PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

RENAN GUEDES MAIDANA

OUTDOOR LOCALIZATION SYSTEM FOR MOBILE ROBOTS BASED ON RADIO-FREQUENCY SIGNAL STRENGTH

Porto Alegre
2018

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM

OUTDOOR LOCALIZATION SYSTEM FOR MOBILE ROBOTS BASED ON RADIO-FREQUENCY SIGNAL STRENGTH

RENAN GUEDES MAIDANA

Thesis submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fullfillment of the requirements for the degree of Master in Computer Science.

Advisor: Prof. Alexandre de Morais Amory
Co-Advisor: Prof. Aurelio Tergolina Salton

Porto Alegre
2018

Ficha Catalográfica

M217o Maidana, Renan Guedes

Outdoor localization system for mobile robots based on radio-frequency signal strength / Renan Guedes Maidana . – 2018.

136 f.

Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Alexandre de Morais Amory.
Co-orientador: Prof. Dr. Aurelio Tergolina Salton.

1. Robot Localization. 2. Robot Perception. 3. Wireless Sensor Networks. 4. Radio-Frequency. I. Amory, Alexandre de Morais. II. Salton, Aurelio Tergolina. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecário responsável: Marcelo Votto Texeira CRB-10/1974

Renan Guedes Maidana

Outdoor Localization System for Mobile Robots Based on Radio-Frequency Signal Strength

This Dissertation/Thesis has been submitted in partial fulfillment of the requirements for the degree of Doctor/Master of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on March 2, 2018.

COMMITTEE MEMBERS:

Prof. Dr. Felipe Rech Meneguzzi (Pontifícia Universidade Católica do Rio Grande do Sul)

Prof. Dr. Mariana Luderitz Kolberg (Universidade Federal do Rio Grande do Sul)

Prof. Dr. Alexandre de Morais Amory (PPGCC/PUCRS - Advisor)

Prof. Dr. Aurelio Tergolina Salton (PPGEE/PUCRS – Co-Advisor)

To my family: my parents, Rubem and Vera, and my brothers, Rodrigo and Ray. Without your support and comprehension, I could not have come this far. Also to Alessandra, my partner-in-crime, for all the love, support, coffee, for "being my robot", and for never giving up on me.

“The one who insists on never uttering an error must remain silent.”
(Werner Heisenberg)

ACKNOWLEDGMENTS

To my co-advisor, Aurelio Tergolina Salton, for all the patience, for always being available to help me, for putting up with my "e-mail dissertations", and for helping me to keep my head off the clouds. I believe that even if the world was coming to an end, you would still find some time to see me between his lectures. Rarely we find someone with such level of instruction and, at the same time, still humble and accessible in his knowledge. You are the model of researcher and educator I strive to someday become.

To my advisor, Alexandre de Morais Amory, for the friendship, the availability and for always extending me opportunities to deepen my knowledge in Mobile Robotics (with the so-called "perfect storms" you put me in). Thank you for betting on me, for encouraging me to seek an interdisciplinary education, and for introducing me to Mobile Robotics, the area I fell in love with and where I discovered myself as a researcher. We both come from areas different from Mobile Robotics, and it was a pleasure to walk down this path with you.

To my lab colleagues and friends, especially Juarez Monteiro, Roger Granada, Davi Henirque, Marcelo Paravisi and Vitor Jorge, for all the tips and for listening and debating my ideas with me, whether they were good or bad. Thank you for understanding and sharing with me the burden of being a postgraduate student, and for encouraging me to keep going forward in spite of all the hardships.

To my therapist, Eliana, for keeping me sane and for helping me through these perilous times. To all the lecturers of the courses I took, for contributing immensely to my intellectual and professional growth. To the National Council for Scientific and Technological Development CNPq, and the companies Dell and Hewlett-Packard, for the financial support. To the Instituto Brasileiro de Geografia e Estatística for allowing me to use their Continuous Monitoring Reference Stations in the RBMC-IP system. And finally, to all the friends and colleagues that in some way helped me and encouraged me during this Masters course.

OUTDOOR LOCALIZATION SYSTEM FOR MOBILE ROBOTS BASED ON RADIO-FREQUENCY SIGNAL STRENGTH

ABSTRACT

In the field of Mobile Robotics, the localization problem consists on determining a robot's position and orientation in a three-dimensional space through sensor information. The most common solution to this problem is to employ a Global Positioning System receiver, also known as GPS, which reports absolute position in relation to an Earth-centered fixed coordinate system. However, GPS signals are greatly affected by atmospheric conditions and line-of-sight occlusion, sometimes providing very poor position estimates, if any at all. Inspired by these problems, this project proposes a localization system to be used by a robot in an uncontrolled outdoor environment, where GPS measurements are poor or unavailable. As common sensors provide inaccurate position estimates due to environmental factors (e.g. rough terrain), we propose the use of Radio-Frequency receiver-transmitter pairs, in which the Received Signal Strength Indicator is used for estimating the distances between receiver and transmitter, which in turn are used for positioning. This measurement has the advantage of being independent from lighting conditions or the state of the terrain, factors which affect other localization methods such as visual or wheel odometry. A mean positioning error of 0.41 m was achieved by fusing wheel odometry, angular velocity from a gyroscope and the received signal strength, in an Augmented Extended Kalman Filter algorithm, with an improvement of 82.66% relative to the mean error of 2.38 m obtained with a common GPS sensor.

Keywords: Robot Localization, Robot Perception, Wireless Sensor Networks, Radio-Frequency.

SISTEMA DE LOCALIZAÇÃO PARA ROBÔS MÓVEIS EM AMBIENTE EXTERNO BASEADO EM POTÊNCIA DE SINAIS DE RÁDIO-FREQUÊNCIA

RESUMO

Na área da Robótica Móvel, o problema da localização é definido como a determinação da posição e orientação de um robô em um espaço tri-dimensional através de informações de seus sensores. A solução mais comum para esse problema é utilizar um receptor de GPS (do inglês, Global Positioning System), que reporta posição absoluta com relação a um sistema de coordenadas fixo e centralizado na Terra. Porém, o sinal de GPS é muito afetado por condições ambientais e oclusão de linha de visão, por vezes fornecendo estimativas de posição de baixa qualidade, se houverem. Com inspiração nestes problemas, este projeto propõe um sistema de localização para ser usado por um robô terrestre em um ambiente externo não-controlado, onde há indisponibilidade de GPS ou que suas medidas são de baixa qualidade. Tendo em vista que sensores de baixo custo apresentam medições imprecisas devido a fatores ambientais (e.g. terreno acidentado), é proposta a utilização de pares receptor-transmissor de Rádio-Frequência, onde a medida do Indicador de Potência de Sinal Recebido é usada para estimar as distâncias entre receptor e transmissor, que são por sua vez usadas para posicionamento. Essa medida possui a vantagem de ser independente da iluminação do ambiente e do estado do terreno, que afetam outros métodos de localização como Odometria Visual ou por rodas. Um erro médio de posicionamento de 0.41 m foi alcançado através da fusão de odometria por rodas, velocidade angular de um giroscópio e potência de sinal recebido, em um algoritmo de Filtro de Kalman Estendido Aumentado, com uma melhoria de 82.66% referente ao erro médio de 2.38 m obtido com um sensor GPS comum.

Palavras-Chave: Localização Robótica, Percepção Robótica, Redes de Sensores Wireless, Rádio-Frequência.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ACRONYMS

A-EKF – Augmented Extended Kalman Filter

ARIA – Advanced Robot Interface for Applications

CORS – Continuous Operating Reference Stations

D-GPS – Differential GPS

EKF – Extended Kalman Filter

FSPL – Free-Space Path Loss

GPS – Global Positioning System

GPST – GPS Time

HRATC – Humanitarian Robotics and Automation Technology Challenge

IBGE – Instituto Brasileiro de Geografia e Estatística

ICC – Instantaneous Center of Curvature

IMU – Inertial Measurement Unit

KF – Kalman Filter

LDPL – Log-Distance Path Loss

LIPO – Lithium-Polymer

NTRIP – Networked Transport of RTCM via Internet Protocol

P3AT – Pioneer 3-AT

PDF – Probability Density Function

PF – Particle Filter

RBMC – Rede Brasileira de Monitoramento Contínuo

RF – Radio-Frequency

RMSE – Root-Mean-Square Error

ROS – Robot Operating System

RP-SMA – Reverse Polarity SubMiniature A

RSSI – Received Signal Strength Indicator

RTK – Real-Time Kinematics

SSH – Secure Shell

TAI – Temps Atomique International

TDOA – Time-Difference-Of-Arrival

USV – Unmanned Surface Vehicle

UTC – Universal Time Coordinated

CONTENTS

1. INTRODUCTION

The notion of "Robot" has gained a foothold in modern culture, being associated in the minds of the public with large and heavy machines that work tirelessly towards a single goal, most commonly in the form of robotic manipulators (also known as "robotic arms"). Mainly applied to mass-production, they can be found in factories around the globe, assembling cars or mounting electronics boards, for example, with inhuman speed and precision.

In the area of Mobile Robotics, autonomous navigation is defined as a robot's ability to navigate complex and highly dynamic indoor and outdoor environments without the aid of an operator. For complete navigation, a robot must be successful at its four building blocks: Perception, Localization, Cognition and Motion Control [28, p. 265]. In particular, the problem of localization has received attention lately, in part due to the decreasing costs and increasing quality of sensors. This problem consists on determining a robot's 3D position and orientation (x, y, z, θ), also known as the robot pose. Solving the localization problem is of utmost importance in Mobile Robotics, as almost every task a robot may perform depends on information of its position in space. By improving this solution, we can build better and cheaper mobile robots, that can perform tasks better and more reliably.

The requirement for localization accuracy vary with each application. The quality of the estimated position is directly dependant on the quality of the sensors used, and unfortunately most sensors are inaccurate, as their measurements may be affected by noise, external disturbances, aliasing effects and drifting error [28, p. 267].

Another important aspect of perception to consider when designing a localization system is sensor availability. One of the most commonly used sensors for localization are the Global Positioning System (GPS). The GPS reports absolute position with reference to a fixed coordinate system, whose origin is placed on the center of the earth, by estimating the distances to at least 4 satellites and then using these distances to perform a technique called Multi-Lateration. Thus, if we place such a receiver in our robots, position information would readily be available. Unfortunately, this is not always the case, as the use of GPS is severely limited by environmental factors such as atmospheric conditions (which are out of our control) and by the robot's area of operation, as the GPS signal propagation times are affected by line-of-sight obstructions such as buildings and vegetation. In some cases, GPS measurements are simply unavailable due to these factors [28, p. 266][27, p. 746].

A practical example of an environment where GPS measurements are unavailable is the test arena[1] of the 4th Humanitarian Robotics and Automation Technology Challenge (HRATC) [20]. The motivation behind this challenge is to promote solutions to the humanitarian problem of landmine removal in war-torn areas using robots, thus reducing the monetary and human cost of landmine removal. The 4th edition of the challenge was conducted

[1] https://github.com/ras-sight/hratc2017_robot/tree/master/maps

in Porto Alegre, Brazil, where the selected test arena was an environment surrounded by buildings and vegetation. This choice of location was to encourage the competing teams to develop localization solutions that did not rely on GPS, as its availability can not be always guaranteed. Nevertheless, the teams and the organizers suffered from the lack of positioning, as outdoor localization is a non-trivial problem.

Inspired by the problems seen in HRATC, the goal of this project is to design and implement a localization system to be used by mobile robots in outdoor scenarios where there are little or no GPS measurements available. To collect data, a robot was constructed with a Pioneer 3-AT (P3AT) mobile base and a sensor suite consisting of an Inertial Measurement Unit (IMU), two wheel encoders and a common GPS. However, these sensors are ill-suited for outdoor navigation, as effects from rough terrain such as vibration and wheel slippage affect the IMU and encoders respectively, and as GPS is affected by line-of-sight obstructions between the GPS and satellites. Thus, the Received Signal Strength Indicator (RSSI) was used to enhance the robot pose estimation, a measurement of the received power in a Radio-Frequency (RF) transmission, which decreases with distance. A Radio-Frequency (RF) receiver was placed on the robot and three RF transmitters were placed in a triangle, and the received power for each pair was measured. The localization system thus consisted of and Extended Kalman Filter (EKF), fusing the RSSI for each of the three receiver-transmitter pairs with the IMU's gyroscope, the wheel encoders' odometry and the common GPS' data. To record the robot's ground-truth, a Real-Time Kinematics (RTK) GPS was used, an enhanced global positioning method with 2 to 5 centimeters of accuracy [11, p. 78].

To evaluate the characteristics and performance of our localization solution, three experiments were executed. In the first experiment, two datasets of the robot's sensors were collected with the robot completely still, in order to verify the sensors' variances and the effect of adding an absorption plate to the base of the RF receiver's antenna, a hardware modification proposed by Graefenstein and Bouzouraa [15] and reported beneficial to the RSSI measurement. The second experiment was to evaluate the accuracy and precision of the localization system, and six datasets were collected, three with the absorbing plate and three without. These two experiments were performed in a parking lot, without cars or nearby obstructions, with the robot performing a predefined zig-zag trajectory starting from a known initial position. As an RTK GPS was used for recording the ground-truth trajectory, an unobstructed skyview was required for this sensors' full operation. The third experiment was to evaluate the operation of the localization system in an environment without GPS signal. The environment chosen was a garden, with tall trees and a nearby building, where the ground-truth was a manually marked straight line, since the RTK GPS could not be used here. Six datasets were collected, three in the parking lot and three in the garden, with the same straight line trajectory.

The results obtained from the first experiment validate the encoders, gyroscope and GPS measurements for localization while eliminating the IMU's accelerometer measurements, due to great amounts of noise coming from vibrations induced by the parking lot's rough terrain. The inclusion of the absorption plate was also validated, as it decreases the RSSI measurement's variance. In the second experiment, the RSSI, gyroscope, odometry, and GPS measurements were fused in the EKF to recover the robot trajectory, which was then compared to the ground-truth from the RTK GPS to obtain the localization error. The inclusion of RSSI reduced the error, but a semi-constant bias present in the three RSSI corrupted the trajectory, especially in the datasets recorded with the absorption plate. As such, an Augmented Extended Kalman Filter (A-EKF) was implemented, fusing data from the same sensors as the EKF while iteratively correcting the biases for the three RSSI. The smallest mean error of 0.413 meters was achieved with the A-EKF by fusing the RSSI (while using the absorption plate) with wheel odometry and gyroscope measurements, without GPS, which is an improvement of 82.66% compared to the baseline GPS error (2.382 meters). Thus, the main contribution of this project is the development of a mobile robot localization system independent from GPS, obtained from the data fusion of gyroscope, odometry and RSSI measurements, with an absorption plate, in the A-EKF. A limitation of our solution is that the pose estimation error is unbounded, since the RSSI bias correction and pose estimation processes are co-dependant in the A-EKF: Pose estimation errors affect the bias correction, and vice-versa.

This dissertation is structured as follows. Chapter 2 offers theoretical background on Autonomous Mobile Robots, including the mobile platforms, sensors and localization methods commonly used, as well as intuition on the area of Radio-Frequency signal propagation. Chapter 3 describes the used resources, implementation and the methodologies for dataset collection and analysis. Chapter 4 details the results obtained from the models and sensors validation, as well as the results for pose estimation through sensor fusion with the EKF and A-EKF. Chapter 5 reviews the literature of Autonomous Robot Localization, discussing the different methods and the current state-of-the-art in the area, as well as the literature of Wireless Sensor Network localization. Finally, Chapter 6 discusses the results obtained, details the project's contributions and describes possible future works.

2. THEORETICAL BACKGROUND

This Chapter introduces the topics relevant to this project. In Section 2.1, the basic concepts of Autonomous Mobile Robots are discussed, specifically the common mobile platforms, sensors, localization methods and the Robot Operating System (ROS), the currently most widely used robotic framework. Section 2.2 explores the properties of Wireless Sensor Networks, including the concepts of Radio-Frequency signal propagation, the signal strength loss models and the methods for localization in wireless sensor networks.

2.1 Autonomous Mobile Robots

Mobile Robots have enjoyed a growth in applications recently. As the robots become smarter, they are capable of performing increasingly complex tasks autonomously. For example, they may work in inhospitable environments such as outer space [27, p. 1423], or to aid in construction [27, p. 1493] and agriculture [27, p. 1463].

There is a great deal of effort towards autonomous robot behavior, which is the ability of operating with reduced or no human intervention. The counter-example to autonomous behavior is *tele-operation*, where the robot is completely controlled by an operator. A formal definition of autonomy can be achieved by dividing it in three cases, in regards to the level of human involvement [23]:

- *Human-in-the-loop* is when a robot operates for an amount of time and then waits for commands in order to continue. In this case, the human has knowledge of the robot's mission (i.e. move parts on the ground floor of a factory) and provides specific high-level tasks (i.e. go from point A to point B), and the robot is responsible for low-level autonomous planning and locomotion.
- *Human-on-the-loop* is used to describe monitoring or supervisory action by an operator. In this case, the robot has knowledge of its mission and performs all tasks autonomously, but a human has the ability to interfere in dangerous scenarios or in the case of failure.
- *Human-off-the-loop*, or *fully autonomous*, is when the robot is completely independent on human intervention. It has knowledge of its mission, performs all tasks autonomously and is responsible for failure avoidance and recovery.

In all three cases, localization plays an integral role. The necessary accuracy will depend on the mission and the type of robot, but some knowledge of its position is always required. In this project, we exploit the Received Strength Signal from Radio-Frequency

transmissions in order to improve the accuracy of the localization solution of a ground robot. As such, this section will explore the most common mobile ground platforms, the sensors used in this project, the localization methods implemented and the robotic framework used in development.

2.1.1 Ground Robots

Locomotion is the ability of a robot to move around in an environment without restriction [28, p. 13]. There is a wide variety of possible movements for a robot to perform, and selecting the most appropriate is an important step in the robot's design phase. Wheels are simple and extremely efficient for locomotion on flat ground. However, this efficiency greatly depends on the ground's flatness and hardness [28, p. 14], as wheeled robots perform very poorly (or not at all) on unstructured or unstable terrain.

As man-made infrastructure on Earth tends to be flat and hard, conventional wheels have been the most popular locomotion mechanism in Mobile Robotics [28, p. 35]. Apart from the high efficiency and ease of implementation, a wheeled robot usually has no problems with balance, as they are designed so that all wheels are in contact with the ground at all times. Wheeled robot design is more concerned with traction, stability and maneuverability, all which depend on the movement model employed. The design space of a wheeled robot is vast, with countless possible wheel arrangements and models. For example, there are robots with as many as 6 wheels, and robots with only two wheels. One of the most popular movement mechanisms is the differential drive, in which the two wheels are mounted on a common axis and are rotated independently.

