7
1600	1612	12	29.1
1800	1787	-13	27.2
2000	2020	20	33.9

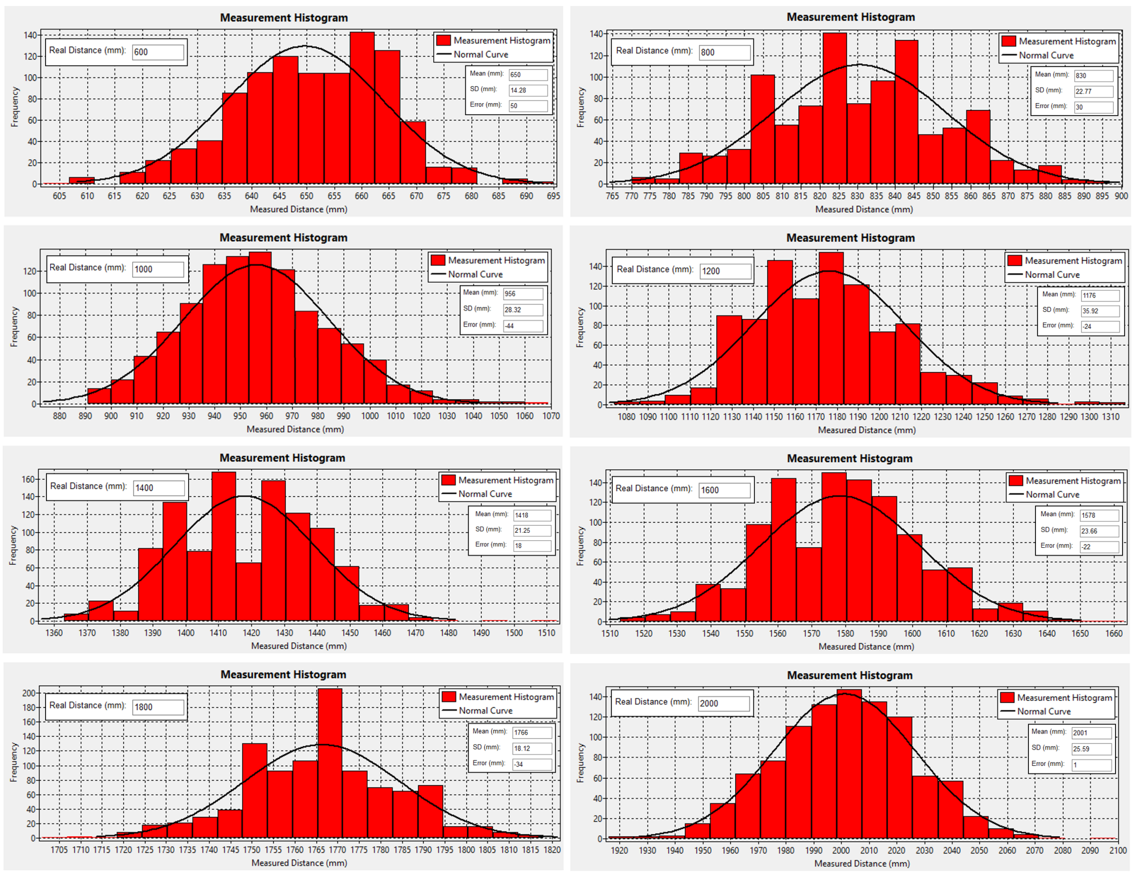


Figure 6.2: Distribution of Pozyx measurements - Anchor 2

Table 6.2: Offset and SD as a function of measurement Pozyx distance - Anchor 2

Real Distance (mm)	**Mean (mm)**	**Offset (mm)**	**SD (mm)**
600	650	50	14.3
800	830	30	22.8
1000	956	-44	28.3
1200	1176	-24	35.9
1400	1418	18	21.3
1600	1578	-22	23.7
1800	1766	-34	18.1
2000	2001	1	25.6

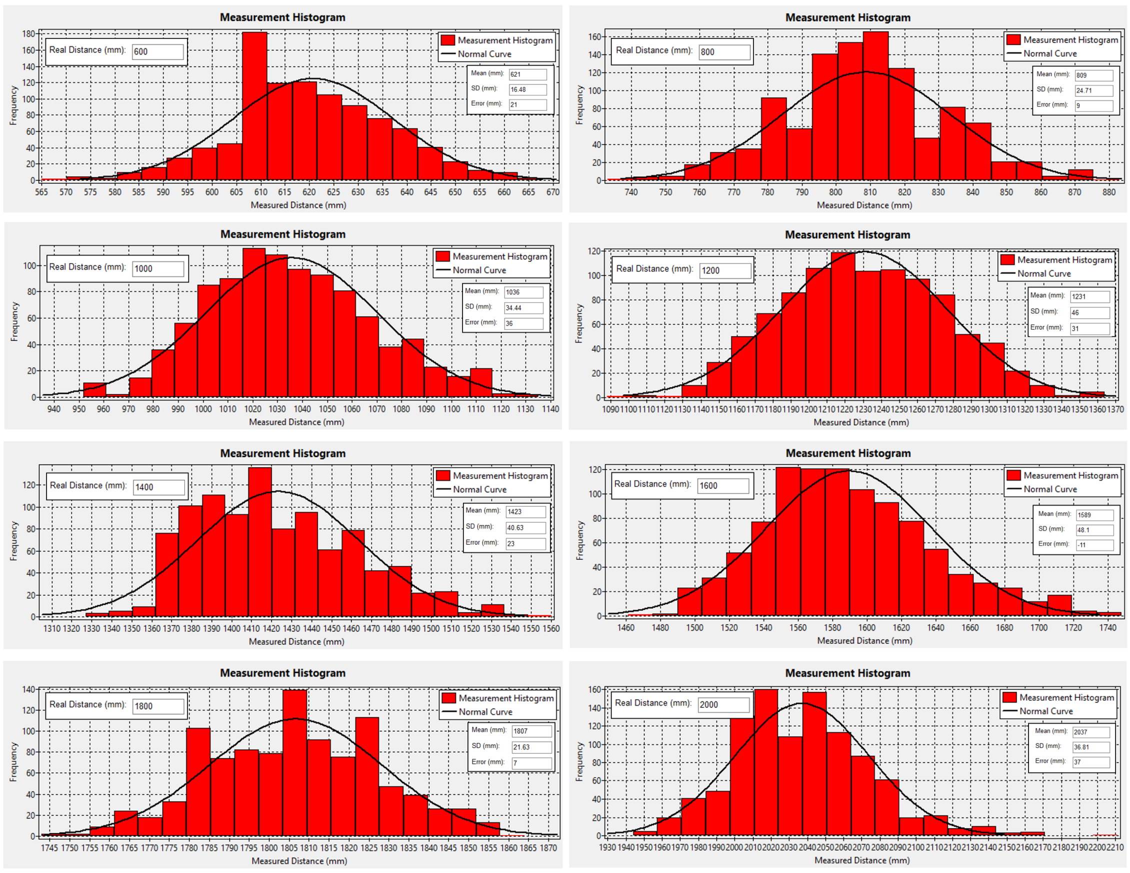


Figure 6.3: Distribution of Pozyx measurements - Anchor 3

Table 6.3: Offset and SD as a function of measurement Pozyx distance - Anchor 3

Real Distance (mm)	**Mean (mm)**	**Offset (mm)**	**SD (mm)**
600	621	21	16.5
800	809	9	24.7
1000	1036	36	34.4
1200	1231	31	46.0
1400	1423	23	40.6
1600	1589	-11	48.1
1800	1807	7	21.6
2000	2037	37	36.8