2.1.1.1 Differential Drive Model

The main idea behind differential drive mechanism is that the robot's movement is dictated by the difference in rotation speed between its two wheels [9, p. 39]. It can be mathematically modelled by considering that the vehicle's movement is described as a circle, with angular velocity ω, turning about a point on the same axis as the wheels, called the Instantaneous Center of Curvature (ICC). The right and left linear velocities, V_r and V_l respectively, describe the robot's total linear velocity V, tangential to the curvature dictated by the ICC:

$$V = V_r + V_l \tag{2.1}$$

V_r and V_l themselves can be described in terms of the robot's angular velocity ω:

$$\begin{cases} V_r = \omega \times (R + \frac{L}{2}) \\ V_l = \omega \times (R - \frac{L}{2}) \end{cases} \tag{2.2}$$

where L is the distance between the center of the two wheels and R is the distance from the center of curvature to the midpoint of L. Figure 2.1 provides insight on the model, depicting the linear velocities, the location of the ICC, the angular velocity and the distances. Applying the rule of subtraction in the linear equation system (2.2), we can find ω as:

$$\begin{aligned} \omega L &= V_r - V_l \\ \therefore \omega &= \frac{V_r - V_l}{L} \end{aligned} \tag{2.3}$$

We can also determine the ICC radius R by solving the equation system (2.2) with the rule of addition and by substituting ω by the result of equation (2.3):

$$\begin{aligned} 2\omega R &= V_r + V_l \\ \therefore 2(\frac{V_r - V_l}{L})R &= V_r + V_l \\ \therefore R &= \frac{L}{2} \times \frac{V_r + V_l}{V_r - V_l} \end{aligned} \tag{2.4}$$

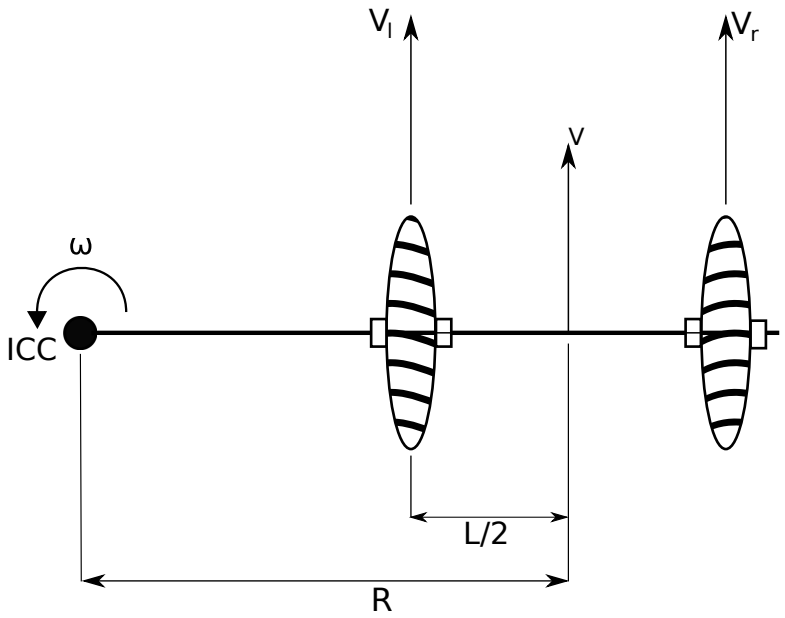


Figure 2.1: Differential drive model [27, p. 738]

With R and ω, the vehicle's trajectory is defined. R defines the robot's direction and curvature of movement, while ω defines the speed in which it turns. As they are calculated by the difference in linear wheel velocities, four special cases arise [9, p. 40] (illustrated in Figure 2.2):

1. $V_l > V_r$: R and ω become negative. The ICC lies on the right side of the vehicle, indicating clockwise movement with negative angular velocity. The distance from the

ICC to the vehicle indicates the trajectory's curvature. If the difference between V_r and V_l is large, the vehicle will turn in a small circle.

2. $V_l < V_r$: R and ω become positive. It is the opposite of case 1, where now the ICC lies on the left, indicating counter-clockwise movement with positive angular velocity.

3. $V_l = V_r$: ω and the denominator part of R become zero (ICC lies on infinity), meaning that there is no angular movement. The vehicle moves forward in a straight manner. A sub-case of this is $-V_l = -V_r$, where the vehicle moves backwards in a straight manner.

4. $V_l = -V_r$: The numerator part of R becomes zero, meaning that the ICC is at the mid-point between the wheels. The vehicle rotates in place counter-clockwise with angular velocity $2V_r/L$. By symmetry, $-V_l = V_r$ is the same except for clockwise rotation and $-2V_l/L$ angular velocity.

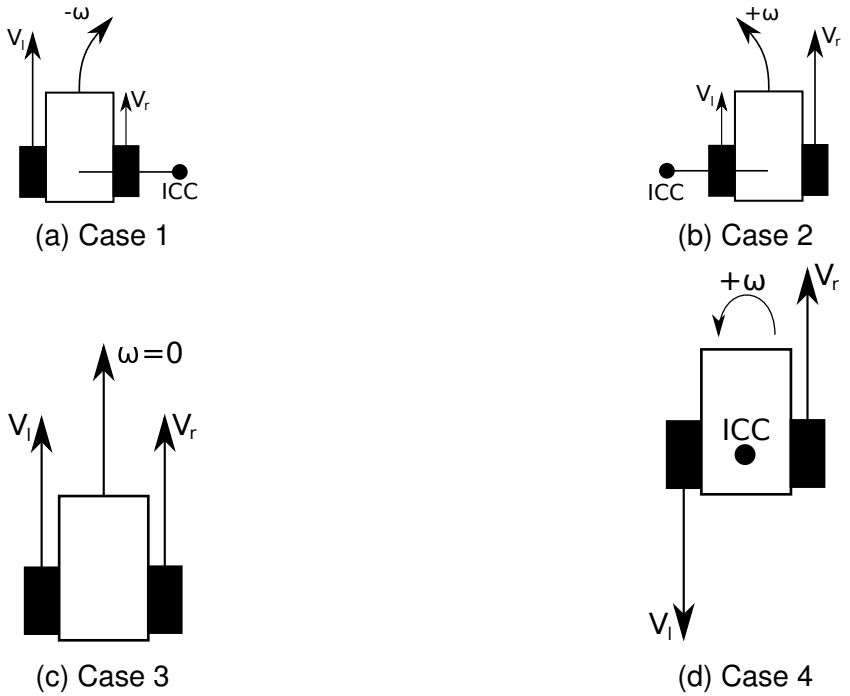


Figure 2.2: Four possible differential drive trajectory cases

2.1.1.2 Forward Kinematics

The differential drive model is not without limitations. For example, movement parallel to the wheels' axis is impossible: The vehicle can only move forwards and backwards, albeit with curvature and angular velocity. Another limitation is that perfectly straight movement is very hard to do in practice, as there will always be some small difference in the wheel velocities. Finally, though the model gives us an idea of the trajectory through the angular

velocity and ICC, it does not informs us on the vehicle's (x,y) position. This information can obtained by expanding differential drive to be a particular solution of the *Forward Kinematics* problem [9, p. 41].

To understand this problem, consider a robot in a two-dimensional Cartesian space, with position (x,y) and its front at an angle θ with the x axis (the robot's orientation). These three coordinates describe the robot pose ξ, formally defined as $\xi = [x, y, \theta]$. By manipulating the linear wheel velocities, we can make the robot assume different poses in the Cartesian space, here defined as the *global coordinate frame*. As such, modelling the robot pose in this Cartesian space is done by considering the pose transformation between the global frame and the robot frame [28, p. 59], as seen in Figure 2.3.

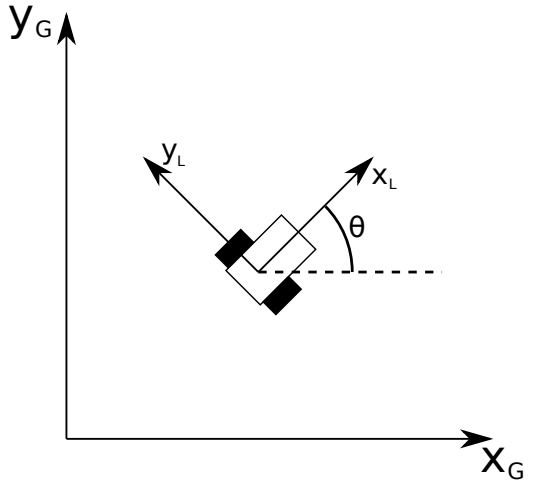


Figure 2.3: The global and local coordinate frames [28, p. 59]

Let ξ_G be the robot pose relative to the Cartesian frame, also known as the *global* frame, and ξ_L be the robot pose relative to its own *local* frame:

$$\xi_G = \begin{bmatrix} x_G \\ y_G \\ \theta \end{bmatrix} \tag{2.5}$$

$$\xi_L = \begin{bmatrix} x_L \\ y_L \\ \theta \end{bmatrix} \tag{2.6}$$

The transformation between them is given by the rotation matrix $R(\theta)$ [28, p. 60]:

$$\xi_L = R(\theta)\xi_G = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_G \\ y_G \\ \theta \end{bmatrix} \tag{2.7}$$

As we are interested in knowing the robot's pose in the global frame, we can solve Equation (2.7) for ξ_G:

$$\xi_G = R(\theta)^{-1}\xi_L = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_L \\ y_L \\ \theta \end{bmatrix} \tag{2.8}$$

The robot's motion in the global frame can be described by its velocities, obtained by derivating the global pose:

$$\dot{\xi}_G = R(\theta)^{-1}\dot{\xi}_L = R(\theta)^{-1} \begin{bmatrix} \dot{x}_L \\ \dot{y}_L \\ \dot{\theta} \end{bmatrix} \tag{2.9}$$

where $\dot{\xi}_L$ contains the robot's frontal, lateral and angular velocities. This is known as a *velocity motion model* [30, p. 121], which considers $\dot{\xi}_L$ as an input vector u to the robot. As such, Equation (2.9) can be rewritten as:

$$\dot{\xi}_G = R(\theta)^{-1}u = R(\theta)^{-1} \begin{bmatrix} v_{front} \\ v_{lat} \\ \dot{\theta} \end{bmatrix} \tag{2.10}$$

where v_{front} and v_{lat} are the robot's frontal and lateral velocities respectively, while $\dot{\theta}$ is the angular velocity ω. By applying the differential drive model to the velocity motion model, we have a specific solution to the Forward Kinematics problem [28, p. 61], which describes motion in terms of wheel velocities and distance between wheels. To do this, first we must consider the restriction of the differential drive model that there is no movement in the wheels' axis. As such, the lateral velocity v_{lat} will always be zero. The frontal and angular velocities are defined in Equations (2.1) and (2.3), and thus the differential drive solution to Forward Kinematics is given by:

$$\dot{\xi}_G = R(\theta)^{-1}u = R(\theta)^{-1} \begin{bmatrix} V_r + V_l \\ 0 \\ \frac{V_r - V_l}{L} \end{bmatrix} \tag{2.11}$$

The robot's global pose in relation to the input vector can be obtained by integrating $\dot{\xi}_G$ over time. In discrete-time, this is a finite integration from the initial global pose to the pose after an interval T:

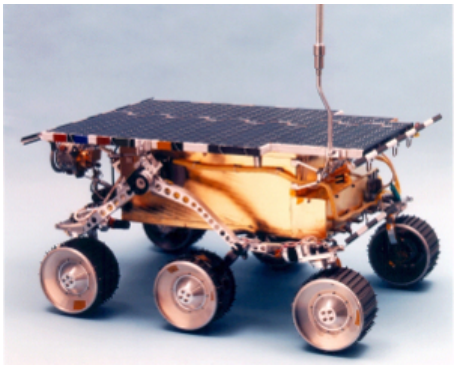
$$\xi_G(t) = \int_0^T R(\theta(t))^{-1}u(t)\ dt \tag{2.12}$$

As we are obtaining ξ_G by integrating past poses, this is said to be a *dead reckoning* method: An estimation method which depends on its past estimates.

2.1.1.3 Common Ground Robots

There are many examples of wheeled robots used in a variety of practical applications. An interesting one is in the area of space robotics, in which NASA frequently employs ground robots for extraterrestrial exploration missions. The Sojourner (Figure 2.4a) is such a robot, used to explore the surface of Mars in 1997 [28, p. 2]. Another example is the HELPMATE (Figure 2.4b), a mobile robot for transportation of medicine and instruments in a hospital [28, p. 6]. It is capable of autonomous navigation through various sensors, of which the main is a camera pointed at the ceiling. Through it, the robot is able to perform localization by identifying ceiling lamps as landmarks.

There are also commercially available ground robots used in academic research, usually referred to as "platforms", due to the fact that they are merely mobile bases from whence a complete robot may be built. The Pioneer P3-AT (Figure 2.5a) is such a platform, designed to be used in outdoor environments. Its priority is to be capable of supporting a high payload, in order to house numerous or heavy sensors. On the other hand, an example of research platform for indoors use is the TurtleBot. Currently on its third iteration (Figure 2.5b), this robot is designed to be used in more controlled environments, for prototyping and testing of autonomous behavior.



(a) The Sojourner NASA rover

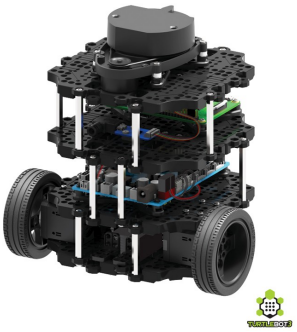
(b) HELPMATE medical robot

Figure 2.4: Examples of wheeled ground robots used in practical applications

2.1.2 Mobile Robot Sensors

As discussed previously, sensors are integral parts of a mobile robot. Through them, the robot perceives the world and is able to infer information that will be used to make decisions. This relates to the field of perception, where the main objective is to extract useful data from the environment or from the robot itself, with one or more sensor measurements

(a) Mobile Platform Pioneer P3-AT



(b) Mobile Platform TurtleBot 3

Figure 2.5: Examples of mobile research platforms

[28, p. 101]. A wide variety of sensors is used in mobile robots, as it is usually advantageous to have an abundance of information regarding every part of the robot. Many important modules (i.e. Localization) use redundant sensors, in case of hardware failures and fault mitigation, which is when a sensor misreports data.

Sensors can be categorized by two main properties, which are further divided into two groups each. As such, there are four possible classifications for sensors. The first property is concerned with the perspective of the measurement relative to the robot:

- *Proprioception* is specific to *internal* measurents of the robot. Proprioceptive sensors monitor the state of the robot in relation to itself [28, p. 101] [27, p. 91]. Examples are temperature, voltage, joint angle, motion and wheel rotation sensors.
- *Exteroception* is the opposite, where measurements are of the environment, *external* from the robot. Exteroceptive sensors monitor the state of the robot in relation to the environment. Examples are ultrasonic sensors, cameras and tactile sensors (measure the contact of the robot with a surface) [28, p. 104].

The second property relates to the manner in which the measurements are made:

- *Passive* sensors make measurements by *absorbing* energy. In other words, they observe a quantity or event through the information that is entering the sensor on its own. Examples of passive sensors are temperature probes (absorbs thermal energy), microphones (mechanical waves) and cameras (light).
- *Active* sensors detect changes in an environment produced by an emission of energy from the sensor itself. In other words, they *emit* energy into the environment and then measure the response. Examples are laser rangefinders, GPS and quadrature encoders.

These properties are mutually exclusive between the groups, but not between the properties. A sensor may be any combination of the first and second properties, such as an

ultrasonic sensor, which is exteroceptive active, or a camera, which is exteroceptive passive. As active sensors allow for more controlled interactions with the environment, typically they produce more accurate results. However, there are several challenges in active sensing, often introduced by the interaction of the emitted energy with the environment, such as scattering or ambient absorption, or other sources of interference. For example, a robot using two ultrasonic sensors may have error in the second measurement due to residual sound from the first sensor, and as such the sensors are interfering with each other [28, p. 103].

Only the sensors used in this project will be explored here, namely wheel rotation (encoders), Inertial Measurement and Global Positioning System sensors. The Receiver-Transmitter pairs are similar in functionality to the GPS, and thus can be considered as sensors from the robot's point of view. However, they will be discussed in Section 2.2, as their distance measurement originates from the area of Wireless Sensor Network Localization.

2.1.2.1 Encoders

Encoders are simple, proprioceptive sensors, commonly used in Mobile Robotics to measure revolutions in a robot's motor axes, and thus wheel rotation and velocity. For this reason, they are also known as wheel/motor sensors. Due to the fact that they are established and well understood, encoders are varied, high-quality and low cost [28, p. 115]. One of the most popular and accurate types is the optical encoder, which consists of a disk with thin slits through which light may pass. This disk sits in the middle of a light emitter-receiver pair and, as the motor rotates the disk, the light stream is interrupted, generating a sequence of pulses for each revolution. An encoder's output is normally the number of pulses measured in a fixed interval of time δt, called *ticks*.

A variant on the optical encoder is the quadrature encoder, used extensively in Mobile Robotics. In this version, two emitter-receiver pairs are used, referred to as channels A and B, with the second pair shifted 90º relative to the first. In other words, the two pairs are placed in a way that initially the first pair is facing a slit, while the second pair is not. The phase relationship between the two channels can be used to determine the direction of rotation of the motor shaft, as seen in Figure 2.6. For example, if the second pair is shifted 90º clockwise, then the shaft rotation will be clockwise when the pulses come first from channel A. If the pulses come first from channel B, the direction of rotation has been inverted [28, p. 116] [27, p. 94].

It is possible to estimate the angular velocity of a wheel (ω_{wheel}) by measuring how much it rotated in a given time interval δt, using the encoder's resolution (α) and output in ticks:

$$\omega_{wheel} = \frac{\alpha \times \text{ticks}}{\delta t} \tag{2.13}$$

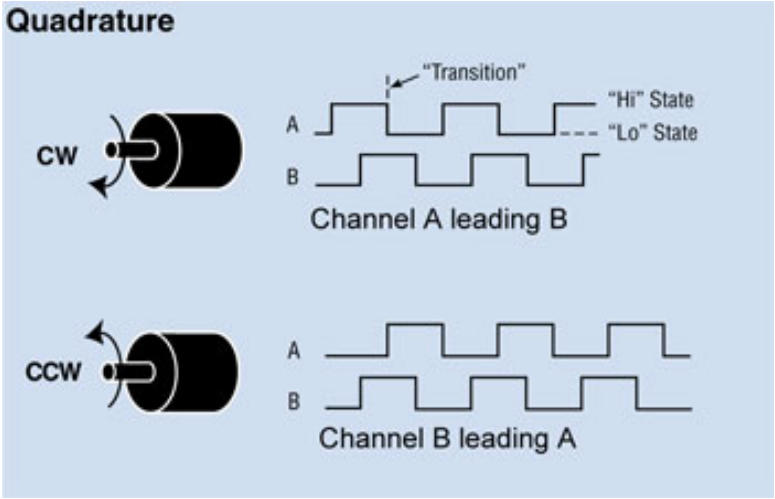


Figure 2.6: Operation mode of a quadrature encoder

Assuming the wheel has perfect traction with the ground, the amount of forward movement over time can be found by using the wheel's perimeter length. Multiplying the amount rotated in the time interval by the wheel radius r, and then dividing by the interval δt, gives us the forward distance travelled by the wheel in δt, which is precisely the definition of linear instantaneous velocity:

$$v_{wheel} = \frac{\alpha \times \text{ticks} \times r}{\delta t} \tag{2.14}$$

2.1.2.2 Inertial Measurement Unit

An Inertial Measurement Unit is an aggregate of sensors that is capable of estimating position, velocity and acceleration based on motion. It measures position in 6 Degrees-of-Freedom using accelerometers and gyroscopes, which are proprioceptive passive sensors. Specifically, three orthogonal accelerometers are used to measure linear accelerations and three orthogonal gyroscopes are used to measure angular velocities, a pair for each dimension as seen in Figure 2.7 (an orthogonal sensor is sensitive to measurements in a specific axis). The robot's orientation θ is directly integrated from the angular velocities, and the robot's position (x, y, z) is integrated from the linear accelerations.

As the acceleration data is represented in multiples of the Earth's gravity (i.e. 1g, 2g), there are biases in the accelerations due to Earth's non-constant gravitational field. To remove this effect, the direction of the current gravity vector is estimated from the orientation and extracted from the acceleration measurements. A scheme of this process is depicted in Figure 2.8 [27, p. 744].

The IMUs are extremely sensitive to noise and external disturbances, requiring precise calibration to eliminate biases and then be used effectively. Error in the gyroscopes creates a snowball effect: It leads to wrong orientation estimates, which leads to wrong es-

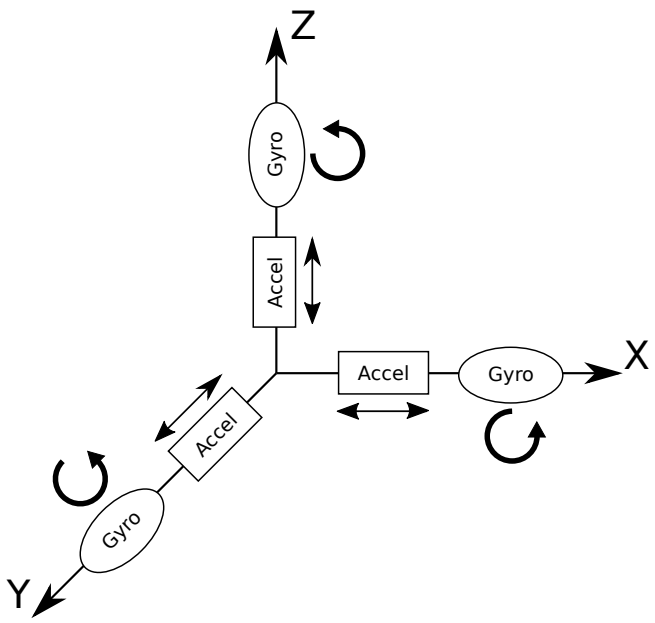


Figure 2.7: The Inertial Measurement Unit's sensors

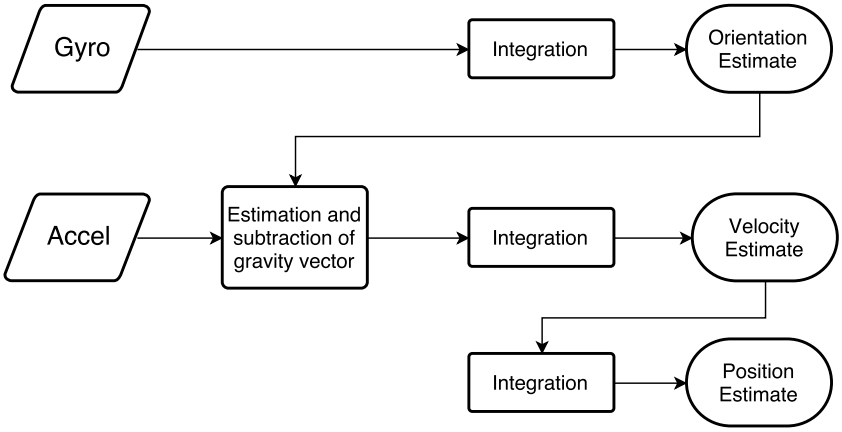


Figure 2.8: The IMU's position, velocity and acceleration estimation process [27, p. 744]

timations of the gravity vector, which leads to wrong velocity and position estimates. In spite of that, gyroscopes are robust sensors, and the orientation estimate is less prone to propagating errors, as the angular velocity measurements only need to be integrated once. In the case of the accelerometers, its measurements have to be integrated twice to obtain position, and thus errors are doubly propagated (quadratic errors). With enough time, these effects cause the estimations to drift (accumulate error), and then external, more reliable sensors are needed to correct this. For this reason, there are IMUs that integrate magnetometers (digital compasses), and GPS (originating sensors known as Attitude-Heading Reference Systems).

2.1.2.3 Global Positioning System

The Global Positioning System is the most commonly used localization mechanism in the world [27, p. 744]. It determines the position of a device on the ground through an

array of 30 satellites, which have been arranged in a manner that guarantees that there are at least 4 satellites visible in the sky at any given time and in any place on Earth [27, p. 745].

In order to perform position estimation, each satellite has a highly accurate atomic clock, and their location in orbit is always known with precision. They continuously transmit their own time and positions, which are received by a GPS device on the ground. This receiver computes the signal propagation time from the satellites by comparing the received time with its own time. With the times from 4 or more satellites, *pseudo-ranges* are calculated, which are distance estimations corrupted by measurement noise. Finally, the position is estimated through a Kalman Filter (discussed in subsection 2.1.3.4), based on the geometric problem of Multi-Lateration [27, p. 745].

Figure 2.9 illustrates this problem. The square represents a GPS device, which receives signals from three emitters, A, B and C. The pseudo-ranges are represented as d_A, d_B and d_C. The first distance localizes the receiver on a circle of radius d_A, whose center is emitter A. The second distance also describes a circle in the same manner, now of radius d_B and centered in emitter B, which intersects with the first circle in two points. The receiver's position is then contained on an infinite line described by the two points. A third circle with radius d_C intersects with the other circles in a single point, which corresponds to the receiver's position in two dimensions (x,y). In three dimensions, the same logic applies, only now d_A, d_B and d_C describe the radii of spheres. The intersection from four spheres gives us the position (x,y,z).

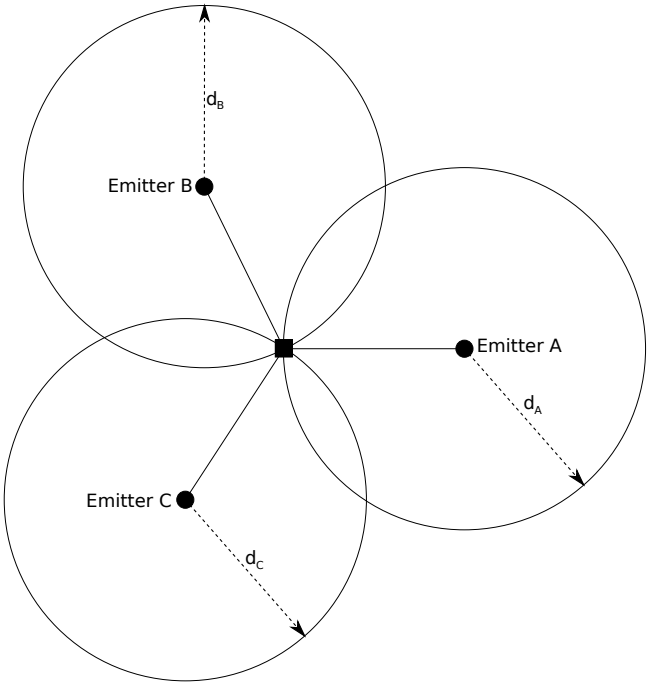


Figure 2.9: The Multi-Lateration problem [27, p. 746]

GPS are extensively used in robots due to their wide availability, and the fact that they provide good general positioning for a relatively low cost. However, as discussed in Chapter 1, its main limitation is availability. Their accuracy greatly suffers when used in

areas with line-of-sight occlusion between the receiver and the satellites, or with the effect of multi-path when there are many large objects nearby. As such, it can not be used in indoor or underground environments, for example. The number and arrangement of satellites also affects accuracy, which decreases if there are few satellites or if they are closely clustered together. Finally, atmospheric conditions affect the signal propagation times, and thus cause distance estimation errors that in turn affect the position estimation [27, p. 746]. When used in mobile robots, the GPS is usually not sufficient, as their position estimation also does not include any information on the robot's orientation, and a second sensor must be used to extract that information (typically a gyroscope or compass).

2.1.2.4 Real-Time Kinematic Positioning

Real-Time Kinematics (RTK) is not a unique sensor or measurement per se, but instead a method for enhancing the Global Positioning Systems, greatly increasing their accuracies. It works with two GPS receivers: One on the robot, called the *Rover*, and one stationed in a position known with high accuracy, called *base* [11, p. 78]. The base at first works like a normal receiver, computing the pseudo-range distance measurements from the satellites' information. However, as the base's position is known, ideal pseudo-ranges are computed and then an error can be calculated. This error is passed on to the rover, which uses it to refine its own distance measurements in real time and estimate its position accurately through Multi-Lateration [27, p. 748].

The difference between normal GPS and RTK lies in the manner in which the satellite-receiver distances are calculated. RTK uses information from the transmission's electromagnetic carrier wave itself, instead of the received times as before. Specifically, the distance is calculated by estimating the number of carrier wave cycles n that occurred in the path of the electromagnetic transmission, and then by multiplying this number of cycles by the signal's wavelength λ [11, p. 21]. Figure 2.10 provides intuition on this concept. For the sake of convenience, let us assume that the wave is described perfectly by a sine function, moving from satellite to receiver. One period of the sine corresponds to one cycle in the carrier wave. The distance travelled by the wave in one cycle is called the wavelength, which can be calculated as $\lambda = c/f$, where c is the speed of light in meters-per-second, and f is the wave's frequency in Hertz. With knowledge of the wavelength (i.e. how much a signal travelled in one cycle) and the number of cycles in a transmission, it is possible to determine how much the signal travelled to reach its destination.

The problem is that estimating the number of cycles in a transmission is non-trivial. Determining the unknown n is an *integer ambiguity problem*, and its solution is called *ambiguity resolution* [11, p. 85]. The initial number of cycles is normally estimated with the uncorrected GPS position, using least-squares regression or Kalman filtering, but this results in a real number. As we know that n is an integer, refining this estimate is necessary

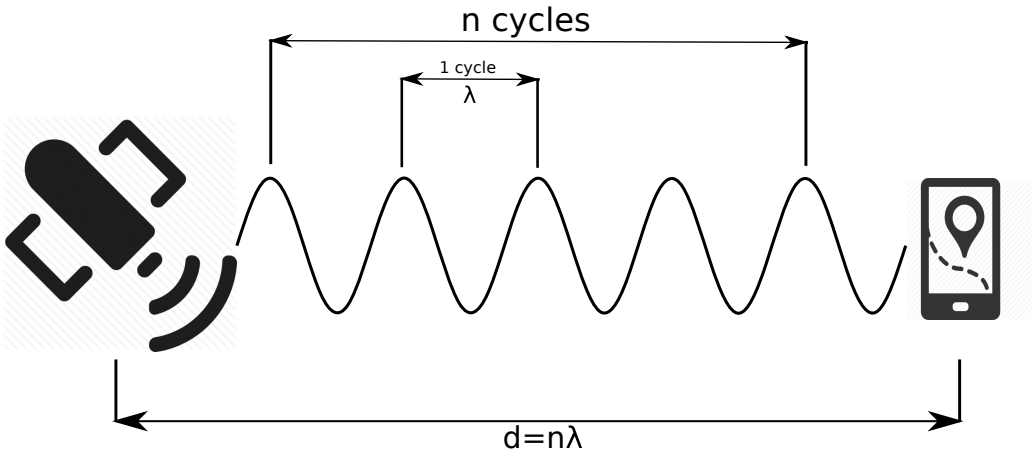


Figure 2.10: Real-Time Kinematic distance estimation through carrier wave cycle estimation

to achieve high accuracy. One method for this is known as *On-the-fly* resolution, which consists in defining an elliptical likelihood field around the rover's position, calculated with the initial estimation for *n* [11, p. 89]. Since *n* is an integer, a grid is drawn in the likelihood field as seen in Figure 2.11, and the intersections between the lines give us the possible rover positions which correspond to the true integer value of *n*. This space of intersections is searched for the most likely estimation, and the process is repeated.

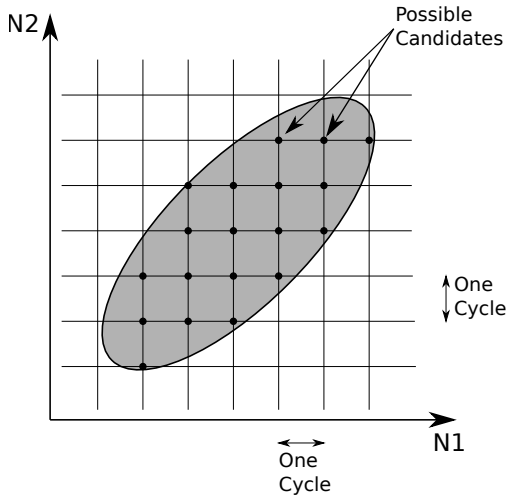


Figure 2.11: On-the-fly ambiguity resolution [11, p. 89]

As the quality of the initial ambiguity estimation is often poor (i.e. the GPS position is wrong), RTK receivers usually have large convergence times for accurate positioning. One advantage is that the carrier-wave cycle distance measurements have a finer resolution compared to conventional pseudo-ranges, and thus the RTK is capable of more accurate measurements. On the other hand, RTK implies a communication link between rover and base, which limits the operation range of an RTK-enabled robot. This issue can be somewhat alleviated by using the Continuous Operating Reference Stations (CORS), which are reference GPS base stations that broadcast their corrections over the internet via the Networked Transport of RTCM via Internet Protocol (NTRIP) [14], a standard RTK correction

broadcasting protocol. In Brazil, a CORS network is provided by the Instituto Brasileiro de Geografia e Estatística (IBGE), which has base stations in most state capitals. It is important to note that RTK-enabled GPS receivers are still susceptible to the same problems as a regular GPS, such as atmospheric condition errors, multi-path interference and line-of-sight occlusion.

2.1.3 Localization Methods

As stated in Chapter 1, autonomous navigation is one of the greatest challenges for mobile robots. In regards to localization, it is formally defined as the problem of determining a robot's pose $\xi = [x, y, \theta]$ in space through sensor data [30, p. 191]. One of the sources of error in localization are deterministic or systematic errors, which can be eliminated through calibration [28, p. 270]. For example, an image that has been distorted by the camera lens, which can be un-distorted with the camera's intrinsic parameters obtained through calibration.

Sensor noise is another influential factor in localization quality, defined as random deviations on a sensor's measurement centralized around the true measurement (when unbiased) [28, p. 267]. They may occur due to internal disturbances, such as supply voltage fluctuations, or external disturbances, such as vibration on the robot's rigid body. *Sensor aliasing* is another cause for problems, causing the sensors to yield little useful information. It relates to the uniqueness of the sensor inputs for different states, or how different each possible pose looks from the robot's perspective. For example, if two possible poses have exactly the same sensor inputs, there is ambiguity in the position estimation.

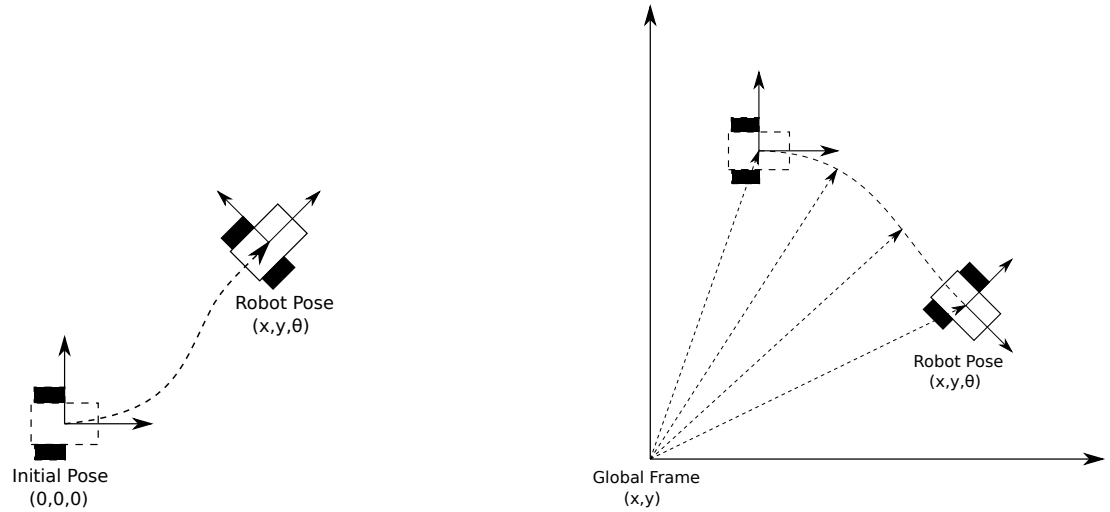
Finally, uncertainty introduced by effector performance can be just as damaging to localization. This is caused by the effectors' inability to produce exactly the desired output. For example, in practice it is virtually impossible for a motor to turn its axis exactly 360º, and as such there is uncertainty in how much it really turned. Uncertainy also comes from incomplete environmental models. A robot estimating its own position with encoder information through the Forward Kinematics model is none the wiser when its wheels slip due to unstable terrain, affecting its real pose. From the perspective of the robot, nothing happened, but in reality it lagged behind a bit, causing the real pose and the estimate to diverge.

There are a few factors which dictate the complexity of the localization solution. One of them is the availability of an initial condition, which splits the localization problem in two categories [30, p. 193]:

- *Position Tracking* is when the initial pose is known, and the solution is to follow the robot as it moves around while eliminating or reducing the effect of the errors discussed above. This is a local problem, as the errors are locally distributed around the

true pose. In other words, it is only possible to estimate a local pose, relative to the coordinate frame given by the initial condition, as seen in Figure 2.12a.

- *Global Localization* is the opposite, where the robot does not know where it is initially. The robot pose must be estimated relative to a global coordinate frame, as seen in Figure 2.12b. In this case, we can not assume the estimation error to be locally bounded, and normal distributions are often inadequate uncertainty models here. This localization variant is much more complex, subsuming position tracking. In other words, a solution to global localization is also a solution to position tracking.



(a) Position Tracking: The trajectory is obtained relative to the initial pose

(b) Global Localization: The trajectory is obtained relative to a global coordinate frame

Figure 2.12: Variations on the localization problem regarding initial conditions

Another factor is if the environment is *static* or *dynamic* [30, p. 194]. *Static* environments are typically indoors, controlled environments (i.e. laboratory), where only the robot moves and all other elements remain stationary. Thus, changes in the robot state are consequences only of the robot's actions. These environments are deterministic (with not much uncertainty) and can be adequately approximated by mathematical models. *Dynamic* environments are most common in the real world, where elements other than the robot change their configuration over time. The more persistent changes are of interest, those which significantly affect sensor measurements and thus the localization solution. Examples are moving people, cars, tree leaves in the wind, daylight (for cameras), etc. Localization in dynamic environments is obviously more difficult, and approaches to dealing with dynamism include modelling the more prominent dynamic elements and then eliminating their effect, and sensor data filtering.

Although there are other factors that influence complexity, these two are the most important in the case of this project, which is concerned with the position tracking of a ground robot in an outdoor dynamic environment. Thus, prominent methods for position tracking

will be discussed here, namely the Kalman Filter and its variations. For later comparison between this project and the work by Graefenstein and Bouzouraa [15] it is based on, the Particle Filter is discussed as well, a probabilistic estimation method which solves global localization. Some basic concepts on probability will also be explored, important to the understanding of the Kalman Filter algorithms.

2.1.3.1 Belief Representation

Apart from deterministic systematic errors, other sources of error can be considered *stochastic*. They induce random variations in the observations made by a sensor, for example, deviating its measurement from the real value. The amount of noise and uncertainty define the amplitude of these deviations. Stochastic errors can be modelled by Probability Density Functions (PDF) [30, p. 14] [28, p. 299]. To illustrate, consider that a sensor's output $\overline{z}$ can be represented as the true value z corrupted by a random variable $\mathcal{N}$:

$$\overline{z} = z + \mathcal{N} \tag{2.15}$$

For the first measurement, let $\mathcal{N} = -1$, and thus the corrupted measurement lies on the left of z. Measuring a second time, let $\mathcal{N} = 0.5$, and thus the sensor output now lies on the right of z. With enough measurements, we can see that $\mathcal{N}$ produces a set of corrupted measurements centered around the true value. This behavior can typically be modelled by a probability density function, where the true value is equal to the mean μ, and $\mathcal{N}$ is equal to the covariance σ^2. The normal distribution, also known as the *Gaussian* distribution, is the best known model for probability density, as seen in Figure 2.13.