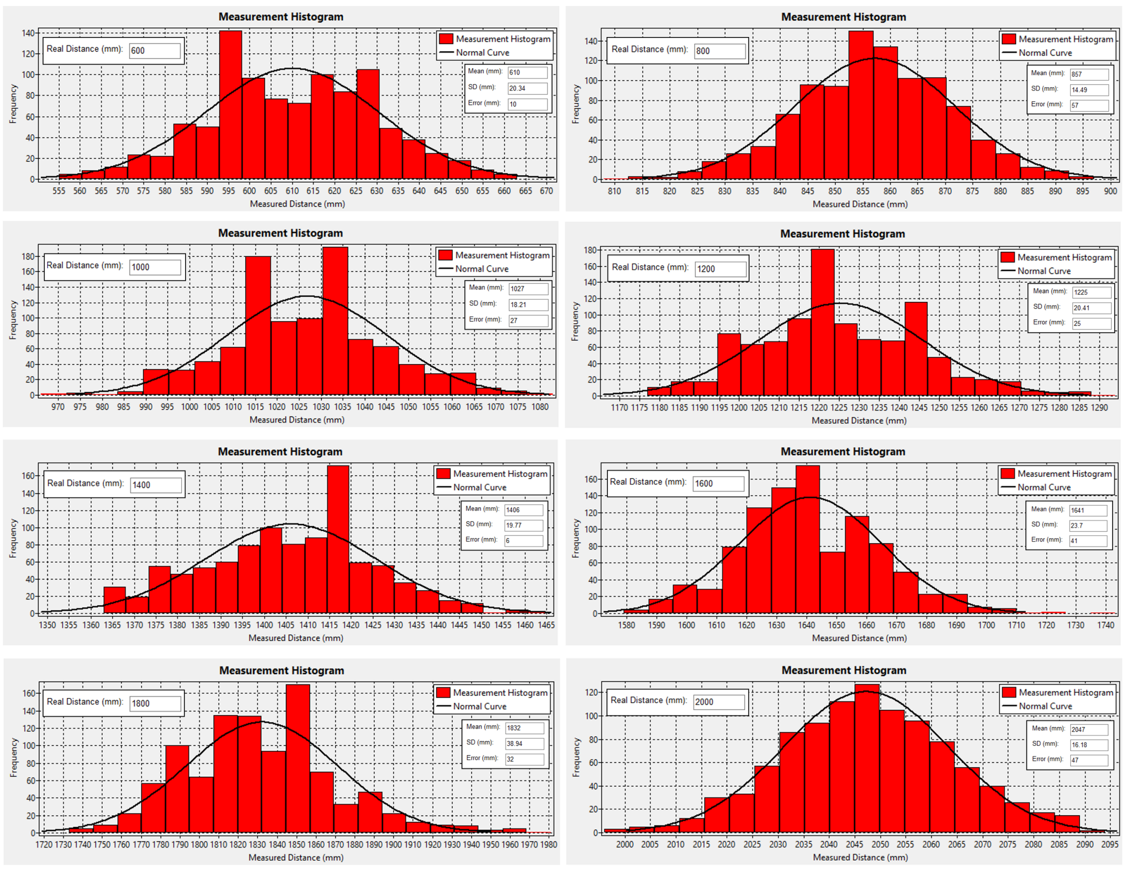


Figure 6.4: Distribution of Pozyx measurements - Anchor 4

Table 6.4: Offset and SD as a function of measurement Pozyx distance - Anchor 4

Real Distance (mm)	Mean (mm)	Offset (mm)	SD (mm)
600	610	10	20.3
800	857	57	14.5
1000	1027	27	18.2
1200	1225	25	20.4
1400	1406	6	19.8
1600	1641	41	23.7
1800	1832	32	38.9
2000	2047	47	16.2

For the characterization of the Pozyx system one more experiment was performed. This experiment was useful to understand how often the Pozyx measurements arrived at the remote computer, to then be able to use this value in the simulation of the Pozyx system in SimTwo. The experiment resulted in an average of 18 ms for the arrival of a measurement relative to any of the anchors. Since SimTwo's control cycle has a periodicity of 40 ms it was assumed that every 40 ms measurements from two anchors were sent to the remote computer.

6.2 ArUco Characterization Results

From the ArUco markers it is possible to extract several useful measurements for estimating the robot's pose. Therefore it was necessary to perform a series of experiments in order to characterize these measurements and verify if they are accurate. These experiments are described in detail in Section 5.2.2. Next, it will be present the results for each of the experiments performed.

ϕ Angle Measurements

Regarding the measurements of the ϕ angle, five different experiments were performed. The results of these experiments are summarized in Table 6.5 and Figure 6.5 presents the histograms obtained for these experiments.

Through the analysis of Table 6.5 and Figure 6.5 it is possible to conclude that the ϕ angle measurements are very low in noise, i.e. low standard deviation, and have a low error as well. Therefore, it can be said that these measurements are accurate and useful to include in the robot's pose estimation algorithm.

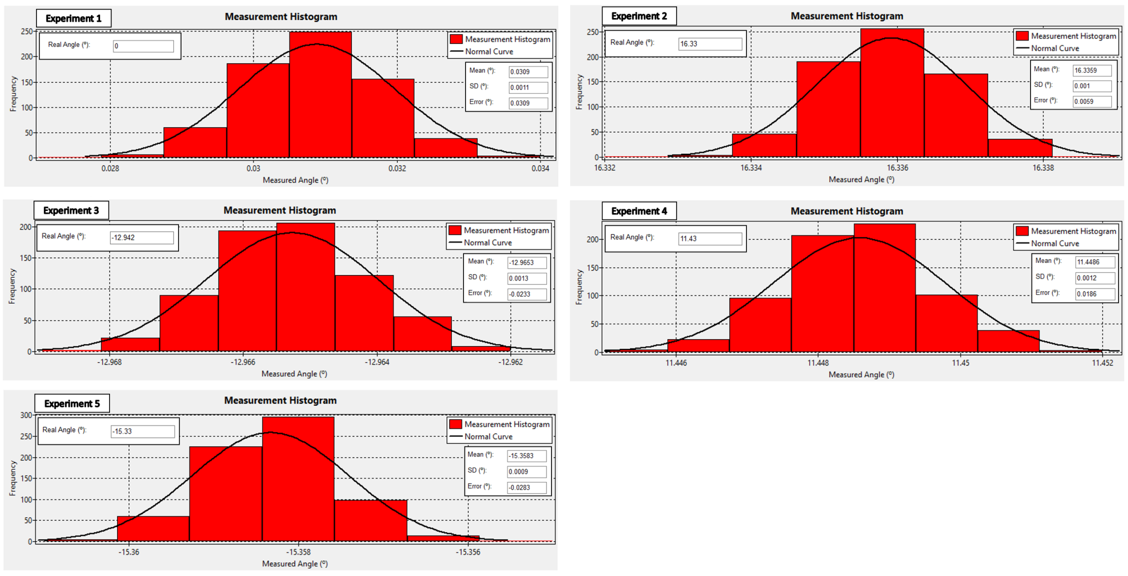


Figure 6.5: Distribution of ϕ angle measurements

Table 6.5: ϕ angle measurement characterization experiments results

Experiment	Real ϕ Angle (°)	Mean (°)	Error (°)	SD (°)
1	0	0.0309	0.0309	1.1e-3
2	16.33	16.3359	0.0059	1.0e-3
3	-12.94	-12.9653	-0.0233	1.3e-3
4	11.43	11.4486	0.0186	1.2e-3
5	-15.33	-15.3583	-0.0283	9.0e-4

Distance Measurements

Concerning the distance measurements of ArUco markers to the origin of the robot's referential frame, the experiments described in Section 5.2.2 were performed. The results obtained in these experiments are summarized in Table 6.6. By analyzing this table it is possible to conclude that just like the ϕ angle, the distance measurements are useful for the robot localization algorithm, since it presents a very low error relative to the real value. Besides the error being low, it should be noted that these measurements are not very noisy, given the low standard deviation.