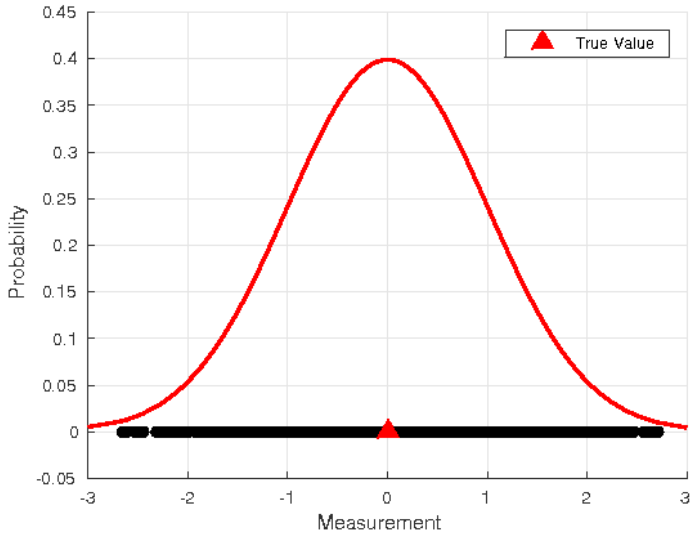


Figure 2.13: Gaussian PDF of a stochastic measurement ($\mu = 0,\ \sigma = 1$)

When z is a scalar, the Gaussian distribution can be mathematically defined as [28, p. 300]:

$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \times e^{-\frac{(z-\mu)^2}{2\sigma^2}} \tag{2.16}$$

When there are multiple different measurements, z is a k-dimensional vector and the Gaussian PDF can be defined as:

$$p(z) = \frac{1}{(2\pi)^{k/2}\sqrt{det(\Sigma)}} \times e^{\frac{1}{2}(z-\mu)^T\Sigma^{-1}(z-\mu)} \tag{2.17}$$

where Σ is a positive semi-definite symmetric matrix known as the *Covariance* matrix, containing the cross-covariances between the elements of the measurement vector. Thus, a PDF gives us a probabilistic representation of a single stochastic variable with its mean μ and covariance σ^2, or of a vector of stochastic variables with their means vector and covariance matrix Σ. This is known as the *belief representation* of a stochastic variable:

$$bel(z) = p(z) = (\mu_z, \sigma_z^2) \tag{2.18}$$

In mobile robots, this belief is expressed in terms of conditional probabilities, and is closely related to the Bayes Theorem which will be discussed ahead [30, p. 25].

2.1.3.2 Bayes Theorem

The *Bayes Theorem*, also known as *Bayes' Rule*, is one of the most important probability concepts, especially in the case of Mobile Robotics [30, p. 16]. The rule relates the *conditional probability* $p(A \mid B)$ of an event A given that an event B is true, and vice-versa. The Bayes' Theorem can be mathematically defined as:

$$p(A \mid B) = \frac{p(B \mid A)}{p(B)} p(A) \tag{2.19}$$

for $p(B) \neq 0$, where $p(A)$ is the probability of A occurring independently, $p(B \mid A)$ is the inverse conditional, or the probability B occurs given A is true, and $p(A)$ is the marginal probability, given by sum of the possible combinations in the probability space. In practice, the rule describes the conditional probability of an event based on prior knowledge and observations. In other words, it improves the prior probability estimate of an event with evidence.

When used iteratively to correct prior probabilities, the rule continuously improve its estimate towards the true probability. In mobile robots, Bayes' Rule can be used to continously improve a robot's pose estimate. For example, let x_t be the current state that describe the robot pose, x_{t-1} be the previous pose and z_t be the current sensor measurements [30,

p. 17]. We can treat all of them as stochastic variables using PDFs, and then use Bayes' rule to correct the state probability using the previous state and sensor measurements:

$$p(x_t \mid z_t) = \frac{p(z_t \mid x_t)}{p(z_t)} p(x_{t-1}) \tag{2.20}$$

The previous state probability $p(x_{t-1})$ is known as the *prior probability*, which can be considered a naive estimation of the pose, before any observations. z_t is the observation data used to correct the prior. $p(z_t \mid x_t)$ is the inverse conditional, the probability of a set of measurements occurring given a specific state. Finally, $p(x_t \mid z_t)$ is the *posterior probability*, the corrected conditional of the state after incorporating the observations.

2.1.3.3 Odometry

The concept of *Odometry* can be dated back more than 2000 years ago, as the Roman architect Vitruvius describes it as "a useful invention of the greatest ingenuity, transmitted by our predecessors, which enables us to know how many miles of a journey we have accomplished" [27, p. 737]. Odometry in mobile robotics is a method for obtaining the trajectory travelled by a robot using data of the robot's motion, measured through sensors. Specifically, the coordinate transformation between a robot's previous and current pose is recovered from sensor data and a movement model (i.e. forward kinematics), and the current pose is integrated onto the robot's total trajectory. As such, this is a *position tracking* and *dead reckoning* method.

The simplest way to perform odometry in a ground robot is by measuring how much its wheels rotate, which is called *Wheel Odometry*. Using the differential drive solution to Forward Kinematics (Section 2.1.1.2), the robot pose $\xi = [x, y, \theta]$ can be obtained by integrating the robot's linear and angular velocities as seen in Equation (2.12). These velocities can be measured through encoder sensors, as discussed in Section 2.1.2.1. However, there are two limitations to Wheel Odometry, the first being that the movement models do not account for external events such as wheel slipping, which are consequences of the operation environment [27, p. 738]. The second is that the integrated trajectory is very likely to drift due to effector uncertainty, which are consequences of the robot's hardware. As odometry is a dead reckoning method, these errors will be integrated along with the pose estimates, eventually causing the estimation and true pose to diverge. Thus, even if the encoder measurements are of high quality, the environment uncertainty and robot imperfections cause errors in the trajectory estimation.

2.1.3.4 Kalman Filter

Belief representation and Bayes' theorem are the cornerstones of many mobile robot localization methods, in a field of study known as *Probabilistic Robotics*. Through

these two axioms, we can define a class of estimation algorithms where the pose states are modelled as Probability Density Functions, of which the means and covariances compose the belief, to be corrected with sensor data in Bayes' Rule. The belief is the conditional probability of the states x_t occurring given the sensor measurements z_t and control inputs u_t [30, p. 26]:

$$bel(x_t) = p(x_t \mid z_t, u_t) \tag{2.21}$$

This belief is the posterior, after incorporating the sensor measurements. This is the second step of the estimation algorithm, after the application of Bayes' Rule, called the *update* or *correction* step. It is of interest to also define the prior, the belief before incorporating sensor data and just after executing the control inputs u_t. This is the first step of the algorithm, called the *prediction step*:

$$\overline{bel}(x_t) = p(x_t \mid u_t) \tag{2.22}$$

Thus, we can define Algorithm 2.1 for estimating and correcting beliefs based on Bayes' Rule, the Bayes Filter. Its inputs are the belief at the previous algorithm iteration (or the initial belief), and the most recent control and measurements [30, p. 27]. The prediction step is performed in line 3 and the update step is performed in line 4. The Bayes Filter is *complete*, eventually producing a closed-form estimation of the belief. In other words, as the rule used iteratively produces ever more accurate results, $bel(x_t)$ converges to the true states.

Algorithm 2.1 Bayes Filter Algorithm

1: **function** *Bayes_Filter*$(bel(x_{t-1}), u_t, z_t)$
2: **for all** x_t **do**
3: $\overline{bel}(x_t) = p(x_t \mid u_t)\ bel(x_{t-1})$
4: $bel(x_t) = p(z_t \mid x_t)p(z_t)^{-1}\ \overline{bel}(x_t)$
5: **return** $bel(x_t)$

The intuition of Bayes' rule is at the core of the Kalman Filter. It is a specific case of the Bayes Filter, where Gaussian Probability Density Functions are used [30, p. 40]. To define its algorithm, first let us define the states that compose the belief to be estimated. In the case of mobile robot localization, these states correspond to the robot pose:

$$x_t = \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{2.23}$$

In continuous time, the next states are described by their derivatives. These differential equations are called the state-transition functions χ_t. For the robot pose, they are the linear and angular velocities:

$$\chi_t = \dot{x}_t = \begin{bmatrix} \dot{x}_{1,t} \\ \dot{x}_{2,t} \\ \dot{x}_{3,t} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{2.24}$$

In discrete time, χ_t is described as:

$$\chi_t = \begin{bmatrix} \Delta(x_{1,t}, x_{1,t-1}) \\ \Delta(x_{2,t}, x_{2,t-1}) \\ \Delta(x_{3,t}, x_{3,t-1}) \end{bmatrix} = \begin{bmatrix} \Delta(x_t, x_{t-1}) \\ \Delta(y_t, y_{t-1}) \\ \Delta(\theta_t, \theta_{t-1}) \end{bmatrix} \tag{2.25}$$

where $\Delta(x_{i,t}, x_{i,t-1})$ (i = 1..3) represents the finite difference (or discrete derivative) of a state $x_{i,t}$. The past value of the i-th state is represented as $x_{i,t-1}$. Over a fixed time interval T, the finite difference is written as $\Delta(x_{i,t}, x_{i,t-1}) = (x_{i,t} - x_{i,t-1}) \times T^{-1}$. This can be understood as the current state being calculated through the past state, a *discretization* method called *Euler Backward* [32]. Likewise, *Euler Forward* makes a prediction of the next state $x_{i,t+1}$ using the current state. In this case, $\Delta(x_{i,t+1}, x_{i,t}) = (x_{i,t+1} - x_{i,t}) \times T^{-1}$, over the same time interval T.

For the Kalman Filter's completeness, three conditions must be met. Consider a linear system in its state-space form, where x_{t-1} are the previous states, A_t is the state matrix and B_t is the control input matrix:

$$\chi_t = A_t x_{t-1} + B_t u_t \tag{2.26}$$

The first condition is that the state transition functions must be a linear with added Gaussian noise ϵ_t, which models the uncertainty in the state-space model [30, p. 41]:

$$\chi_t = A_t x_{t-1} + B_t u_t + \epsilon_t \tag{2.27}$$

The second condition is that the measurement functions z_t must also be linear with added Gaussian noise, defined as:

$$z_t = C_t x_t + \mathcal{N}_t \tag{2.28}$$

C_t is also a state-space vector and $\mathcal{N}_t$ is the measurement noise. Finally, the third condition is that the initial belief must be normally distributed. If these three conditions are met, the belief at any given time is always a Gaussian with mean μ_t and covariance Σ_t,

which represents the robot's most likely pose [30, p. 42]. Thus, the Kalman Filter algorithm [30, p. 42] is defined as:

Algorithm 2.2 Kalman Filter Algorithm

1: **function** *Kalman_Filter*(μ_{t-1},Σ_{t-1},u_t,z_t)
2: $\quad\overline{\mu}_t = A_t\mu_{t-1} + B_t u_t$
3: $\quad\overline{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R$
4: $\quad K_t = \overline{\Sigma}_t C_t^T \times (C_t\overline{\Sigma}_t C_t^T + Q)^{-1}$
5: $\quad\mu_t = \overline{\mu}_t + K_t(z_t - C_t\overline{\mu}_t)$
6: $\quad\Sigma_t = (I - K_t C_t)\overline{\Sigma}_t$
7: $\quad$**return** μ_t, Σ_t

The prediction step is performed in lines 2 and 3, where the prior probability is computed using the state-space matrices A_t and B_t, the previous states' covariance matrix Σ_t, and the covariance matrix R of the Gaussian state transition noise ϵ_t [30, p. 43]. The update step is performed in lines 4 to 6. Line 4 calculates the *Kalman Gain*, which dictates the degree to which the sensor observations will affect the correction of the prior belief. Q represents the covariance of the sensor measurements. Following the Bayes rule intuition, line 5 incorporates the difference between the actual sensor measurements and the expected sensor measurements into the prior belief, proportionally to the Kalman Gain. In other words, the error between the expected sensor data (the value the sensors should have at the estimated states x_t) and the actual sensor data is incorporated into the prior state belief, with varying degrees of importance according to the Kalman Gain. Finally, line 6 calculates the new posterior covariance through Bayes' rule, with the prior covariance and the Kalman Gain. As with the Bayes Filter, the Kalman Filter holds the same convergence property, meaning that eventually the estimated states will be equal to the true states, and the measurement error in line 5 will be zero. It is possible to incorporate as many sensor measurements as wanted in the measurement functions vector z_t. They will be *fused* when calculating the corrected states, and as such the Kalman Filter is normally known as a *sensor fusion* technique.

A good visual representation for how the filter works can be seen in Figure 2.14. Imagine a robot moving about in an 1-dimensional indoors environment. For localization, the robot has a sensor that can detect doors. The Gaussian PDF representing the initial pose belief has a low mean probability, as there is yet not much knowledge on whether that pose is correct. As the robot moves around, the pose belief gradually improves, as is wont of Bayes' rule and as the door sensor indicates the ever closer presence of a door. When the robot in fact passes by door, assuming that the sensor is very good, the mean probability of the pose belief is near the maximum value of 1. As the sum of all probabilities in a PDF must be equal to 1, this means that the probabilities other than the mean are small, and thus the distribution assumes a slender appearance. With enough time, the mean probability will become 1 and thus the pose estimation will be optimal.

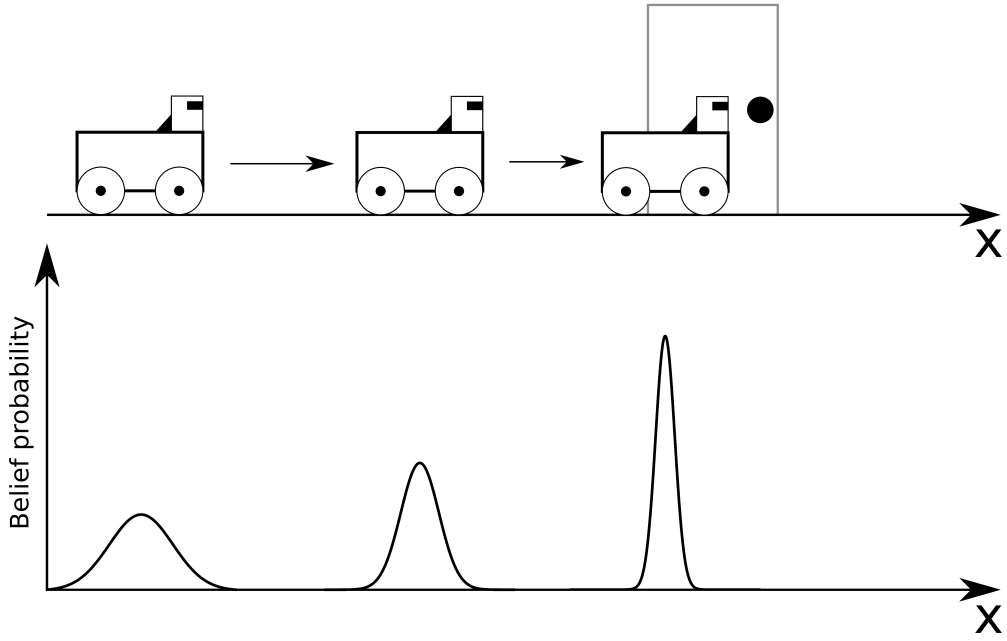


Figure 2.14: Illustration of the Kalman Filter process

In terms of computational complexity, the Kalman Filter is quite efficient. Its bottlenecks are the matrix operations, whose complexity depend on the matrix sizes. The filter is typically $\mathcal{O}(k^{2.4})$ complex, where k is the dimension of the measurement vector z_t. This stems from the matrix inversion in line 4. Even when using sparse updates (where the matrices are sparse), the algorithm is bounded at $\mathcal{O}(n^2)$, where n is the dimension of the state-space. This is due to the matrix multiplication $K_t C_t$ in line 6. In most robotic localization applications, the measurement space is much smaller than the state-space, with for example 1 or 2 measurements and 5 states. The update step's complexity is then dominated by $\mathcal{O}(n^2)$ operations.

2.1.3.5 Extended Kalman Filter

The correctness of the Kalman Filter (the capacity of producing an optimal estimation) depends on the conditions for linearity and initial Gaussian belief stated previously. The problem is that the the vast majority of state-space systems encountered in the real world are non-linear. The *Extended Kalman Filter* (EKF) is a variation on the Kalman Filter algorithm that can be used with non-linear systems [30, p. 54]. It starts by relaxing the linearity conditions for the state-space transition functions χ_t, with current and past states described by x_t and x_{t-1}, and the measurement functions z_t. Here, χ_t and z_t are given by two arbitrary non-linear functions g and h:

$$\chi_t = g(u_t, x_{t-1}) + \epsilon_t \tag{2.29}$$

$$z_t = h(x_t) + \mathcal{N}_t \tag{2.30}$$

The function g substitutes the A_t and B_t matrices in the first condition and the function h substitutes the C_t matrix in the second condition [30, p. 56]. The Extended Filter inherits the intuition from its linear counterpart, with the only difference being that the belief calculated is an approximation instead of an exact estimation. This is because, with the non-linear functions, the belief is longer a Gaussian distribution. Thus, the EKF can not exactly estimate the true states, but it can approximate them. As there is no way of calculating the mean and covariance in closed-form, first we must approximate linear functions from the non-linear g and h.

Linearization is the crux of the Extended Kalman Filter. It is the process of approximating a non-linear function by a linear function around a specific point in its state space. In the case of the EKF, this means approximating g and h to linear functions around the mean μ_t. The linearized functions may not be perfect representations of the state-space system, but the overall aspect of a Gaussian probability density distribution is mantained. There are many forms of linearization, and the *First-Order Taylor Expansion* is the one used in the EKF [30, p. 57]. Other Kalman Filter variations for non-linear systems use other methods, as with the *Unscented Kalman Filter* for example, which uses the Unscented Transform linearization [30, p. 65]. The Taylor Expansion gives us a linear approximation of a non-linear function g through its value and slope [30, p. 58]. The slope is defined as the partial derivative:

$$g'(u_t, x_{t-1}) = G_t = \frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}} \tag{2.31}$$

where G_t is a square matrix of size equal to the dimension of the state vector, called a *Jacobian*. The value for the linearization depends on the argument of g, that is, x_{t-1} (u_t are the control inputs). For a Gaussian PDF, the most logical choice is to use the mean μ_{t-1}, which is the most likely state at the time of linearization. Thus, the Taylor Expansion can be further defined as the approximation of g by its value at μ_{t-1} and the control inputs u_t:

$$g(u_t, x_{t-1}) = g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}) \tag{2.32}$$

In Equation (2.32), the approximation is obtained by combining the value of g at the mean μ_{t-1} with the product of the Jacobian and the error of the states (estimated states minus the mean). The exact same linearization process is repeated for the measurement function h, only now with the prior belief $\overline{\mu}_t$:

$$h(x_t) = h(\overline{\mu}_t) + H_t(x_t - \overline{\mu}_t) \tag{2.33}$$

with $H_t = \partial h(x_t)/\partial x_t$. To calculate the Jacobian G_t, we must take the partial derivatives of the state transition functions in regards to all n states, resulting in an $n \times n$ matrix. For brevity, let g_i ($i = 1..n$) denote the non-linear functions for each state:

$$G_t = \begin{bmatrix} \frac{\partial g_1}{\partial x_{t,1}} & \frac{\partial g_1}{\partial x_{t,2}} & \frac{\partial g_1}{\partial x_{t,3}} & \cdots & \frac{\partial g_1}{\partial x_{t,n}} \\ \frac{\partial g_2}{\partial x_{t,1}} & \frac{\partial g_2}{\partial x_{t,2}} & \frac{\partial g_2}{\partial x_{t,3}} & \cdots & \frac{\partial g_2}{\partial x_{t,n}} \\ \frac{\partial g_3}{\partial x_{t,1}} & \frac{\partial g_3}{\partial x_{t,2}} & \frac{\partial g_3}{\partial x_{t,3}} & \cdots & \frac{\partial g_3}{\partial x_{t,n}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial x_{t,1}} & \frac{\partial g_n}{\partial x_{t,2}} & \frac{\partial g_n}{\partial x_{t,3}} & \cdots & \frac{\partial g_n}{\partial x_{t,n}} \end{bmatrix} \tag{2.34}$$

Likewise, the H_t Jacobian is calculated using the partial derivatives of the k measurement functions in regards to the n states, resulting in a $k \times n$ matrix. Once again, let h_i ($i = 1..k$) be the non-linear functions for each measurement:

$$H_t = \begin{bmatrix} \frac{\partial h_1}{\partial x_{t,1}} & \frac{\partial h_1}{\partial x_{t,2}} & \frac{\partial h_1}{\partial x_{t,3}} & \cdots & \frac{\partial h_1}{\partial x_{t,n}} \\ \frac{\partial h_2}{\partial x_{t,1}} & \frac{\partial h_2}{\partial x_{t,2}} & \frac{\partial h_2}{\partial x_{t,3}} & \cdots & \frac{\partial h_2}{\partial x_{t,n}} \\ \frac{\partial h_3}{\partial x_{t,1}} & \frac{\partial h_3}{\partial x_{t,2}} & \frac{\partial h_3}{\partial x_{t,3}} & \cdots & \frac{\partial h_3}{\partial x_{t,n}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_k}{\partial x_{t,1}} & \frac{\partial h_k}{\partial x_{t,2}} & \frac{\partial h_k}{\partial x_{t,3}} & \cdots & \frac{\partial h_k}{\partial x_{t,n}} \end{bmatrix} \tag{2.35}$$

Thus, we can define the Extended Kalman Filter algorithm [30, p. 59]. It is almost identical to the Kalman Filter algorithm, and in fact the prediction and update steps continue to be performed on the same line numbers. The difference is that the linearized functions of g and h are used in lieu of the linear state predictions, in lines 1 and 5, and that the Jacobians are used instead of the state-space matrices. In terms of computational complexity, the Extended Kalman Filter is equal to the Kalman Filter, as they are very similar.

Algorithm 2.3 Extended Kalman Filter Algorithm

1: **function** *Extended_Kalman_Filter*(μ_{t-1},Σ_{t-1},u_t,z_t)
2: $\quad\overline{\mu}_t = g(u_t, \mu_{t-1})$
3: $\quad\overline{\Sigma}_t = G_t\Sigma_{t-1}G_t^T + R$
4: $\quad K_t = \overline{\Sigma}_t H_t^T \times (H_t\overline{\Sigma}_t H_t^T + Q)^{-1}$
5: $\quad\mu_t = \overline{\mu}_t + K_t(z_t - h(\overline{\mu}_t))$
6: $\quad\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t$
7: $\quad$**return** μ_t, Σ_t

2.1.3.6 Augmented Extended Kalman Filter

While the Extended Kalman Filter is able to produce good results in localization, it depends on the measurements quality and on the well-conditioning of the state transition models. The sensors' quality affects the filter's output, and thus the sensor covariance matrix

Q must be adjusted in order to determine the influence of a sensor on the corrected pose. If a sensor is too noisy (i.e. accelerometers), its covariance is high, and thus its influence in the filter will be small. Meanwhile, the state covariance matrix R also has to be fine-tuned in order to account for possible uncertainties in the state transition models.

Wrong parameters in modelling and sensor biases also affect the localization produced by the filter. For example, when using the differential drive model, we may erroneously measure a wheel's radius or the distance between the wheel centers, and thus our differential drive model will be slightly wrong. With sensor bias, it may shift the pose estimate constantly to an arbitrary direction. The *Augmented Extended Kalman Filter* (A-EKF) [18, 16] exploits the Kalman Filter's optimal estimation property to automatically estimate these uncertainties and then correct them in their respective models or states. More specifically, we *augment* the state-space of an Extended Filter to include additional states that represent the biases or external disturbances we want to estimate. These biases and disturbances are then subtracted from the measurements or states which they corrupt. To illustrate, consider the simple robot pose state-space seen in Equation (2.23). It is possible to increase the accuracy of this pose estimate by including additional states for sensor biases, $(\beta_1, \beta_2, \beta_3)$:

$$x_t = \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \\ x_{4,t} \\ x_{5,t} \\ x_{6,t} \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} \tag{2.36}$$

These new states are then added to the modelled measurement estimates in the update step. In other words, the biases are iteratively estimated and accounted for when correcting the belief, making for a more accurate estimate of the true pose. Its disadvantage is the increase in the state-space dimensionality, and thus an increase in computational load associated with matrix operations.

2.1.3.7 Particle Filter

As mentioned in Section 2.1.3, the localization problem becomes *global positioning* when the initial robot position is unknown. The Particle Filter (PF) [30, p. 96] solves this by representing the robot's possible position states (i.e. particles) randomly in a known space, and then determining which states are most likely correct through sensor measurements. Thus, even if the initial position is unknown, the states converge to the real robot position if the sensors are reliable. The algorithm for this technique is divided into three parts:

1. Prediction: The filter performs an estimation on the motion of the particles through the sensors available and the robot's movement model. This estimated motion is then applied to the particles, which in theory should be equivalent to the robot's motion.

2. Update: The filter gives each particle a probability of it coinciding with the robot's real position. The weight is a function of the difference between the robot's real sensor measurements and the modeled measurements of the particles (i.e. greater weights mean small differences between real and modelled measurements).

3. Resampling: The lowest weight particles are eliminated and the greatest weight particles are duplicated, in a way that the total initial number of particles is not altered. This step can be performed in a fixed interval (e.g. each 15 filter iterations) for reasons of computational cost, and because resampling on each iteration may introduce localization bias (i.e. the particles will always converge in the same way), and the filter then loses its ability to adapt to different conditions during execution.

On the prediction step, the movement model must account for uncertainty in the robot's motion. An example of this is as a robot moves forward, it will almost certainly not move in a straight line and will miss its target position with a degree of error. This uncertainty may be modeled through probability distributions, of which the parameters must be adjusted for the algorithm's correct operation.

The Particle Filter is a powerful method for absolute positioning, able to determine a robot's location with small error. However, a problem lies with its computational cost, which increases with the number of particles. The algorithm's complexity is $\mathcal{O}(M)$ (i.e. linear time), with M being the number of particles. The filter's computational cost comes when we consider that a great number of particles (typically in the order of thousands) must be used to determine localization reliably and with precision. Another disadvantage is the fact that most implementations require a known map of the environment in order to spread the particles and simulate their measurements, and this map is not always easily available.

2.1.4 Robot Operating System

Communication is an important part of a robotic system: The way in which its different modules communicate between themselves and with the user. One of the challenges in communication is due to the heterogeneous nature of a robot, as normally they are an amalgam of different processors. The *message-passing frameworks* alleviate this problem by implementing communication protocols that encapsulate data in standard message types, that will be decoded differently in each processor architecture. This allows for a message to be understood in the same manner across all modules and processors.

The *Robot Operating System* (ROS) [24] is currently the most popular framework in mobile robotics, based on the TCP/IP protocol. Apart from defining standard message types, it implements a topology for communication based on message pipelines called *topics*, which may be accessed by any module (in ROS terminology, any *node*). A node that receives messages from a topic is called a *subscriber*, while a node that pushes messages to a topic is called a *publisher*. These two modes are not mutually-exclusive, and a node may publish or subscribe to as many topics as it wants. Thus, communication in a robot using ROS does not happen directly between modules, but instead through the topics. Figure 2.15 illustrates this topology.

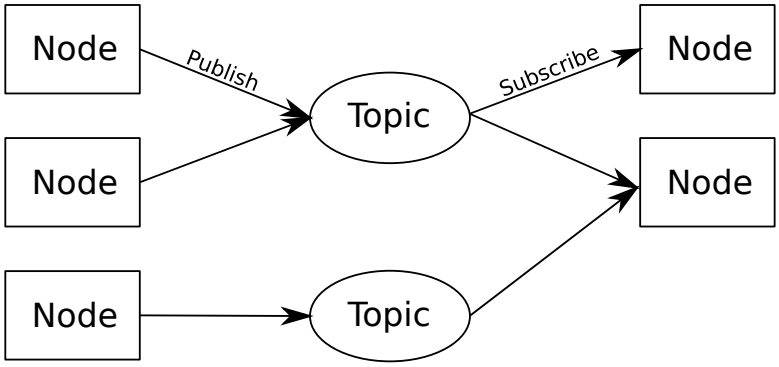


Figure 2.15: Message-passing topology of the Robot Operating System

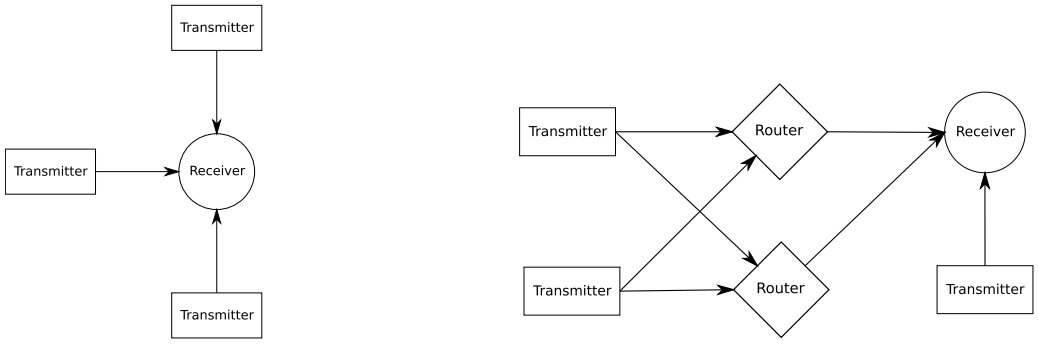
Apart from allowing for heterogeneous computing, ROS allows for distributed computing as well, through its communication topology. This is done via a "Network setup", in which a *master* device is specified to be the center of the ROS architecture in use. The master's IP is known to all other devices (*hosts*), and all the nodes in the ROS architecture communicate through the master. Thus, different ROS nodes can be executed on different hosts (and on the master itself), splitting the computational load of a robotic system between several devices. A disadvantage in this setup is that all devices must be connected to the same Wi-Fi network, and the operation range becomes limited to the low range of the Wi-Fi protocol. Another limitation is that when the link between master and host is severed, the whole ROS system shuts down and a full reboot is required, which is a problem in some applications (i.e. a boat getting stranded in the middle of a lake). Specific requirements of network bandwidth also apply, due to the nature of Wi-Fi communication.

2.2 Wireless Sensor Networks

In this project, the loss of signal strength with distance is used to gauge the range between Radio-Frequency receiver-transmitter pairs. More specifically, a receiver is placed on the robot and three transmitters are placed on fixed locations around the operating environment. A transmission is made for each pair and the power of the RF signal is measured

in the receiver. This power decreases as the electromagnetic wave travels, and it is possible to use the received power with mathematical models to estimate the distance between the receiver and transmitter. Three distances can be used for multi-lateration, such as with the GPS (that is, *beacon-based* localization).

Thus, in a sense, the receiver-transmitter pairs can be thought of as extra sensors on the robot. They have the advantage of not suffering from external disturbances common to other sensors used in localization, such as wheel slips in encoders and lighting changes in cameras. However, their distance models have uncertainties, and the received signal strength is extremely volatile, varying with many factors. To better understand these uncertainties and error sources, it is worthwhile to explore how the signal strength loss in regards to distance is modelled. The RF networks are normally called *Wireless Sensor Networks* (WSN), as they are used for wireless communication of sensor data in factories, hospitals and other environments. These networks may be structured or ad-hoc, such as star or mesh networks seen in Figures 2.16a and 2.16b respectively.



(a) Example of a star network

(b) Example of a mesh network

Figure 2.16: Examples of wireless network structures

2.2.1 Signal Strength Loss Models

As an electromagnetic wave travels through space, it loses power with distance. More specifically, a transmission with power p_t radiates uniformly in all directions, and a power density s can be described in regards to the distance d from the transmitter [4, p. 11]:

$$s = \frac{p_t}{4\pi d^2} \tag{2.37}$$

On the receiver side, the power of a signal arriving at the antenna can be described as [4, p. 14]:

$$p_r = s \times a_e \tag{2.38}$$

where a_e is the antenna's *aperture*, the effective receiving area of an antenna given by $\lambda^2/4\pi$ for an ideal loss-free antenna, where λ is the RF signal's wavelength. Substituting s by Equation (2.37) and a_e by the ideal antenna aperture, we have the relationship between transmitted and received power:

$$p_r = p_t \times \frac{\lambda^2}{(4\pi d)^2} \tag{2.39}$$

The ratio p_t/p_r is called the *transmission loss*:

$$\frac{p_t}{p_r} = \frac{(4\pi d)^2}{\lambda^2} \tag{2.40}$$

Equation (2.40) models the power loss of a signal travelling in free-space, and as such is referred to as the *Free-Space Path Loss* (FSPL) model. It is normally measured in decibels, and thus expressed in logarithmic terms [4, p. 15]:

$$FSPL = 10log_{10}\left(\frac{p_t}{p_r}\right) = 20log_{10}\left(\frac{4\pi d}{\lambda}\right) \tag{2.41}$$

This assumes that the transmission environment is a loss-less vacuum, where no other influences to signal strength are present. However, all radio waves are affected by the Earth and its atmosphere when operating on air, even in high altitudes (i.e. ionosphere, 1000 km) [4, p. 4]. Atmospheric pressure, temperature and humidity all have effects on RF signals. In addition, diffraction, reflection and scatter of a signal occur due to environmental obstacles (i.e. mountains, buildings), causing multi-path interference [4, p. 5].

Another important consideration when discussing path loss is with the *Fresnel Zone* in line-of-sight transmissions [4, p. 137]. This zone is described by an ellipsoid with the transmitter and receiver antennas at the extremes, as seen in Figure 2.17. The ellipsoid's smallest radius describes the largest area that a signal occupies vertically in space, and a large enough radius may intersect with obstacles, causing multi-path interference due to reflection. The Fresnel Zone radius is formulated as:

$$F_r = \frac{1}{2}\sqrt{\frac{cD}{f}} \tag{2.42}$$

where D is the distance between the antennas. Avoiding obstacles in this radius is called *Fresnel Clearance*, where the RF engineer must design the communication link as such that the Fresnel Zone is clear of obstacles.

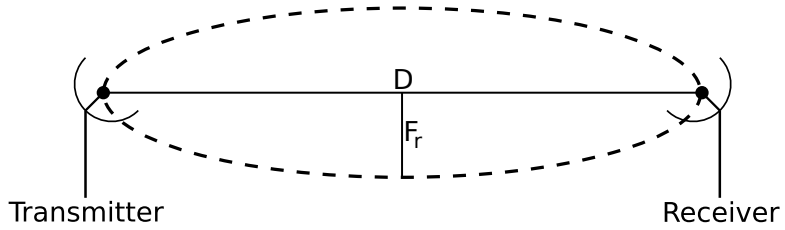


Figure 2.17: Fresnel Zone between two antennas

2.2.2 Wireless Sensor Network Localization

The area of Wireless Sensor Networks consists in the study of wireless networks for the over-the-air communication of sensor data. In this area, the field of Wireless Sensor Network Localization explores the localization problem applied to WSNs: The discovery of an unknown node's position in space. To illustrate, consider the star network topology presented in Figure 2.16a. The (x,y) positions of all the transmitter nodes are known, and we want to find the position of the receiver node. This is done by exploiting the properties of signal propagation between the nodes to measure distances, and then using those distances in typical localization methods such as multi-lateration [13, p. 61]. There are mainly two properties of a signal which can be used for distance estimation: *Time of Flight* and *Received Signal Strength*.

The latter consists of gauging the power loss between transmitter and receiver, and then using this loss with path-loss models to estimate the distances. In the case of the Free-Space Path Loss model, it is possible to solve Equation (2.41) for the distance:

$$d_{FSPL} = \sqrt{\frac{\lambda^2 \times 10^{\frac{FSPL}{10}}}{(4\pi)^2}} \tag{2.43}$$

The distances then may be used with various positioning methods. An advantage of using the received signal strength is that no complex additional hardware is needed, as most RF modules come with circuitry capable of measuring the received power. However, this measurement is very noisy and highly uncertain, due to the many external disturbances and environmental factors previously discussed.

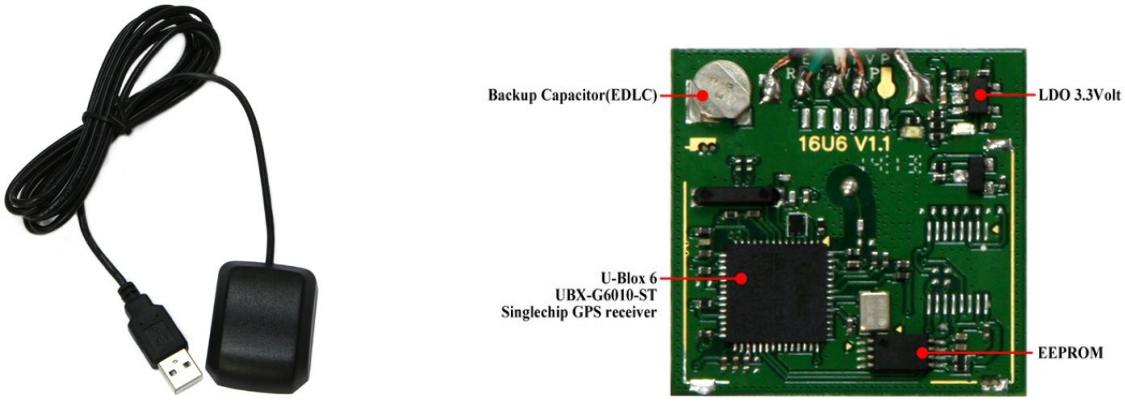
3. MATERIALS AND METHODS

This chapter discusses the software and hardware resources used in the course of this project, the methodology used in data acquisition and analysis, and the implementation of the robot architecture and localization methods previously discussed. Section 3.1 describes the hardware and software resources used in the project. Section 3.2 details the practical implementation of the robot's hardware and software architectures, as well as the implementations of the Extended Kalman Filter and Augmented Extended Kalman Filter. Finally, Section 3.3 details the experimental setup used to collect the datasets, including discussion of the test environments, and the methodologies for data acquisition, data treatment and sensor validation.

3.1 Used Resources

3.1.1 Sensors

The sensors used in this project were the mobile base's quadrature optical encoders, an USB GPS module, a CHRobotics UM6 Inertial Measurement Unit and an Emlid Reach RTK GPS module. The USB GPS module (Figure 3.1) is a breakout board based on U-Blox's *UBX-G6010-ST*[1], a GPS chipset with accuracy of 2.5 meters and convergence time of approximately 30 seconds. The breakout board provides connection to computers via an USB-A cable, a 3.3V voltage regulator which supplies power to the chipset, an energy-backup capacitor and an EEPROM for the chipset to store information.



(a) (b)

Figure 3.1: The USB GPS module and breakout board

[1] https://www.u-blox.com/sites/default/files/products/documents/UBX-G6010-ST_ProductSummary_%28GPS.G6-HW-09001%29.pdf

The *UM6*[2] is a 3.3V powered IMU (Figure 3.2) with 9 Degrees-of-Freedom, capable of measuring every 2 ms (500 Hz). It contains accelerometers, gyroscopes and a magnetometer (digital compass).It provides data access via Serial TTL or SPI, as well as interfaces for temperature compensation (with an internal thermometer), gyroscope bias correction and GPS module integration.

Figure 3.2: The UM6 Inertial Measurement Unit

The *Emlid Reach* (Figure 3.3) is a Real-Time Kinematics GPS module based on the U-Blox NEO M8T, a GNSS chipset capable of monitoring up to 3 satellite constellations concurrently [12]. This chipset is integrated with an Intel Edison processor, which provides Serial, SPI and Wi-Fi communication, as well as a web-based graphical user interface named ReachView and non-volatile memory for logging trajectories. There are two main modes of operation for the Reach: Rover, when it is being used for real-time positioning, and Base, when it is used as a reference correction station. As such, the RTK configuration described in Section 2.1.2.4 can be performed using two Reach modules, one as the rover and one as the base. The rover can be further configured to work in static of kinematic mode, with accuracies of 5 and 7 mm respectively.