Table 6.6: Distance measurements characterization experiments results

Experiment	Real Distance (mm)	Mean (mm)	Error (mm)	SD (mm)
1	500	503	3	5.0e-1
2	1000	995	-5	4.6e-1
3	1500	1496	-4	8.1e-1
4	2000	1992	-8	5.2e-1

Pitch Angle Measurements

As mentioned before, the characterization of the *pitch* angle measurements is important, since this angle, along with the previous measurements can be used to calculate the (x_r, y_r) position of the robot. Therefore, the experiments referred to in Section 5.2.2 were performed. Figure 6.6 presents the histograms resulting from the experiments. The Table 6.7 summarizes the results obtained for the error and standard deviation of the measurements, for each of the experiments performed.

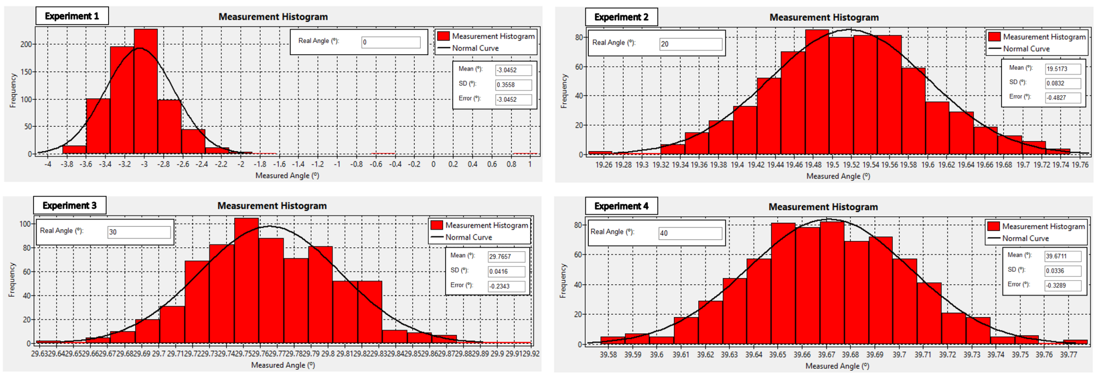


Figure 6.6: Distribution of *pitch* angle measurements

Table 6.7: *Pitch* angle measurement characterization experiments results

Experiment	Real pitch angle (°)	Mean (°)	Error (°)	SD (°)
1	0	-3.0452	-3.0452	3.6e-1
2	20	19.5173	-0.4827	8.3e-2
3	30	29.7657	-0.2343	4.2e-2
4	40	39.6711	-0.3289	3.4e-2

Through the analysis of Figure 6.6 and Table 6.7 it is possible to conclude that the *pitch* angle measurements present more error and more noise, i.e. higher standard deviation, than the ϕ angle measurements. The error is greater when the robot and the ArUco marker are facing each other, that is, when they have a *pitch* angle of 0°. Although these results were verified, the robot's position measurements (x_r, y_r) were still characterized, as described in Section 4.4.

Robot Position Measurements

Concerning the experiments of the (x_r, y_r) position measurements of the robot, the robot was placed in several positions as described in Section 5.2.2. The purpose of these experiments is to verify if the error in obtaining the position is high or low, to then decide if these are good measurements to insert in the developed localization algorithm. In order to synthesize all the information resulting from these experiments, the results obtained are summarized in Table 6.8. Note that the number of the experiment indicated in the table refers to the position indicated with the same number in Figure 5.4.

Table 6.8: Robot position (x_r, y_r) measurements characterization experiments results

Experiment	**Real Position (m)**	**Visible Markers ID**	**Mean (m)**	**Error (cm)**
1	(0.4, 0.5)	12 and 13	ID 12: (0.433, 0.501) ID 13: (0.190, 0.523)	ID 12: (3.3, 0.1) ID 13: (-21, 2.3)
2	(0.4, 1.5)	15 and 16	ID 15: (0.419, 1.419) ID 16: (0.424, 1.618)	ID 15: (1.9, -8.1) ID 16: (-2.4, 11.8)
3	(0.8, 1.5)	18 and 19	ID 18: (0.835, 1.558) ID 19: (0.708, 1.443)	ID 18: (3.5, 5.8) ID 19: (-9.2, -5.7)
4	(1.6, 0.5)	15 and 16	ID 15: (1.323, 0.504) ID 16: (1.589, 0.514)	ID 15: (-27.7, 0.4) ID 16: (-1.1, 1.4)

After analyzing the data presented in Table 6.8 it is possible to conclude that the experiments performed resulted in quite high position errors, reaching a maximum error of 27.7 cm. Due to this, it was decided not to use these measurements in the localization algorithm developed, since this would worsen the results. For the calculation of x_r and y_r several measures are used, namely ϕ angle, *pitch* angle and distance from ArUco markers, measures that contain some error, especially the *pitch* angle. Therefore, the high error in the position measurements can be justified by the fact that it includes several measurements with error, resulting in an accumulation of errors.

In addition to all the experiments mentioned above, an additional experiment was performed. This experiment was performed with the robot in motion and allowed to conclude that when the robot enters in rotation to change direction, the measurements estimated by the Kalman Filter and the measurements that arrived to the remote computer, of the ϕ angle become distant. This no longer happens when the robot moves in a straight line, where both measurements are close to each other. This behavior can be verified in Figure 6.7. The ϕ angle measurements resulting from the ArUco markers when the robot is in rotation can be considered outliers, not being considered for the estimation of the robot's pose. Therefore, an outlier filter was used, the same as the one used with the Pozyx system and explained in Section 4.5.4.

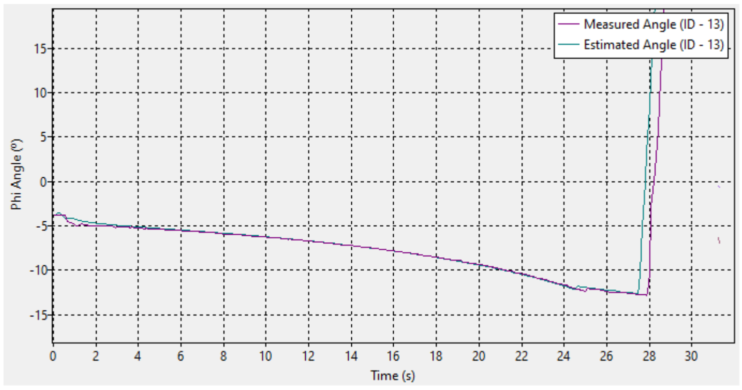


Figure 6.7: Comparison of ϕ estimation and ArUco ϕ measurements

6.3 Ground-Truth Characterization Results

In order to verify if the Ground-Truth system developed acquires precise information of the robot's pose ((x_r, y_r, θ_r)), the set of experiments described in Section 5.2.2 was performed. Through this system it will be possible to perform a comparison with the results obtained in the different variants of the localization algorithm based on EKF developed, concluding their accuracy. Through the experiments performed it is then possible to conclude whether the system is accurate enough to be used as the Ground-Truth of the system.

Therefore, the results obtained in the experiments performed are summarized in Table 6.9. Note that the experiment number in the table coincides with the robot's pose number in Figure 5.5. Also note that the x and y values in the Real Pose and Mean columns are in meters, while the Error and SD columns are in millimeters. The θ values are always indicated in degrees. Through the analysis of the Table 6.9 it is possible to conclude that the errors obtained are low. Thus, it is concluded that this Ground-Truth system developed is accurate. It should be noted that the Ground-Truth system also presents some noise in the measurements, which is usual in measurements with sensors, in this case the camera used.

Table 6.9: Ground-Truth x_r, y_r, θ_r measurements experiments results

Experiment	**Real Pose**	**Mean**	**Error**	**SD**
1	(1, 1, 90)	(0.998, 0.998, 89.92)	(-2, -2, -0.08)	(4.7e-1, 4.6e-1, 5.6e-2)
2	(0.8, 0.5, 180)	(0.8, 0.498, 179.72)	(0, -2, -0.28)	(1.7e-1, 7.8e-1, 4.4e-2)
3	(0.8, 1.5, 0)	(0.796, 1.495, 0.004)	(-4, -5, -0.004)	(3.2e-1, 6.4e-1, 4.4e-2)
4	(1.2, 0.5, 0)	(1.204, 0.498, 0.02)	(4, -2, -0.02)	(1.1e-1, 4.9e-1, 5.4e-2)

6.4 Localization Tests Results

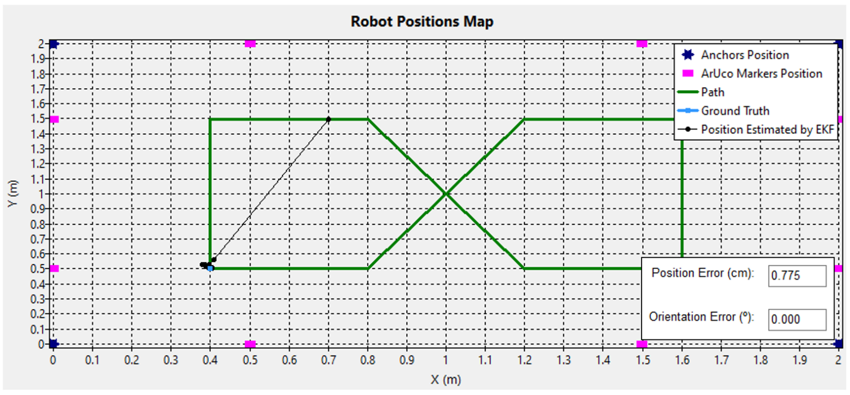
This is the section where the results obtained in the localization tests are demonstrated and discussed. For these tests it was important to tune the EKF in order to achieve greater precision. Changing the EKF parameters affects the convergence of the algorithm, as well as the smoothness of the resulting trajectory estimate. The parameters R and Q were adjusted so that the estimation results were neither sensitive to noise nor too slow. Thus, the adopted implementation relates the dynamic noise Q with the measurements noise R, where the latter is divided into three, mentioned in Section 4.5.2. R_{Pozyx}, which is relative to Pozyx distance measurements, $R_{ArUco \to \phi}$, relative to ϕ angle measurements through ArUco markers, and $R_{ArUco \to distance, \phi}$, referring to combined distance and ϕ angle measurements through ArUco markers. R_{Pozyx}, $R_{ArUco \to \phi}$ and $R_{ArUco \to distance, \phi}$ are determined from the noise characterization previously performed, using an average of the standard deviation values obtained. The Q was determined from trial error tests. After some tests, Q was set to $0.001 \cdot R_{Pozyx}$ in case of using only the Pozyx system and $0.001 \cdot R_{ArUco \to \phi}$ in the case of also using the measurements of the ArUco markers. Concerning the initial state estimates, P_0 is a 3×3 identity matrix.

6.4.1 Simulator Localization Tests Results

In this subsection the results obtained from the tests performed on the SimTwo simulator are presented. Note that the order in which the results are presented is the same order in which the tests were described in Section 5.2.4.1.

Test 1 - Test with the robot stopped at the initial pose and with the Pozyx system on

Figure 6.8 demonstrates the result obtained in the first test performed on the SimTwo simulator. In this test the EKF was initialized with a pose of $(x_r, y_r, \theta_r) = (0.7, 1.5, 90°)$ and the simulator sends to the remote computer the simulated distances relative to the four anchors of the Pozyx system. As can be seen in Figure 6.8 the Kalman Filter converges to the correct estimated pose of the robot, that is $(x_r, y_r, \theta_r) = (0.4, 0.5, 90°)$ with an error of 7.75 mm, which is quite low. Therefore, it is concluded that the standard deviation used for the Pozyx system noise is adequate. Note also that in this case, since the Pozyx system does not provide information about the orientation of the robot, there is no change in the orientation estimation.



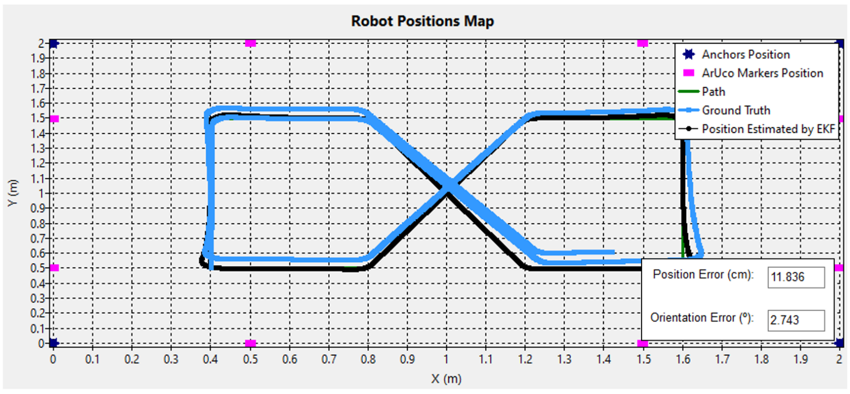
Position Error (cm): 0.775

Orientation Error (°): 0.000

Figure 6.8: SimTwo localization test 1 result

Test 2 - Test with the robot in motion and with the Pozyx system turned off

Figure 6.9 shows the results obtained in the second test performed on the SimTwo simulator. In this test only information from the robot's odometry is considered in the localization algorithm. As expected, and as it is possible to verify by analyzing Figure 6.9, the localization of the robot is clearly affected by wheel slippage on curves and for this reason the error increases as the robot travels longer distances. A position error of 11.84 cm is achieved, and this error would increase more and more each time the robot completes more laps. For this reason it is possible to conclude that using only the robot's odometry information is not a good choice for determining the robot's pose.



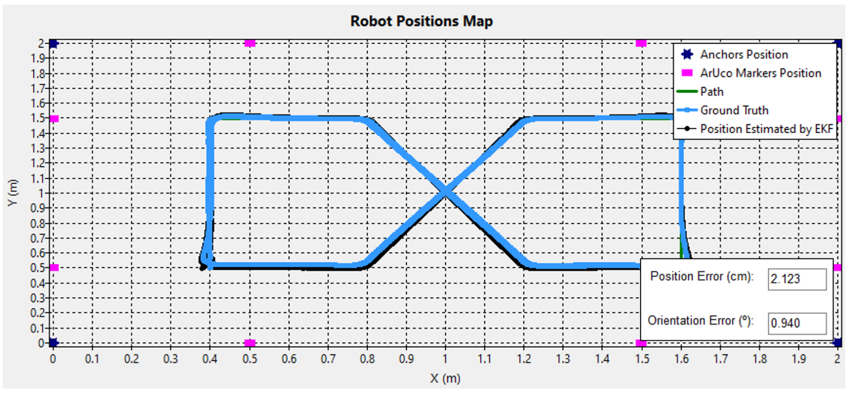
Position Error (cm): 11.836

Orientation Error (°): 2.743

Figure 6.9: SimTwo localization test 2 result

Test 3 - Test with the robot in motion and the Pozyx system on

The last test performed on the simulator considers the fusion of the robot odometry information and the distance measurements from the simulated Pozyx system. The result of this test is illustrated in Figure 6.10. A position error of 2.1 cm and an orientation error of 0.94°, relative to the real pose of the robot is achieved, which are low errors compared to the error obtained in the previous test. Clearly the localization of the robot has improved significantly compared to the previous test, where only odometry was used. Therefore, it can be concluded that using distance to anchor measurements are a good option for improving the robot's pose estimation.