Figure 3.3: The Emlid Reach RTK GPS module

Four *quadrature optical encoders* are used in the mobile base, one encoder for each motor shaft. They consist of a metal disk with one circular row of slits and two pairs of light emitter-detectors which are positioned as described in Section 2.1.2.1, creating the A

[2]http://www.chrobotics.com/docs/UM6_datasheet.pdf

and B channels. The encoders have 500 slits per revolution, meaning that they are capable of measuring rotations as small as $2\pi/500 = 0.01$ radians. They come built-in on the motor casings, as seen in Figure 3.4, and their procedence is unknown since no part number or brand information can be found.

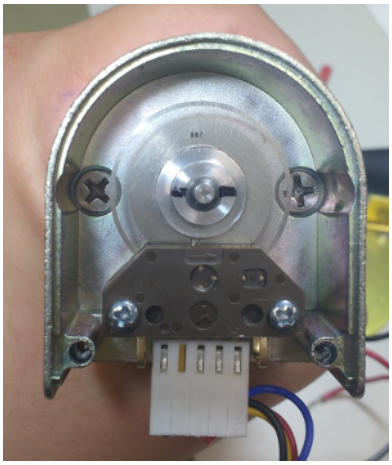


Figure 3.4: The quadrature encoder used in the mobile base

3.1.2 Radio-Frequency Module

The Radio-Frequency modules used in this project were four XBee PRO S2C, with *Reverse-Polarity SubMiniature A* (RP-SMA) connectors and omnidirectional antennas, as seen in Figure 3.5. The XBee is a family of 2.45 GHz RF modules, manufactured by Digi. Each model is based on different RF chipsets, but they all come pre-configured to work in the same frequency channel, and thus can be used out-of-the-box.

There were four key requirements that were considered when selecting the appropriate radio module for this project. The first was the ability to measure the received signal strength and to make these measurements available. The XBee modules present the measured power in terms of a decibel scale called Received Signal Strength Indicator (RSSI), with 1 dB resolution, which is available on all XBees through a specific command or embedded in the frame of a received data packet, among others.

Second, there must be a way of knowing where the received signal is coming from. As each transmitter has a different known position, knowing the source of the RSSI measurements is imperative for localization. This is possible in the XBees through *addressing*, where each radio can be assigned a unique 16-bit address.

Third, the radio must have an adequate range of communication, with at least 30 meters in an urban environment. This range was deduced from the largest length of HRATC's minefield, which was 20 meters plus a 10 meters margin. The communication radius depends on the transmission power and the antenna gain, and in the case of XBees,

Figure 3.5: The XBee PRO S2C module with an RP-SMA antenna connector

some models proved inadequate because of low power and low gain antennas. The XBees PRO are more potent versions of the mainstream models, with transmission power levels up to 18 dB, which translates to a range of 90 meters in indoor/urban environments [8].

Fourth, there should be appropriate hardware to mitigate the effects of noise and uncertainty in the received signal strength measurements. These effects are mostly related to the environmental conditions discussed in Section 2.2.1, such as pressure, temperature and interference from other devices. The receiver antenna's gain and aperture can help to minimize the variability in the RSSI measurements, and thus a radio module with an appropriate antenna was necessary. As seen in [15], the use of a long omnidirectional antenna with an absorbing plate at its base greatly improves the uniqueness of the received signal strength. It is possible to solder an RP-SMA connector in an XBee PRO S2C model, and then use an omnidirectional antenna.

Filtering by these criteria, the XBee PRO S2C (Series 2) model was selected, bought with RP-SMA connectors already soldered and 2 dBi gain omnidirectional antennas. A rectangular metal plate, measuring 7×10 centimeters, was added to the base of the receiver's antenna as the absorbing plate. Compared to [15], where the authors thoroughly designed an antenna with a specific absorbing plate, the plate proposed here is much simpler (i.e. a commonly available metal slab), and thus can be easily implemented. This setup can be seen in Section 3.2.1, along with the complete robot. Although the XBees were selected here, we have no evidence to suggest that this project would not work if other RF modules were to be used, as long as they fit the four main criteria.

3.1.3 Robotic Platform

An Adept-MobileRobots' Pioneer 3-AT mobile base was used in this project, depicted in Figure 2.5a of Chapter 2. This base is an all-terrain vehicle, constructed with outdoor operation in mind. It is capable of traversing rough terrain such as uneven asphalt, sand or dirt [1]. In total, the mobile base weighs 12 kg and supports payloads from 5 to 12 kg depending on the terrain: 5 kg on asphalt, 10 kg in grass and 12 kg in tile floors.

The wheels are driven by four 12V motors in a scheme called *skid-steering*, which is closely related to the differential drive model. In this scheme, the wheels in each pair move together, independent from the opposite pair. This means that rotation speed can be controlled independently for the left and right side of the robot, essentially reducing skid-steering to a differential drive model. The advantage of using this scheme is the added stability and traction power of the two extra wheels.

In regards to power, the base counts with up to 3 Lead-Acid batteries (7.2 Ah). This amounts to between 2 and 4 hours of continued use, a time which is reduced with the addition of extra sensors. There are 5 and 12V switched power supplies available for these sensors. For processing, the P3AT has a micro-controller with pre-loaded firmware.

3.1.4 Software

The Robot Operating System was ubiquitous in this project, as it provides seamless communication and remote robot control, and as the mobile base and almost all sensors have drivers in this framework. The sensors without ROS drivers were the XBee module and RTK GPS: A ROS package was built to interface with the XBee, while the RTK was not integrated in robot software architecture. The ROS packages used in the robot are listed below:

- *RosAria*: A ROS implementation of Adept-MobileRobots' ARIA software, for control of the Pioneer 3-AT and data acquisition of its sensors.
- *rosaria_client*: A service package for RosAria, it provides nodes for tele-operation of the P3AT.
- *um6*: A ROS driver for the UM6 Inertial Measurement Unit.
- *nmea_navsat_driver*: A ROS implementation for a generic parser of NMEA strings, a format for providing GPS position and velocity estimates.
- *xbee_loc*: A custom-built package for this project, it integrates XBee modules into the robot architecture. Details of its implementation will be discussed in Section 3.2.2.

Apart from the ROS robot packages, the *rosbag* command was also used. It allows for the recording of some or all topics currently running in a ROS session, saving them in a ".bag" extension file. To manipulate bag files, the rosbag Python API[3] was used, and the ROS message filter API[4] was used for data synchronization. Other programs employed here include the X-CTU[5], RTKLIB[6] and MATLAB[7]. The first was used to configure the XBee radios to operate according to specific conditions, which will be discussed in Section 3.2.1. RTKLIB was used for post-processing of the ground-truth trajectories obtained with the RTK GPS, and MATLAB was used for sensor validation, dataset treatment and trajectory estimation with the EKF and A-EKF algorithms.

3.2 Implementation

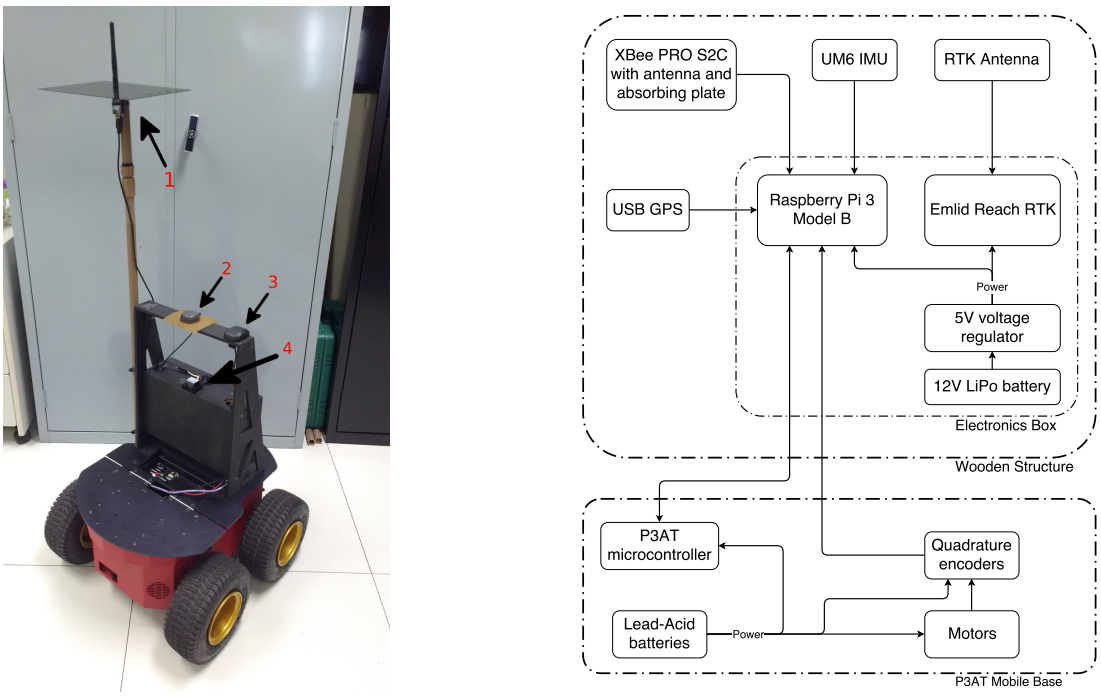
This Section discusses the steps taken towards building the complete robot, the implementation of the custom ROS package for XBee integration, and the formulation of the Extended and Augmented Extended Kalman Filters. The source code for the ROS package and Kalman Filters can be found in the project's GitHub repository[8].

3.2.1 Robot Construction and Software Architecture

The robot used in this project is a simplified version of the one used in the HRATC competition. It is a combination of the sensors and the Pioneer 3-AT mobile base, along with a Raspberry Pi 3 board[9] as the main processing unit, and a 12V Lithium-Polymer (LiPo) battery for powering the sensors and electronics, thus increasing the mobile base's autonomy. The resulting robot, nicknamed "*Trouble*", can be seen in Figure 3.6a with its relevant parts numbered, along with a connection diagram in Figure 3.6b. is then controlled and monitored in a Dell XPS laptop through ROS.

Table 3.1 shows the individual sensor and electronics prices, and the total cost of the localization system proposed here. The price of the RTK GPS is not shown, as it is not a part of the proposed localization system, and the IMU price presented is for the UM7, as the UM6 was discontinued. As the 3 external RF beacons are integral to this project's localization system, the cost of their parts is also presented.

[3] http://wiki.ros.org/rosbag/Code%20API
[4] http://docs.ros.org/api/message_filters/html/python/
[5] https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu
[6] http://www.rtklib.com/
[7] https://www.mathworks.com/products/matlab.html
[8] https://github.com/rgmaidana/rssi-localization-hratc
[9] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/



(a) (b)

Figure 3.6: The Trouble Robot and its composing parts diagram

Table 3.1: Sensors and electronics cost for the localization system

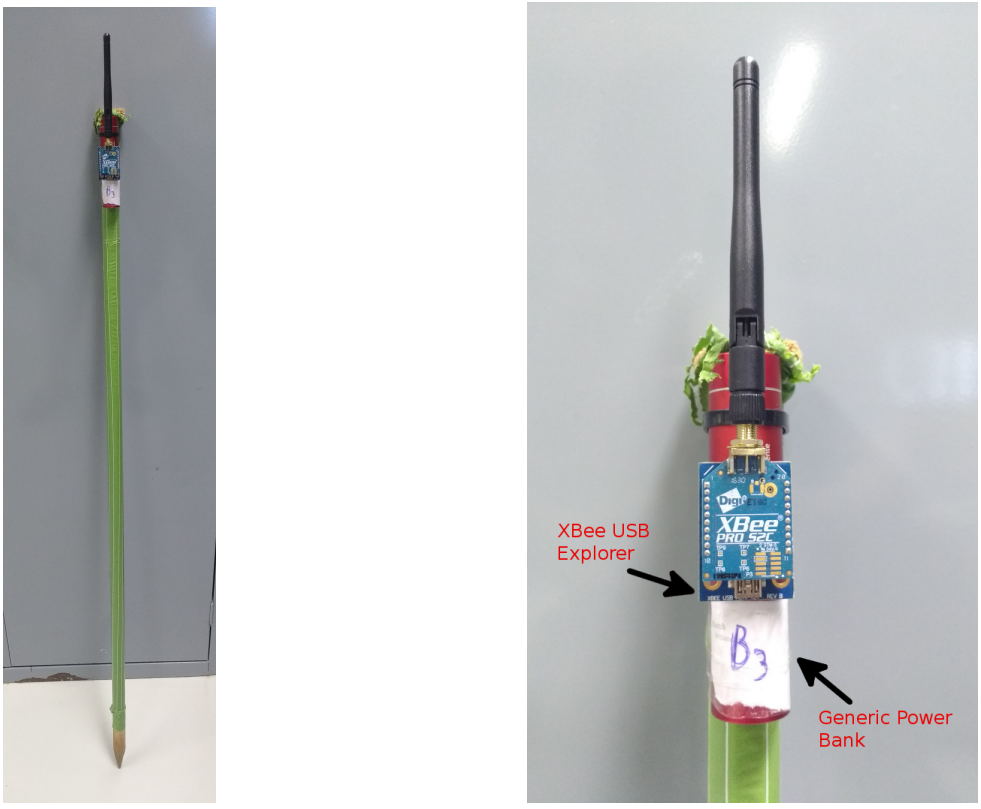
Part Description	Cost (U$)
UM7 Inertial Measurement Unit[10]	164.95
USB GPS Receiver[11]	21.95
Raspberry Pi 3 Model B[12]	35.00
4x XBee PRO S2C Module with RP-SMA connector[13]	114.00
4x 2-dBi gain Omnidirectional Antenna[14]	17.60
FTDI-to-USB Adapter[15]	12.90
4x XBee Explorer breakout board[16]	99.80
Total	**466.20**

In regards to software, the robot is fully operated via ROS Kinetic [17]. The network setup is performed between the Raspberry Pi (master) and a Dell XPS notebook (host), using a 4G Hotspot Wi-Fi network. An active internet connection was required, as the Reach receives corrections through the NTRIP protocol.

[10] https://www.pololu.com/product/2764
[11] https://ameridroid.com/products/usb-gps-module
[12] https://www.adafruit.com/product/3055
[13] https://www.arrow.com/en/products/xbp24cdmsit-001/digi-international
[14] https://www.amazon.com/2-4GHz-RP-SMA-Rubber-Duck-Antenna/dp/B0093SA2EY
[15] https://www.dfrobot.com/product-147.html
[16] https://www.sparkfun.com/products/11812
[17] http://wiki.ros.org/kinetic

3.2.2 Radios Configuration and Integration to the Robot

To understand the configuration of the radios and the design of the *xbee_loc* package, the experimental setup must be briefly discussed. The main objective of this project is to perform mobile robot localization using the distances from a receiver on the robot to three transmitters in distinct coordinates on an environment. In localization, landmarks are usually called *beacons*, so let the three transmitters be designated as B_1, B_2 and B_3, and the receiver be designated as the *target* to be localized, T_0. The beacons are constructed by placing the radios on top of wooden stakes with 1.3 meters of height, once again for Fresnel Zone clearance. An example can be seen in Figure 3.7, which shows beacon B_3. The radios themselves, also with 2 dBi omnidirectional antennas, are mounted in XBee Explorer boards and powered via USB through generic power banks.



(a) (b)

Figure 3.7: The B_3 Radio-Frequency localization beacon

To be able to communicate with each other, the beacons and target need to be configured to work in the same RF network. This can be done via the X-CTU program, with the XBees connected to a computer through USB and the XBee Explorers. The parameters changed for the radios are listed below, and all the other parameters are left in their default configurations.

- *Coordinator Enable (CE)*: Specifies if the node is a coordinator or not. In this case, T_0 is set as the coordinator, while the beacons are configured as routers.
- *Personal Area Network ID (ID)*: Also known as PAN ID, this parameter specifies an identifier so that a network may be formed by all nodes with the same ID. This can be any positive 16-bit hexadecimal value, and in this case it is set to "F1A5" in all four radios.
- *Power Level (PL)*: Specifies the transmission power of the radio, in the PRO version ranging from -3 to 15 dB. It is a discrete scale from 0 to 3 indicating the power levels, in which 0 is the lowest and 3 is the highest. The lowest level is selected in all radios, as higher levels result in larger power density areas and consequently less sensitivity to distance changes.
- *Power Mode (PM)*: Enables and disables "Boost Mode", which increases receiver sensitivity by 2 dB and transmitter power by 3 dB. It is disabled in the radios for the same reason as with the Power Level.
- *API Enable (API)*: Enables the Application Programming Interface mode, in which the data received is encapsulated in a structured frame, containing source and destination addresses as well as the received strength of any transmission. In this mode, it is also possible to send remote commands from the target to the beacons, and thus read any register or reconfigure them over-the-air. All radios are configured to work in API mode.

Another interesting parameter to note is the "DB", which contains the RSSI value of the last transmission received. Thus, with this parameter and the radios in API mode, we can define a basic work-flow for obtaining the three received strength values from the beacons:

1. T_0 sends some data to B_1 (i.e. character "A") to update the DB register.
2. T_0 sends a remote command to B_1, requesting the value of the DB register.
3. B_1 responds with the last RSSI in decibels.
4. T_0 stores this value for B_1.
5. Repeat steps 1-4 for B_2 and B_3.

To do this, the beacons' addresses must be known, as data transmissions and commands are sent directly to each beacon, and all beacons must be within the communication range. This work-flow is implemented in the "xbee_loc" ROS package, written in the Python language.