Position Error (cm): 2.123

Orientation Error (º): 0.940

Figure 6.10: SimTwo localization test 3 result

6.4.2 Real Robot Localization Tests Results

In this subsection the results obtained from the tests performed with the real robot are presented. Note that the order in which the results are presented is the same order in which the tests were described in Section 5.2.4.2. Note also, that the first, third, and fourth tests are the same as those performed on the simulator, but were repeated on the real robot to observe and compare the results with the other tests performed.

Test 1 - Test with the robot stopped at the initial pose and with only the Pozyx system on

Figure 6.11 demonstrates the result obtained in the first test. In this test, such as in the simulator, the EKF was initialized with a pose of $(x_r, y_r, \theta_r) = (0.7, 1.5, 90°)$ and the Pozyx system provides distance measurements relative to each of the four anchors placed in the environment. As can be seen in Figure 6.11 the Kalman Filter converges to the correct estimated pose of the robot, that is $(x_r, y_r, \theta_r) = (0.4, 0.5, 90°)$ with an error of 4.7 mm, which is a fairly acceptable error. Therefore, it is concluded that the standard deviation used for the noise of the Pozyx system measurements is adequate. The Pozyx system does not provide information about the orientation of the robot, so there is no change in the orientation estimation.

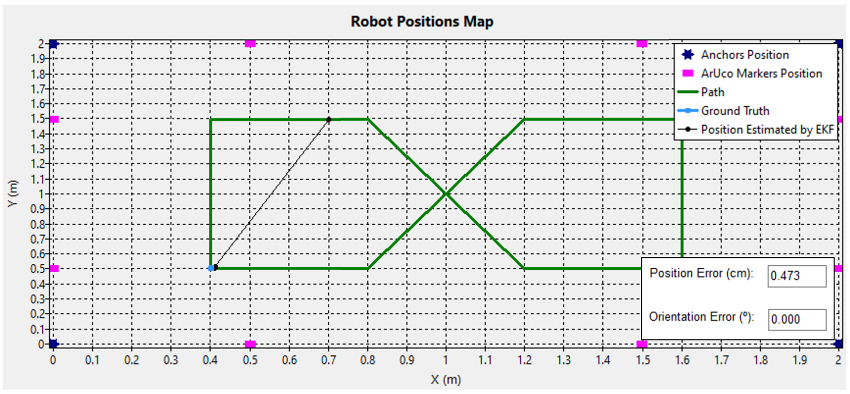


Figure 6.11: Real robot localization test 1 result

Test 2 - Test with the robot stopped in the initial pose and with only the ArUco system on, considering that the distance and ϕ angle of the markers relative to the robot is obtained

This test is similar to the previous one, except that in this case, the ArUco system can already provide information about the orientation and position of the robot, unlike the Pozyx system that only provides information regarding the robot's position. The initial pose of the EKF is $(x_r, y_r, \theta_r) = (0.7, 1.5, 90°)$, just like in the previous test. Figure 6.12 illustrates the results obtained in this test. As can be seen in the figure the robot pose estimated by the EKF algorithm converges to the correct robot pose, $(x_r, y_r, \theta_r) = (0.4, 0.5, 90°)$, with a position error of 8.8 mm and an orientation error of 0.27°, which are acceptable errors. As such, it can be concluded that the standard deviations used for the noise of both measurements of the ArUco markers are adequate.

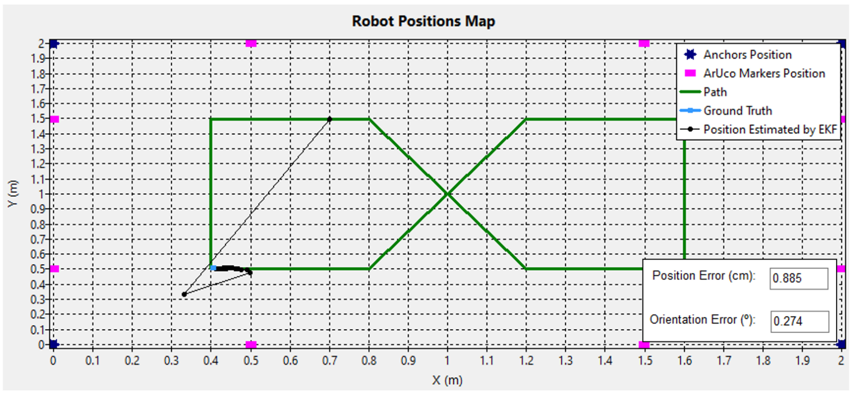


Figure 6.12: Real robot localization test 2 result

Test 3 - Test with the robot in motion and with the Pozyx and ArUco systems off

This test is a repetition of one of the tests performed in the simulator, where only the robot's odometry information is used in the robot's pose estimation algorithm. Figure 6.13 shows the results obtained in this test. Once again, as expected, the localization of the robot is clearly affected by wheel slippage on curves and for this reason the error increases as the robot travels longer distances. A position error of 10.38 cm is achieved, and this error would increase more and more each time the robot completes more and more laps. For this reason it is possible to conclude that using only the robot's odometry information is not a good choice for determining the robot's pose, as had already been concluded in the simulator test.

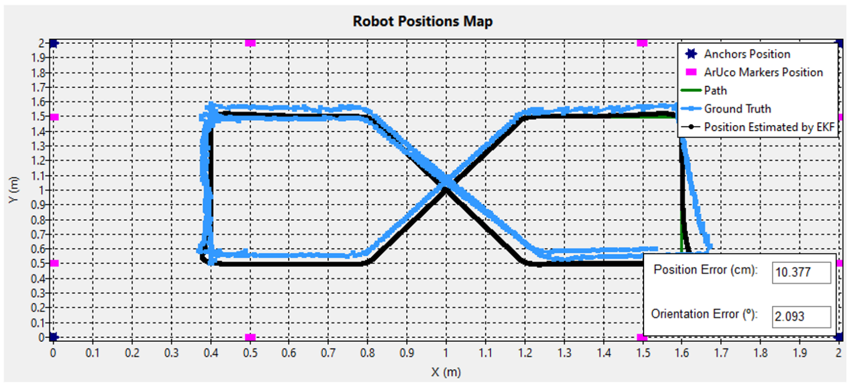


Figure 6.13: Real robot localization test 3 result

Test 4 - Test with the robot in motion, with the Pozyx system on and with the ArUco system off

In this test, only information from the Pozyx system's distance measurements and information from the robot's odometry are used in the EKF algorithm. The result of this test is demonstrated in Figure 6.14. A position error of 1.6 cm and an orientation error of 1.7°, relative to the Ground-Truth system, is achieved. The localization of the robot has improved significantly compared to the previous test, where only odometry was used. The next tests will aim to verify whether using another system, in this case the one related to ArUco markers, or a combination of the two will allow further improvement of the accuracy of the robot's pose estimation.

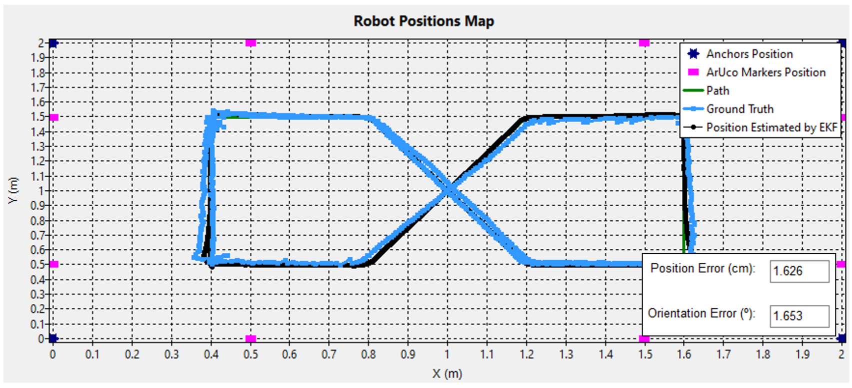


Figure 6.14: Real robot localization test 4 result

Test 5 - Test with the robot in motion, with the ArUco system on (considering only the ϕ angle measurements of the markers relative to the robot) and with the Pozyx system off

In this test, EKF only includes the fusion of the robot odometry measurements and the ϕ angle measurements provided by ArUco markers placed in the environment. The result of this test is illustrated in Figure 6.15. By fusing these measurements, a position error of 6.2 cm and an orientation error of 0.24° was obtained. In comparison with the results of the previous test, in which information from the robot's odometry and the Pozyx system are used, it can be concluded that by using measurements from the ϕ angle it is possible to obtain a higher precision in orientation, that is, a lower error. However, the position accuracy is lower. A fusion of the three information, i.e. ϕ angle, distance provided by the Pozyx system and odometry, would improve the accuracy of the robot's position.

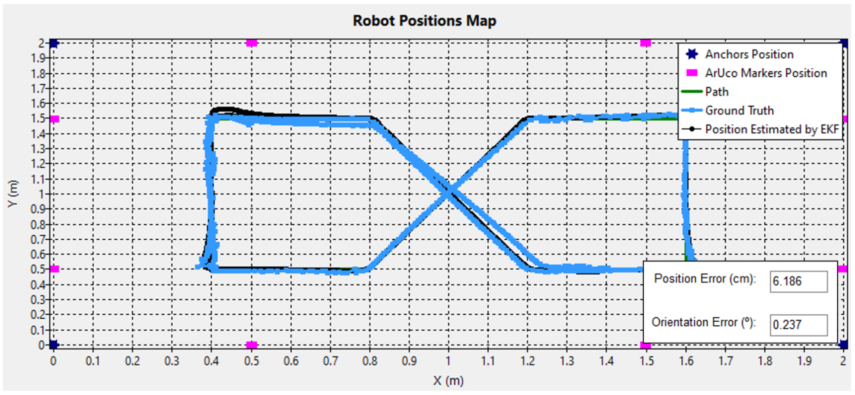


Figure 6.15: Real robot localization test 5 result

Test 6 - Test with the robot in motion, with the ArUco system on (considering the ϕ angle and distance measurements of the markers relative to the robot) and with the Pozyx system off

For this test, EKF includes the fusion of the robot's odometry measurements, and measurements of the ϕ angle and distance to the markers, which are provided by the ArUco markers. The result of this test is shown in Figure 6.16. Through the fusion of these measurements, it is possible to obtain a position error of 1.7 cm and an orientation error of 0.21°.It can be concluded that the accuracy, both in terms of position and orientation, has improved compared to the results obtained in the two previous tests. It is possible to affirm that the use of the distance and ϕ angle measurements, along with the odometry information, for the estimation of the robot's pose is better than the use of the distance measurements provided by the Pozyx system.

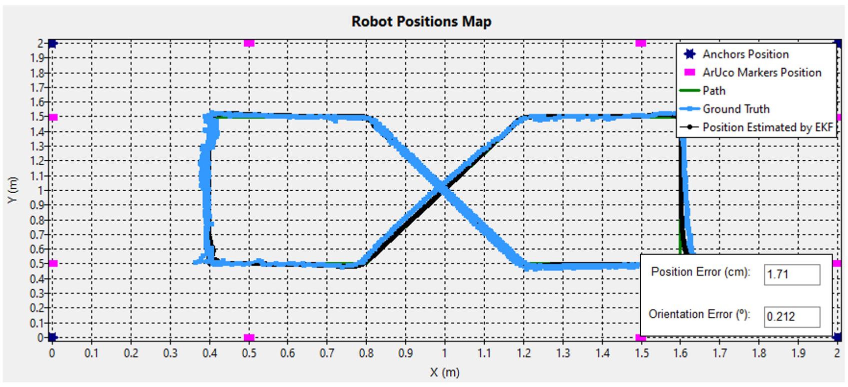


Figure 6.16: Real robot localization test 6 result

Test 7 - Test with the robot in motion, with the ArUco system on (considering only the ϕ angle measurements of the markers relative to the robot) and with the Pozyx system on
Figure 6.17 exhibits the results obtained from this test. This test includes the fusion of the robot's odometry data, ϕ angle measurements provided through ArUco markers and distance measurements relative to the four Pozyx anchors. The robot achieves a position error of 1.9 cm and an orientation error of 0.39°. As expected and as previously speculated, the position error has decreased significantly compared to the test where only ϕ angle and odometry measurements are used. However it should be noted that considering the fusion of both measurements provided through the ArUco markers and information from the robot's odometry the accuracy in the robot's pose is higher, as verified in the previous test.

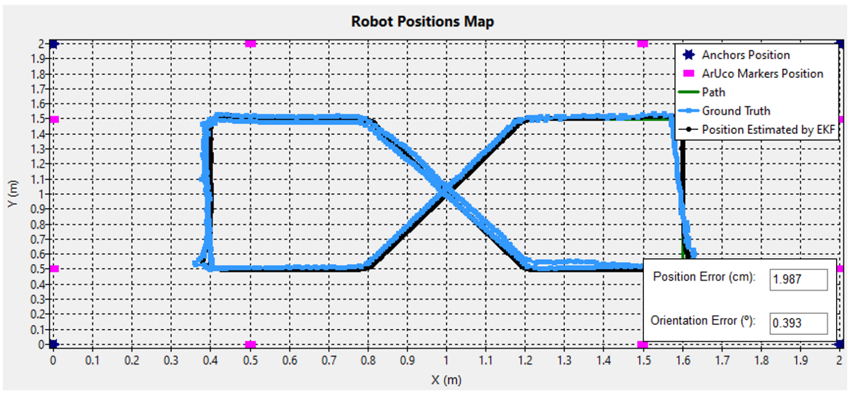


Figure 6.17: Real robot localization test 7 result

Test 8 - Test with the robot in motion, with the ArUco system on (considering the ϕ angle and distance measurements of the markers in relation to the robot) and with the Pozyx system on
Finally, in the last test performed, all measurements are used, that is, odometry, distance to the four Pozyx anchors, and distance and ϕ angle provided by the ArUco markers. In other words, a fusion of all previous measurements is performed. The result of this test is shown in Figure

6.18. A position error of 2.1 cm and an orientation error of 0.75° is achieved. Although these are considered low errors, they are still higher than the errors obtained in the test where only the odometry measurements are fused with the distance and ϕ angle measurements provided by the ArUco markers.