3.2.3 Extended Kalman Filter

Section 2.1.3.5 reviewed the theory behind the Extended Kalman Filter, describing it in general terms. Here we discuss the practical formulation and implementation of the filter, specifically in the case of mobile robot localization. The robot position and orientation are approximated through the state models, the sensors model, and by fusing the three received signal strengths, odometry and GPS. Thus, the first step is to model the state-space to be estimated, which in this case is the robot's pose $\xi = [x, y, \theta]$. We can define this state-space as being the three coordinates as well as the velocities in x and y, v_x and v_y:

$$x_t = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ \theta \end{bmatrix} = \begin{bmatrix} x \\ v_x \\ y \\ v_y \\ \theta \end{bmatrix} \tag{3.1}$$

As seen in Section 2.1.3.5, the state-transition functions are given by the states' derivatives:

$$\dot{x}_t = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \end{bmatrix} = \begin{bmatrix} x_2 \\ \ddot{x} \\ x_4 \\ \ddot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} x_2 \\ a_x \\ x_4 \\ a_y \\ \omega \end{bmatrix} \tag{3.2}$$

where a_x and a_y are the accelerations in x and y of the global coordinate frame, and ω is the robot's angular velocity. The velocity motion model can be used to represent the next state accelerations, as seen in Section 2.1.1.2 (page 34), which justifies the use of the Extended Kalman Filter, as it is a non-linear model. Equation (2.10) presents the global pose velocities in terms of the robot's frontal, lateral and angular velocities. Differentiating this expression again gives us the x and y accelerations in the global frame:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = R(\theta)^{-1} \begin{bmatrix} a_{front} \\ a_{lat} \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} a_{front} \\ a_{lat} \end{bmatrix} \tag{3.3}$$

where a_{front} and a_{lat} are the accelerations in the robot local frame, the latter which should always be zero as the robot can not move sideways. Along with the angular velocity ω, they describe the filter's input vector, $u = [a_{front}, a_{lat}, \omega]$. The state-transition functions in Equation (3.2) can then be rewritten to include the expressions for the global accelerations in x and y, with the input vector:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \end{bmatrix} = \begin{bmatrix} x_2 \\ a_{front}cos(x_5) - a_{lat}sin(x_5) \\ x_4 \\ a_{front}sin(x_5) + a_{lat}cos(x_5) \\ \omega \end{bmatrix} = \begin{bmatrix} x_2 \\ u_1 cos(x_5) - u_2 sin(x_5) \\ x_4 \\ u_1 sin(x_5) + u_2 cos(x_5) \\ u_3 \end{bmatrix} \tag{3.4}$$

However, as the filter is to be implemented digitally, the discrete-time representation must be used. The state-transition functions in discrete-time are given by the finite differences in the states. Using the Euler-Forward discretization, these differences are given by:

$$\Delta(x_i) = \frac{x_{t+1,i} - x_{t,i}}{T} \tag{3.5}$$

where T is a constant sampling time and $x_{t+1,i}$ is the next i-th state. We can solve this equation for $x_{t+1,i}$, obtaining the state-transition functions in discrete-time:

$$x_{t+1,i} = x_{t,i} + \Delta(x_i)T \tag{3.6}$$

This can be considered to be numerical derivation, and as such $\Delta(x_i)$ is the discrete derivative of the states. In the case of the robot pose, the derivative for the three coordinates are their velocities, linear for (x,y) and angular for θ. We can apply Equation (3.6) to write the discretized state-transition functions as:

$$x_{t+1} = \begin{bmatrix} x_{t+1,1} \\ x_{t+1,2} \\ x_{t+1,3} \\ x_{t+1,4} \\ x_{t+1,5} \end{bmatrix} = \begin{bmatrix} x_1 + x_2 T \\ x_2 + [u_1 cos(x_5) - u_2 sin(x_5)]T \\ x_3 + x_4 T \\ x_4 + [u_1 sin(x_5) + u_2 cos(x_5)]T \\ x_5 + u_3 T \end{bmatrix} \tag{3.7}$$

where $x_1 = x_{1,t}$, for example, for clarity. Expressed as a matrix multiplication, Equation (3.7) is simplified to:

$$x_{t+1} = x + f \times T \tag{3.8}$$

where $x = [x_1, x_2, x_3, x_4, x_5]^T$ and f is a function of the states and the input vector u:

$$f(x_t, u_t) = \begin{bmatrix} x_2 \\ u_1 cos(x_5) - u_2 sin(x_5) \\ x_4 \\ u_1 sin(x_5) + u_2 cos(x_5) \\ u_3 \end{bmatrix} \tag{3.9}$$

This equates to the first step of the prediction stage in the Extended Kalman Filter. In order to predict the covariance matrix, we need to calculate the states' Jacobian matrix. To do this, it is easier to define the Jacobian as a zero matrix and calculate the non-zero terms with the state-transition functions in Equation (3.7) and the Jacobian definition in Section 2.1.3.5 (page 53). Once again for clarity, let $\dot{x}_i = x_{i,t+1}$ (i = 1..5). Specifically, the non-zero partial derivatives are:

$$\frac{\partial \dot{x}_1}{\partial x_1} = \frac{\partial \dot{x}_2}{\partial x_2} = \frac{\partial \dot{x}_3}{\partial x_3} = \frac{\partial \dot{x}_4}{\partial x_4} = \frac{\partial \dot{x}_5}{\partial x_5} = 1 \tag{3.10}$$

$$\frac{\partial \dot{x}_1}{\partial x_2} = \frac{\partial \dot{x}_3}{\partial x_4} = T \tag{3.11}$$

$$\frac{\partial \dot{x}_2}{\partial x_5} = -u_1 sin(x_5) - u_2 cos(x_5) \tag{3.12}$$

$$\frac{\partial \dot{x}_4}{\partial x_5} = u_1 cos(x_5) - u_2 sin(x_5) \tag{3.13}$$

With this, we can assemble the Jacobian matrix for the states, a 5×5 matrix with a row for each transition function and a column for each state:

$$G_t = \begin{bmatrix} 1 & T & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \partial \dot{x}_2 / \partial x_5 \\ 0 & 0 & 1 & T & 0 \\ 0 & 0 & 0 & 1 & \partial \dot{x}_4 / \partial x_5 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

To implement the update stage, the measurement model vector must be defined and the measurements Jacobian must be calculated. First, let h be the measurement model vector, composed of the three beacons' received strength, the odometry velocities in x and y, and the (x,y) coordinates reported by the USB GPS:

$$h = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \end{bmatrix} = \begin{bmatrix} B_{1,rssi} \\ B_{2,rssi} \\ B_{3,rssi} \\ \dot{x}_{odom} \\ \dot{y}_{odom} \\ x_{gps} \\ y_{gps} \end{bmatrix} \tag{3.15}$$

The received strength model used was the Free-Space Path Loss from Equation (2.41), page 59, chosen for its simplicity:

$$B_{i,rssi} = 20 log_{10}(\frac{4\pi d_i}{\lambda}) \tag{3.16}$$

where $B_{i,rssi}$ is the i-th beacon's signal strength and λ is the wavelength (for a 2.45 GHz signal, λ = lightspeed / frequency = 0.12 m). The distance to the i-th beacon is d_i, modelled with the Euclidean norm using the target's estimated coordinates (x,y = x_1, x_3) and the beacon's (x,y) known coordinates, ($B_{i,x}$, $B_{i,y}$):

$$d_i = \sqrt{(x_1 - B_{i,x})^2 + (x_3 - B_{i,y})^2} \tag{3.17}$$

As states 1 and 3 already estimate the (x,y) coordinates for the robot, the models for the odometry and GPS coordinates are simply states 1 and 3 themselves:

$$x_{odom} = x_{gps} = x_1 \tag{3.18}$$

$$y_{gps} = y_{gps} = x_3 \tag{3.19}$$

With h defined, we can calculate the measurements' Jacobian, which will be a 7×5 matrix. As before, it is more efficient to define it as a zero matrix and calculate the positions where it is not zero:

$$\frac{\partial h_1}{\partial x_1} = \frac{20(x_1 - B_{1,x})}{log_e(10)[(x_1 - B_{1,x})^2 + (x_3 - B_{1,y})^2]} \tag{3.20}$$

$$\frac{\partial h_1}{\partial x_3} = \frac{20(x_3 - B_{1,y})}{log_e(10)[(x_1 - B_{1,x})^2 + (x_3 - B_{1,y})^2]} \tag{3.21}$$

$$\frac{\partial h_2}{\partial x_1} = \frac{20(x_1 - B_{2,x})}{log_e(10)[(x_1 - B_{2,x})^2 + (x_3 - B_{2,y})^2]} \tag{3.22}$$

$$\frac{\partial h_2}{\partial x_3} = \frac{20(x_3 - B_{2,y})}{log_e(10)[(x_1 - B_{2,x})^2 + (x_3 - B_{2,y})^2]} \tag{3.23}$$

$$\frac{\partial h_3}{\partial x_1} = \frac{20(x_1 - B_{3,x})}{log_e(10)[(x_1 - B_{3,x})^2 + (x_3 - B_{3,y})^2]} \tag{3.24}$$

$$\frac{\partial h_3}{\partial x_3} = \frac{20(x_3 - B_{3,y})}{log_e(10)[(x_1 - B_{3,x})^2 + (x_3 - B_{3,y})^2]} \tag{3.25}$$

$$\frac{\partial h_4}{\partial x_2} = \frac{\partial h_5}{\partial x_4} = \frac{\partial h_6}{\partial x_1} = \frac{\partial h_7}{\partial x_3} = 1 \tag{3.26}$$

And the Jacobian matrix:

$$H_t = \begin{bmatrix} \partial h_1/\partial x_1 & 0 & \partial h_1/\partial x_3 & 0 & 0 \\ \partial h_2/\partial x_1 & 0 & \partial h_2/\partial x_3 & 0 & 0 \\ \partial h_3/\partial x_1 & 0 & \partial h_3/\partial x_3 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.27}$$

The covariance and uncertainty matrices Q and R, defined in Section 2.1.3.4 (page 49), must be tuned manually by observing the behavior of the filter. The values for these matrices will be discussed in Chapter 4. Finally, the Extended Kalman Filter in Algorithm 2.3 can be implemented.

3.2.4 Augmented Extended Kalman Filter

The idea to use the Augmented EKF came while validating the received signal strength measurements. It was verified that it suffers a great deal with noise and is heavily attenuated relative to the Free-Space Path Loss model. We can use noisy measurements by increasing their associated covariance, and thus reducing their effect on the states' estimation. However, the attenuation equates to a bias in the measurements that always shifts the pose estimation, no matter how large the covariance is. Once again, this comes from un-modelled environmental uncertainties and constructive parameters of each radio, as the biases are different for each beacon. If these biases could be measured, they could be corrected, resulting in more accurate models.

Luckily, the Extended Kalman Filter is a near-optimal estimator, and as such we can estimate the biases using the filter itself and then add them to the relevant measurement models. As seen in Section 2.1.3.6, the Augmented Extended Kalman Filter is used when external disturbances and uncertainties have to be accounted for. In this case, the uncertainties to be estimated and modelled are the RF beacon biases. This is done by augmenting the EKF state-space with three additional states β_1, β_2 and β_3, each representing a beacon bias:

$$x_t = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ \theta \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} \tag{3.28}$$

Let us consider that these biases are mostly constant, and thus their derivatives are zero. In discrete time, the new state transition functions are:

$$x_{t+1} = \begin{bmatrix} x_1 + x_2 T \\ x_2 + [u_1 cos(x_5) - u_2 sin(x_5)]T \\ x_3 + x_4 T \\ x_4 + [u_1 sin(x_5) + u_2 cos(x_5)]T \\ x_5 + u_3 T \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} \tag{3.29}$$

The states Jacobian is also modified, now being an 8×8 matrix:

$$G_t = \begin{bmatrix} 1 & T & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \partial \dot{x}_2 / \partial x_5 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \partial \dot{x}_4 / \partial x_5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.30}$$

The bias states x_6, x_7 and x_8 are then incorporated into the measurement models for the three received signal strengths:

$$B_{1,rssi} = 20 log_{10}(\frac{4\pi d_1}{\lambda}) + x_6 \tag{3.31}$$

$$B_{2,rssi} = 20 log_{10}(\frac{4\pi d_2}{\lambda}) + x_7 \tag{3.32}$$

$$B_{3,rssi} = 20 log_{10}(\frac{4\pi d_3}{\lambda}) + x_8 \tag{3.33}$$

Naturally, the measurements Jacobian changes too, now a 7×8 matrix:

$$H_t = \begin{bmatrix} \partial h_1/\partial x_1 & 0 & \partial h_1/\partial x_3 & 0 & 0 & 1 & 0 & 0 \\ \partial h_2/\partial x_1 & 0 & \partial h_2/\partial x_3 & 0 & 0 & 0 & 1 & 0 \\ \partial h_3/\partial x_1 & 0 & \partial h_3/\partial x_3 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.34}$$

The rest of the measurement vector (odometry and GPS) remains the same. The EKF is implemented as usual with the new state-space, RSSI measurement models and Jacobians.

3.3 Experiments

The first experiment was concerned with verifying is the sensors were suitable for the proposed localization solution (i.e. sensor validation), and with evaluating the accuracy of localization. To do this, several datasets were collected in test runs, where at least three distinct datasets were collected in each run. This triplicate setup was in order to have enough data to draw statistical conclusions on the results. More specifically, the experiment objectives were:

1. To acquire, verify and if possible improve the accuracy of the ground-truth trajectories.
2. To validate the robot sensors, analyze their levels of noise and uncertainty, and verify if they are suited to be used in the Kalman Filters.
3. To obtain trajectories with the EKF and A-EKF using the verified sensor measurements, and compare the trajectories with the ground-truth.
4. To observe the effects of the environment on the received strength measurements, with and without the hardware changes proposed in [15].

Objectives 1-3 could be achieved with the same datasets, recorded in the same test run, with a predefined trajectory. Due to objective 4, another run was performed where two sets of the triplicate setup had to be collected, one with the absorbing plate and one without.

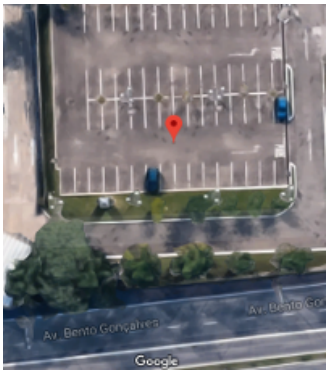
The objective for the second experiment was to evaluate if the proposed solution would work in a GPS-deprived scenario, similar to the HRATC arena. To do this, two sets of

experimental runs were performed in different areas, once again recording datasets in triplicate. In this Section, the details pertaining to these experiments are explained. Specifically, the test areas are discussed, as well as the methodologies for data collection, data treatment and sensor validation. The datasets for both experiments can be found on Zenodo[18].

3.3.1 Test Areas

The area selected to perform the first experiment was a parking lot[19] at PUCRS, with no parked cars, which can be seen in Figure 3.8. The ideal scenario to test the localization systems in the absence of GPS would be the HRATC arena, an area packed with trees and buildings. However, at the same time, satellite line-of-sight was a requirement since an RTK GPS was being used to obtain the ground-truth. The parking lot was the second best scenario, as it was an outdoor urban environment with a clear sky-view.

(a) Photo of the parking lot



(b) Google Maps image of the parking lot

Figure 3.8: The parking lot test area

The areas for the second experiment were the parking lot and a garden with unpaved ground, tall trees and a nearby building, also at PUCRS, which can be seen in Figure 3.9. As there is no GPS availability due to the trees and building, the ground-truth could not be recorded with the RTK GPS. Instead, a straight-line was marked on the ground, and the robot was carefully driven on top of it. The ground-truth was later determined through various measurements of the triangular beacon setup, which will be discussed in Section 3.3.3. In order to compare the results with the localization system in the more controlled environment of the parking lot, the experiment had to be performed on this area as well, with the same beacon setup and ground-truth.

[18] https://zenodo.org/record/1219249
[19] Coordinates: -30.061877, -51.175781

Figure 3.9: The garden test area

3.3.2 Methodology for Dataset Collection

In the first experiment, to collect the datasets with and without the absorbing plate, the Trouble robot with all sensors and the Dell Laptop were taken to the parking lot. Before collection, some preparatory steps were taken. The robot was put in an arbitrary initial position from where a predefined trajectory was performed, seen in Figure 3.10, along with the vertical and horizontal distances which repeat throughout the trajectory. The total distance traveled in the predefined trajectory was 48.78 m, in an area of 111.5 m^2. Figure 3.10 also shows the position of the RF beacons, which were powered on and put in a triangular setup around the field. The beacons were strategically placed near landmarks visible in mapping services such as Google Earth, in order to estimate their positions through the maps. The distances between them were measured as $B_{1,2}$ = 16.9 m, $B_{2,3}$ = 17.5 m and $B_{3,1}$ = 16.1 m. The beacons connect with the robot's XBee, which creates a ZigBee network as soon as it is turned on. The robot, Raspberry Pi and Emlid Reach were powered, and the latter two connected to a 3G Wi-Fi network from which the robot's topics were monitored and the RTK NTRIP corrections were obtained. After these preliminary steps, the robot was tele-operated according to the predefined trajectory and its data was recorded in the laptop.

As previously stated, at least three datasets were collected for the RF receiver with and without the absorbing plate. The recorded sensors were the IMU, encoders' odometry, USB GPS and RF receiver RSSI. The predefined trajectory was selected as such because it covered a great area of the operating radius of the RF beacons, and because the odometry's drift error would be more visible with a long trajectory. Finally, separate datasets are collected also in triplicate setup with the robot in the initial position, turning twice counter-clockwise and twice clockwise, in order to validate the IMU's gyroscope measurements.

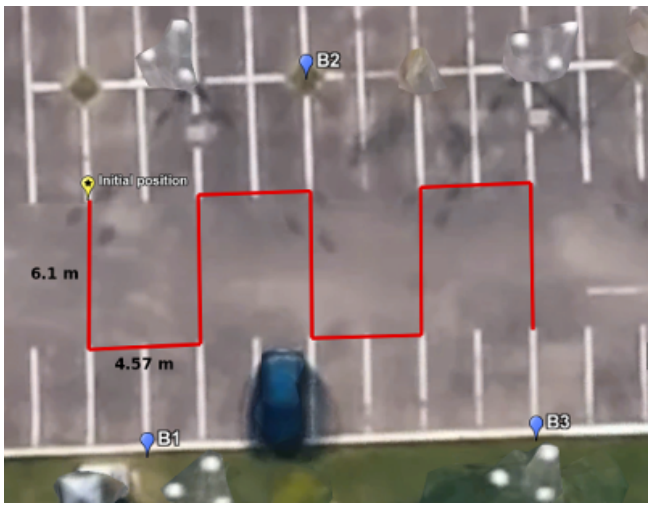


Figure 3.10: Predefined trajectory for dataset collection

In the second experiment, the straight-line ground-truth was defined in the garden, and the beacons were placed in a triangle according to the distances between them, as described above. A scheme for this setup can be seen in Figure 3.11, including the straight-line ground-truth trajectory. After performing the same preliminary steps as before, the robot was tele-operated along the straight-line, while IMU, encoders' odometry and RSSI were recorded, in triplicate. The same experiment was performed in the parking lot, in order to compare the behaviour of localization on the same ground-truth.

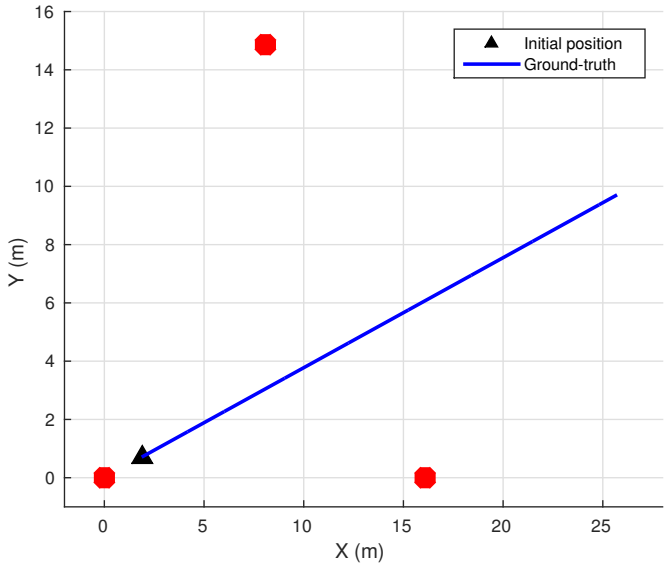


Figure 3.11: Beacon setup and straight-line trajectory for the second experiment

3.3.3 Methodology for Dataset Treatment

For the first experiment, the ground-truth and robot sensor data had to be treated. Firstly, the RTK's base and rover logs had to be downloaded from the Reach's internal memory, which was done through the web-based application. When downloading, the logs are converted to RINEX files, and then the trajectories can be viewed with RTKPLOT, a part of the RTKLIB library. This library also offers a method for post-processing of the RTK logs with the RTKPOST program, greatly increasing the number of accurate points[20]. Thus, the first step in data treatment is to use the RTKPOST program with the best parameters, defined empirically for each run, to improve the accuracy of the ground-truth trajectories as much as possible. The quality of the trajectory can be verified with RTKPLOT, which shows the percentage of accurate points. The output of RTKPOST is a text log file with all the improved measurements.

After post-processing, the ground-truth must be integrated and synchronized with the respective sensor measurements. This is a complicated process mainly because of three problems:

- When recording data with rosbag, there is no synchronization of the data incoming from the topics. Some topics may have different publish rates, and ROS does not discriminate on incoming data. For example, in a recorded bag there may be data recorded at 20, 10 and 5 Hertz, which all need to be synchronized somehow.
- Rosbag timestamps are represented in UNIX format, in which time is represented as seconds since an epoch. For example, the UTC scale (which will be discussed further ahead) counts the number of seconds since 00h00m00s, January 1st, 1970. RTK times are represented in the more friendly ISO 8601 standard, where the time format is "year-month-day hour:minute:second.microsecond".
- The rosbag and RTK times are recorded in different time scales. Rosbag time is measured from the computer's walltime, and thus is in the Universal Time Coordinated (UTC) scale with a timezone offset (-3 hours for Brazil). RTK time is measured in the GPS Time (GPST) scale.