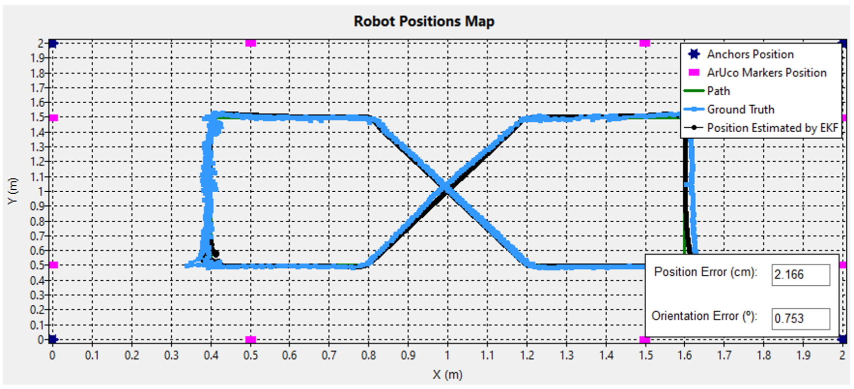


Figure 6.18: Real robot localization test 8 result

Summary of tests where the robot is in motion

The Table 6.10 presents the summary of the obtained results in each of the tests performed with the real robot in motion. The numbers of the tests shown in the table are the same as the ones by which the tests were presented earlier.

Table 6.10: Summary of the obtained results

Test	Position Error (cm)	Orientation Error (°)
3	10.377	2.093
4	1.626	1.653
5	6.186	0.237
6	1.710	0.212
7	1.987	0.393
8	2.166	0.753

By analyzing the Table 6.10 it can be concluded that the combination of input measurements to the Kalman Filter with which the most accurate robot pose is obtained, relative to Ground-Truth, is the combination of the robot's odometry data with the measurements of the ϕ angle and distance extracted from the ArUco markers view, i.e. test 6. This is because these measurements extracted from the ArUco markers have less error and less noise (lower standard deviation), as concluded in previous experiments. Nevertheless, it should also be noted that the results obtained in tests 4, 5, 7 and 8 are also reasonable in comparison with the results obtained when using only the robot's odometry to determine the robot's pose, referring to the test 3.

Chapter 7

Conclusions and Future Work

For this dissertation, the main purpose was to develop a localization system using UWB technology and ArUco markers. This system would be able to estimate the pose of a mobile robot with high accuracy.

To achieve the goal of this work, a localization system based on beacons was developed, using distance and angle measurements relative to these beacons. In this case, the beacons are Pozyx anchors, from which it is possible to extract distance measurements, and ArUco markers, from which it is possible to extract distance and angle measurements. The fusion of the different extracted measurements was achieved by an Extended Kalman Filter.

In order to be able to compare the results with another localization system, an additional localization system was developed, that should be accurate enough to be designated as the Ground-Truth of the system. This additional system was developed by placing an ArUco marker on top of the robot and centered with its referential frame and placing a camera above the environment where the robot is moving. In this way, the camera can detect the marker placed on the robot and determine its pose.

In order to understand the accuracy and noise of the Pozyx and ArUco measurements several initial experiments were performed in different scenarios. These experiments allowed to conclude that the distance and ϕ angle measurements extracted from the ArUco markers were very accurate and not very noisy (low standard deviation). The position measurements (x_r, y_r) of the robot, on the other hand, which are also possible to extract by combining several measurements extracted from the ArUco markers, are not very accurate and have a high error. This is due to the accumulation of errors from all the measurements used for the calculation, especially the $pitch$ angle which, when it should be $0°$, has a larger error. For this reason it was decided not to take into account these (x_r, y_r) measurements in the developed localization algorithm. Regarding the distance measurements of the Pozyx technology it was verified that they do not present a high error, however this error is higher compared with the distance measurements extracted from the ArUco markers, as well as the noise which is also higher. Thus, in the EKF algorithm, this study was important, because it is necessary to quantify the noise of the measurements, through the standard deviation. For the additional system developed, a similar study was performed, from which it was concluded that it is an accurate localization system given the low error achieved. Therefore, this system was used as Ground-Truth.

Through the different localization tests performed, it was possible to conclude that the different tested versions of the algorithm achieved promising results compared to localization results where only the robot's odometry is used. However, it was possible to conclude that the most favorable combination of measurements, i.e., the one that led to the most accurate localization, was the use of distance and ϕ angle measurements extracted from ArUco markers, and data from the robot's odometry. Thus it was possible to conclude that using the Extended Kalman Filter, with careful tuning, is a successful algorithm for estimating the robot's pose, along with the measurements used in it. Overall, the results obtained through the use of these measurements showed a position accuracy in the order of centimeters, unlike other existing technologies, such as GPS and Wi-Fi, whose error is in the order of meters, as verified in the literature review.

It is worth mentioning that a paper was written about the work developed in this dissertation. The paper, entitled Sensor Fusion for Mobile Robot Localization using Extended Kalman Filter, UWB ToF and ArUco Markers, was accepted by the International Conference on Optimization, Learning Algorithms and Applications (OL2A 2021).

As future work, it would be interesting to extend the studies of the different measurements in larger spaces, in order to verify if the accuracy remains similar. Regarding Pozyx technology, it would be interesting to perform studies on the possible restrictions of space, i.e. in NLOS environments. Concerning the visualization of the ArUco markers, an intensive study of the influence of the luminosity of the environment would help to understand its influence on the precision of the acquisition of the measurements. The work could be expanded to use a larger robot, in a larger space, and in an industrial environment. Finally, it would also be interesting to include measurements from other sensors in the Kalman Filter algorithm. This way, it would be possible to conclude if it is possible to improve the accuracy of the algorithm by using data from other sensors.

In conclusion, through the work developed, it was possible to show that the use of Pozyx technology and ArUco markers scattered around the environment, along with the measurements extracted from them, are an effective tool to improve the accuracy of a robot's localization, compared to using only its odometry.

References

[1] Wilson Sakperea, Michael Adeyeye-Oshinb, and Nhlanhla B.W. Mlitwa. A state-of-the-art survey of indoor positioning and navigation systems and technologies. *South African Computer Journal*, 2017.

[2] Mai A. Al-Ammarz, Suheer Alhadhrami, AbdulMalik Al-Salman, Abdulrahman Alarifiy, Hend S. Al-Khalifa, Ahmad Alnafessahy, and Mansour Alsaleh. Comparative survey of indoor positioning technologies, techniques, and algorithms. *International Conference on Cyberworlds*, 2014.

[3] Pozyx. How does ultra-wideband work? Available in `https://www.pozyx.io/technology/how-does-uwb-work`.

[4] P. Dabove, V. D. Pietra, M. Piras, A. A. Jabbar, and Syed Ali Kazim. Indoor positioning using ultra-wide band (uwb) technologies: Positioning accuracies and sensors' performances. *IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2018.

[5] Mark Fiala. Artag, a fiducial marker system using digital techniques. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[6] Andrej Babineca, Ladislav Jurišicaa, Peter Hubinskýa, and František Duchon. Visual localization of mobile robot using artificial markers. *Procedia Engineering 96*, 2014.

[7] Camera calibration, 2020. Available in `https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html`.

[8] Mirella Santos Pessoa de Melo, José Gomes da Silva Neto, Pedro Jorge Lima da Silva, João Marcelo Xavier Natario Teixeira, and Veronica Teichrieb. Analysis and comparison of robotics 3d simulators. *21st Symposium on Virtual and Augmented Reality (SVR)*, 2019.

[9] Coppeliasim, 2019. Available in `https://www.coppeliarobotics.com`.

[10] Webots, 2020. Available in `https://cyberbotics.com/`.

[11] Paulo Costa, José Gonçalves, José Lima, and Paulo Malheiros. Simtwo realistic simulator: A tool for the development and validation of robot software. *Theory and Applications of Mathematics Computer Science*, 2011.

[12] Raspberry pi 3 model b, 2021. Available in `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/`.

[13] Arduino uno, 2021. Available in `https://en.wikipedia.org/wiki/Arduino_Uno`.

[14] Pozyx. Available in `https://www.pozyx.io/`.

[15] Abdulrahman Alarifi, Mansour Alsaleh AbdulMalik Al-Salman, Ahmad Alnafessah, Suheer Al-Hadhrami, Mai A. Al-Ammar, and Hend S. Al-Khalifa. Ultra wideband indoor positioning technologies: Analysis and recent advances. *Sensors*, 2016.

[16] Shoudong Huang and Gamini Dissanayake. Robot localization: An introduction. *American Cancer Society*, 2016.

[17] José Lima and Paulo Costa. Ultra-wideband time of flight based localization system and odometry fusion for a scanning 3 dof magnetic field autonomous robot. *Springer International Publishing, Third Iberian Robotics Conference*, 2017.

[18] Luka Teslic, Igor Škrjanc, and Gregor Klancar. Ekf-based localization of a wheeled mobile robot in structured environments. *Springer Science+Business Media B.V*, June 2010.

[19] Ling Chen, Huosheng Hu, and Klaus McDonald-Maier. Ekf based mobile robot localization. *Third International Conference on Emerging Security Technologies*, 2012.

[20] Andre M. Santana, Anderson A. S. Sousa, Ricardo S. Britto, Pablo J. Alsina, and Adelardo A. D. Medeiros. Localization of a mobile robot based in odometry and natural landmarks using extended kalman filter. 2008.

[21] Yuchuan Liu and Yixu Song. A robust method of fusing ultra-wideband range measurements with odometry for wheeled robot state estimation in indoor environment. 2018.

[22] Jingxiang Zheng, Shusheng Bi, Bo Cao, and Dongsheng Yang. Visual localization of inspection robot using extended kalman filter and aruco markers. *IEEE International Conference on Robotics and Biomimetics*, 12 2018.

[23] Alvin Marquez, Brinda Tank, Sunil Kumar Meghani, Sabbir Ahmed, and Kemal Tepe. Accurate uwb and imu based indoor localization for autonomous robots. *IEEE 30th Canadian Conference on Electrical and Computer Engineering*, 2017.

[24] Nasir Saeed, Haewoon Nam, Tareq Y. Al-Naffouri, and Mohamed-S. Alouini. A state-of-the-art survey on multidimensional scaling-based localization techniques. *IEEE Communications Surveys and Tutorials*, 2019.

[25] Fazeelat Mazhar, Muhammad Gufran Khan, and Benny Sallberg. Precise indoor positioning using uwb: A review of methods, algorithms and implementations. *Springer Science+Business Media*, 2017.

[26] Teresa Conceição. Robot localization in an agricultural environment. Technical report, Faculdade de Engenharia da Universidade do Porto, January 2018.

[27] Faheem Zafari, Athanasios Gkelias, and Kin K. Leung. A survey of indoor localization systems and technologies. *IEEE Communications Surveys and Tutorials*, 2019.

[28] Pozyx. Ultra-wideband and obstacles. Available in `https://www.pozyx.io/technology/uwb-and-obstacles`.

[29] Ubisense. Available in `https://ubisense.com/`.

[30] Decawave. Available in `https://www.decawave.com/`.

[31] Uniset. Available in `https://www.unisetcompany.com/`.

[32] Zebra. Available in https://www.zebra.com/.

[33] Qin Shi, Sihao Zhao, Xiaowei Cui, Mingquan Lu, and Mengdi Jia. Anchor self-localization algorithm based on uwb ranging and inertial measurements. *TSINGHUA SCIENCE AND TECHNOLOGY*, 2019.

[34] Dibyendu Ghosh, Vinayak Honkote, and Karthik Narayanan. Dynamic adaption of noise covariance for accurate indoor localization of mobile robots in non-line-of-sight environments. *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 09 2020.

[35] Francisco J. Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicera. Tracking fiducial markers with discriminative correlation filters. *Image and Vision Computing*, 2020.

[36] Atle Aalerud, Joacim Dybedal, and Geir Hovlando. Automatic calibration of an industrial rgb-d camera network using retroreflective fiducial markers. *Sensors*, 2019.

[37] Aruco: a minimal library for augmented reality applications based on opencv, 2020. Available in https://www.uco.es/investiga/grupos/ava/node/26.

[38] Diogo Oliveira, Luís Simões, Mariana Martins, and Miguel Paulino. Ekf-slam using visual markers for iter's remote handling transport casks. *Instituto Superior Técnico, Autonomous Systems*, 2019.

[39] Tiago Almeida, Vitor Santos, Bernardo Lourenço, and Pedro Fonseca. Detection of data matrix encoded landmarks in unstructured environments using deep learning. *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2020.

[40] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE transactions on systems, man and cybernetics*, 2007.

[41] Fares Alkhawaja, Mohammad Jaradat, and Lotfi Romdhane. Techniques of indoor positioning systems (ips): A survey. *Advances in Science and Engineering Technology International Conferences (ASET)*, 2019.

[42] Yanying Gu, Anthony Lo, and Ignas Niemegeers. A survey of indoor positioning systems for wireless personal networks. *IEEE Communications Surveys and Tutorials*, 2009.

[43] Timothy D. Barfoot. State estimation for robotics. 2020.

[44] Sebastian Thrun, Dieter Fox, and Wolfram Burgard. *Probabilistic robotics*. Open University Press, volume 45 edition, 2002.

[45] Maria Isabel Ribeiro. Kalman and extended kalman filters: Concept, derivation and properties. Technical report, Instituto Superior Técnico, February 2004.

[46] Ehab I. Al Khatib, Mohammad A. Jaradat, Mamoun Abdel-Hafez, and Milad Roigari. Multiple sensor fusion for mobile robot localization and navigation using the extended kalman filter. 2015.

[47] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. 1997.

[48] Antonio Giannitrapani, Nicola Ceccarelli, Fabrizio Scortecci, and Andrea Garulli. Comparison of ekf and ukf for spacecraft localization via angle measurements. 2009.

[49] Luigi D'Alfonso, Walter Lucia, Pietro Muraca, and Paolo Pugliese. Mobile robot localization via ekf and ukf: A comparison based on real data. robotics and autonomous systems. 2015.

[50] Kexin Guo, Zhirong Qiu, Cunxiao Miao, Abdul Hanif Bin Zaini, Chun-Lin Chen, Wei Meng, and Lihua Xie. Ultra-wideband-based localization for quadcopter navigation. *Unmanned Systems*, 2016.

[51] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. 1999.

[52] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 2000.

[53] Sebastian Thrun. Particle filters in robotics. 2002.

[54] Antoni Burguera, Yolanda González, and Gabriel Oliver. Mobile robot localization using particle filters and sonar sensors. 2009.

[55] Jeong Woo, Young-Joong Kim, Jeong on Lee, and Myo-Taeg Lim. Localization of mobile robot using particle filter. *SICE-ICASE International Joint Conference*, 2006.

[56] Gazebo, 2014. Robot simulation made easy. Available in http://www.gazebosim.org.

[57] Simtwo - a realistic simulator for robotics, 2020. Available in https://github.com/P33a/SimTwo.

[58] Lazarus, 2021. Available in https://www.lazarus-ide.org/.

[59] Tiziano Fiorenzani. How do drones work. GitHub repository: https://github.com/tizianofiorenzani/how_do_drones_work.

[60] Héber Miguel Plácido Sobreira. Fiabilidade e robustez da localização de robôs móveis. Technical report, Faculdade de Engenharia da Universidade do Porto, January 2017.

[61] Blaise Omer Yenké, Moussa Aboubakar, Chafiq Titouna, Ado Adamou Abba Ari, and Abdelhak Mourad Gueroui. Adaptive scheme for outliers detection in wireless sensor networks. *International Journal of Computer Networks and Communications Security*, 2017.