Thus, the following plan was formulated to solve the problems:

1. Convert the RTK times to UTC UNIX timestamps, to be in the same time scale and format as the topics.
2. Incorporate the RTK observations into the sensors' bag files.

[20] https://docs.emlid.com/reach/common/tutorials/gps-post-processing/

3. Synchronize the topics in the bag files.

In practice, step 1 was done in the post-processing stage by selecting the output time format to UTC, which resulted in ISO 8601 standard times. These ISO times were transformed into UTC UNIX timestamps via the "datetime" Python library. To account for Brazil's timezone, 2 hours (with Daylight Saving Time) were subtracted from the resulting timestamps. To perform step 2, a ROS Python node called "*RTK_publisher*" was constructed in order to publish the post-processed RTK observations in a topic, which were then recorded in a bag file. The sensors bag and RTK bag were combined into a third bag file with the rosbag API, containing all the sensor topics and the RTK topic. Finally, another ROS node called "*synchronizer*" was built to use the message filters API, where the "*ApproximateTimeFilter*" function was used to synchronize all topics. The synchronizer node produced a fourth bag file, now with all topics synchronized.

To be able to use the synchronized data in MATLAB, where the Kalman Filters were implemented, the synchronized bag file had to be parsed. For this purpose, another Python script called "*bag_parser*" was constructed, which uses the rosbag API to get data in the IMU, odometry, XBee and RTK topics, and save each one in text log files. The log files are then read and interpreted in MATLAB. The robot's trajectory was estimated with the Extended and Augmented Extended Kalman Filters, and the robot sensor data, as described in Sections 3.2.3 and 3.2.4. The received signal strength measurement models used in the Kalman Filters required the RF beacon coordinates, of which the latitude/longitude coordinates were obtained from Google Maps and converted to local coordinates, considering the robot's initial position from the ground-truth trajectories.

The dataset treatment process described above was repeated in the second experiment, with the difference that the straight-line ground-truth had to be determined. For this, several measurements from the parking lot and garden beacon setups were made, such as distance between beacons, the robot's initial position (considering the origin at B_1), among others. From these measurements, the angle between the ground-truth and the X plane (ϕ) was discovered, and thus the robot's position could be considered in polar coordinates, with the radius being the distance between the robot and B_1 (ranging 0 to 30 meters), and the angle being ϕ. The robot's (x, y) position was obtained by converting from polar to Cartesian coordinates.

3.3.4 Methodology for Sensors Validation

To select which sensors are viable to be used in the Kalman Filters, they have to be validated. This is done by comparing the actual sensor measurements with the ideal measurements (i.e. the values the sensors should ideally report), to check if the sensor

measurements indeed follow a Gaussian distribution with mean centered on the true value. For each of the robot's sensors, the validations are:

- *Ground-Truth*: In order to validate all other measurements (except for the angular velocity), the ground-truth trajectories must be validated. This is done by verifying three criteria, in order to determine the suitability of the Reach RTK trajectories.
- *Global Positioning System*: The USB GPS measurements are verified for each dataset, in the same way as the RTK measurements.
- *Odometry*: The x and y velocities are validated, obtained with the derivation of the positions using the finite difference seen in Equation (3.5). They are compared with the ground-truth x and y velocities, obtained in the same manner.
- *Inertial Measurement Unit*: We check the linear accelerations in x and y and the angular velocity in z. The accelerations are compared with the accelerations derived from the ground-truth through double derivation. The gyroscope's angular velocity is validated through a separate dataset, which will be discussed in Section 4.1.4.
- *Received Signal Strength Indicator*: The RSSI measurements are validated by comparing them to the ideal values given by the Free-Space Path Loss model, using the validated beacon positions.

In the case of odometry, the velocities were selected instead of the position coordinates because the finite difference between measurements does not suffer from drift, as will be demonstrated in Section 4.1.3.

4. RESULTS

The results for both experiments are presented here, obtained by collecting and treating the datasets. Section 4.1 discusses the validation of the sensors, where characteristics such as bias and noise for each sensor are evaluated. Section 4.2 presents the trajectories obtained with the Extended and Augmented Extended Kalman Filters in the first experiment, the values found for the covariance and uncertainty matrices Q and R, and the effects of each sensor on the quality of the trajectories for both filters. Section 4.3 describes the results for the second experiment, specifically comparing the robot straight-line trajectories and error dispersion between stand-alone odometry and the full localization system in both the parking lot and garden test areas.

As the Chapter heavily discuss the triplicate datasets collected with and without the absorbing plate, it is interesting to define a common nomenclature for them. As such, let $(A, B, C)_w$ be the three datasets collected with the absorbing plate and $(A, B, C)_{wo}$ be the three datasets without the absorbing plate. For example, A_w refers to the first dataset with the plate, while C_{wo} is the third dataset without the plate. Each experimental run lasted for approximately 120 seconds. All source codes for sensor validation can be found on the project's GitHub repository[8].

4.1 Sensor Validation

This Section discusses the validation of the sensors to be used in the localization, specifically the RTK ground-truth, USB GPS trajectories, the odometry velocities and accelerations, the Inertial Measurement Unit's accelerations and angular velocity and the received signal strength of the XBee radios. Through this analysis, we can select the most appropriate sensors to be used in localization and find out caveats in their utilization.

One of the most crucial steps in developing a localization solution with the Kalman Filters is the fine-tuning of the covariance and uncertainty matrices Q and R (page 51). In order to facilitate this process, a small dataset of all sensor data (except RTK) was collected for 3 minutes with the robot standing still, to approximate their covariances. The intuition is that, as our sensors capture the robot's motion and the robot is standing completely still, variations in the sensor readings are due only to measurement noise and uncertainties. To reference this dataset in the subsections below, let it be denoted as the "stationary" dataset.

4.1.1 Ground-Truth

The validation of the ground-truth trajectories is done by evaluating three criteria: Shape, continuity and scale. The shape has to be consistent with the predefined trajectory seen in Figure 3.10 of Section 3.3.2 (page 78). In regards to continuity, subsequent position observations should not deviate from the robot's path, as the observations were obtained with the robot moving forward and turning, resulting in a continuous trajectory. Finally, the scale must be verified, to see if the travelled distances estimated by the RTK are coherent with the distances travelled in the real world.

The Emlid Reach and RTKLIB library use a specific terminology to indicate the quality of the position observations. When the Reach module on the robot is operating without corrections from the base station, its observations are dubbed "*single*". This is the worst case scenario in terms of accuracy. When the module is receiving corrections but has not yet converged (i.e. solved the ambiguity problem), the observations are called "*float*". With convergence, the observations are named "*fix*", and are as accurate as possible. Thus, the overall quality of the trajectories can be measured in terms of the percentage of fix position estimates.

Post-processing with the RTKPOST software allows for great improvements in the number of fix estimates. The best post-processing parameters (e.g. atmosphere correction, carrier frequency, etc) vary with each trajectory and have to be defined empirically by running RTKPOST and counting the number of fix points in the post-processed log file. The results presented here reflect the set of parameters which resulted in the highest percentage of fix points, and thus the best quality of observations. An important point is that absolute accuracy in latitude/longitude is unimportant, as the ground-truth observations are converted to local coordinates, with the origin at the robot's initial position. Only the distances and path travelled by the robot must be captured by the ground-truth, which can be validated with the three criteria discussed previously.

Tables 4.1 and 4.2 present the percentages of fix, float and single observations for all trajectories, before and after post-processing. While the single observations are presented, they are discarded through the synchronization process discussed in Section 3.3.3. It is possible to conclude from the tables that the quality of the post-processed trajectories is indeed superior. For this reason, the three validation criteria will be analyzed for the post-processed trajectories in MATLAB.

Table 4.1: Percentage of fix, float and single points for the trajectories before post-processing

	A_{wo}	B_{wo}	C_{wo}	A_w	B_w	C_w
Fix %	0	0	69.8	0	0	0
Float %	96.8	92.7	29.3	93.8	96.3	96.7
Single %	3.2	7.3	1.0	6.2	3.7	3.3

Table 4.2: Percentage of fix, float and single points for the trajectories after post-processing

	A_{wo}	B_{wo}	C_{wo}	A_w	B_w	C_w
Fix %	94.9	94.2	91.6	100	100	96.1
Float %	4.5	5.8	8.1	0	0	3.9
Single %	0.6	0	0.3	0	0	0

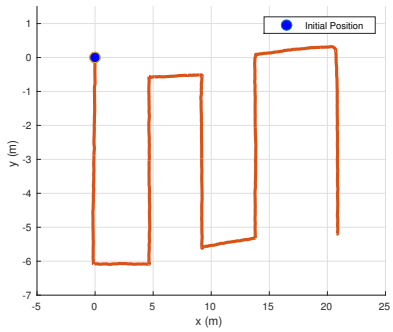
Figure 4.1 shows the trajectories without the absorbing plate. The shape criterion can be verified for the three ground-truths, as they are coherent with the predefined trajectory with some deviation, which is due to uncertainty in the motors' output as well as wheel slippage and other external disturbances. For the continuity criterion, A_{wo} is continuous, while C_{wo} has a small gap in the last straight line. B_{wo} has a severe discontinuity in the final stretch, and as such this trajectory can be used only up to that point, or else there will be a fixed error in the estimated trajectory. Finally, the scale can be analyzed by integrating the displacement between observations to determine the total distance travelled by the robot. With this, we can calculate the accuracy of that trajectory: How far away the total distance travelled is from the predefined one. It is formulated as [28, p. 106]:

$$\text{Accuracy} = \left(1 - \frac{|D_{gt} - D_{pred}|}{D_{pred}}\right) \times 100 \tag{4.1}$$

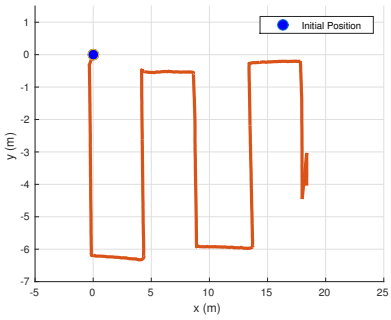
where D_{gt} is the total distance in each ground-truth and D_{pred} is the total distance in the predefined trajectory. When the accuracy is high, the dataset and predefined paths are similar in scale. There is some ambiguity associated with this metric, as different path configurations can produce the same distance. However, in this case, the accuracy of the total distances gives us a good idea of the ground-truth's correctness, as it has been established that the ground-truth shapes are coherent with the predefined trajectory. The distances and accuracies for $(A, B, C)_{wo}$ can be found in Table 4.3.

Table 4.3: Total distances and accuracies for the datasets without absorbing plate

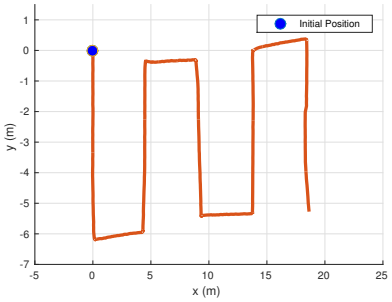
	Predefined	A_{wo}	B_{wo}	C_{wo}
Total Distances (m)	48,78	49,58	46,20	46,15
Accuracy (%)	-	98,35	94,72	94,61



(a) A_{wo}



(b) B_{wo}



(c) C_{wo}

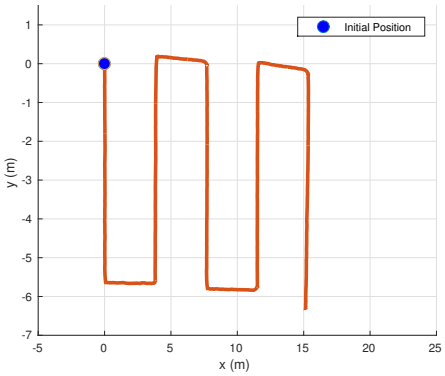
Figure 4.1: Ground-truth trajectories for datasets without absorbing plate

For the datasets with the absorbing plate, the same analysis is done for A_w, B_w and C_w, which can be seen in Figure 4.2. The trajectories in A_w and B_w have no discontinuities due to their observations being 100% fix. There are discontinuities before the final turn in C_w, and as with B_{wo} the experiment has to be stopped early. In regards to shape, the trajectories with the plate are slightly more well-behaved and uniform than the ones without, except in the case of C_w. The scale is once again very similar to the predefined trajectory, with accuracies above 90%, as can be seen in Table 4.4.

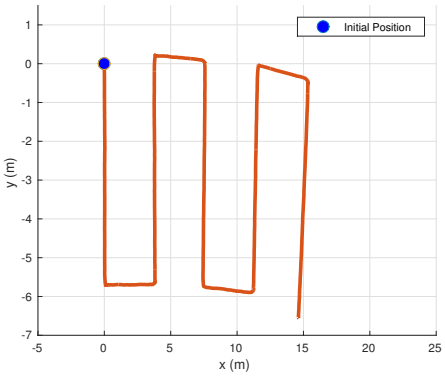
Table 4.4: Total distances and accuracies for the datasets with absorbing plate

	Predefined	A_{wo}	B_{wo}	C_{wo}
Total Distances (m)	48,78	44,74	44,77	45,80
Accuracy (%)	-	94,03	94,10	96,27

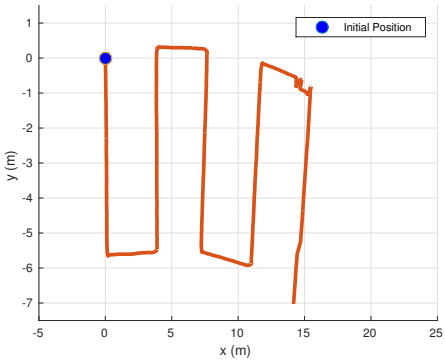
In conclusion, the ground-truths obtained with and without the absorbing plate are adequate for the experiments. Special care must be taken in the cases of B_{wo} and C_w, as they present discontinuities near the end. These experiments must be cut short in order to avoid the discontinuities' effect on the evaluation of the localization performance.



(a) A_w



(b) B_w



(c) C_w

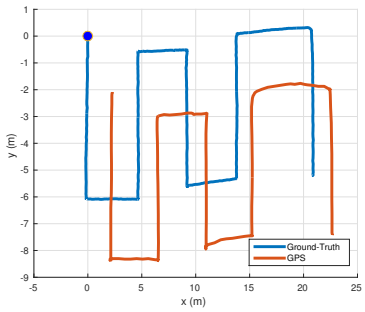
Figure 4.2: Ground-truth trajectories for the datasets with absorbing plate

4.1.2 Global Positioning System

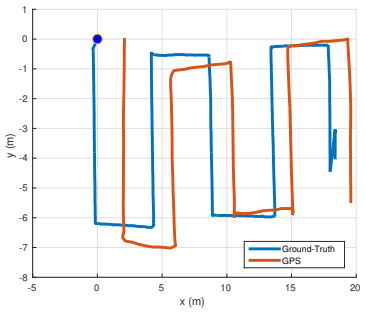
As this project aims to estimate a mobile robot's pose in a GPS-deprived environment, or to reduce the error of the GPS localization, the trajectory obtained by the robot's GPS receiver must be evaluated in order to know how much better or worse our localization system is. As such, the robot's USB GPS trajectories have to be validated. This is done in a process similar to the validation of the ground-truth, except that the only criterion to be considered is the shape. It is not relevant to consider discontinuities or scale because, differently from the ground-truth, the GPS has no commitment in estimating the most correct trajectory possible (i.e. the positions reported may be wrong in x or y, causing discontinuities).

Figure 4.3 shows the plotted GPS and ground-truth trajectories for the $(A, B, C)_{wo}$ datasets. As the results for $(A, B, C)_w$ have similar characteristics, they can be found in Appendix A. In all six datasets, the shape criterion is verified, as the GPS trajectories are consistent with the ground-truth path. There are almost no discontinuities in the trajectories, indicating that the sensor can follow the robot's motion very well once an initial estimate is found. However, there are constant biases for the x and y coordinates, which are different for each dataset. As such, there is a constant error in the localization provided by the

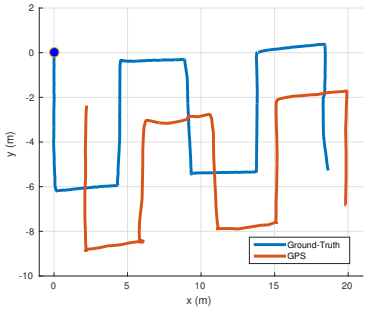
GPS, changing with each experimental run, which justifies the use of the Kalman Filters for improving localization.



(a) A_{wo}



(b) B_{wo}



(c) C_{wo}

Figure 4.3: GPS and ground-truth trajectories for $(A, B, C)_{wo}$

Furthermore, the USB GPS' positioning error can be seen in Figure 4.4, collected from the stationary dataset. With the robot stopped, the position varies up to 0.5 meters in x and y. The sensor covariance can also be extracted from the dataset, being equal to 0.0181 and 0.0891 in x and y respectively.

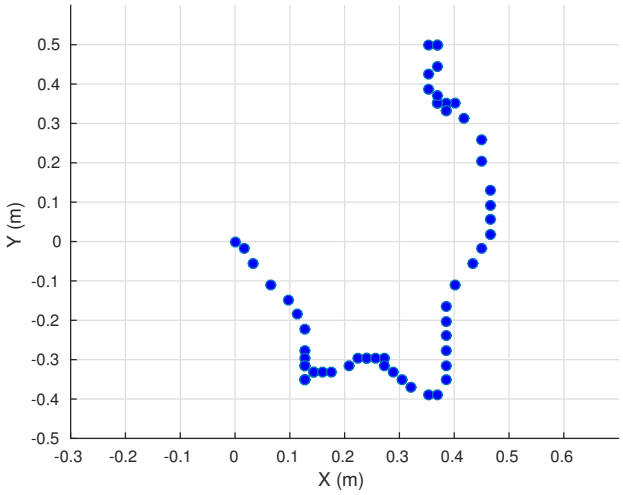
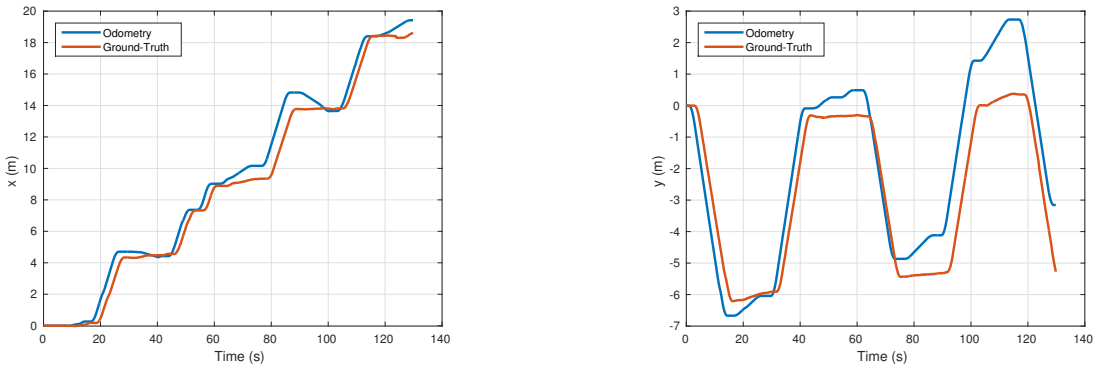


Figure 4.4: Positioning error of the USB GPS sensor

4.1.3 Odometry

As the ground-truth trajectories are validated in Section 4.1.1, they can be used to validate the odometry velocities and accelerations in x and y. As stated previously, these measurements are better suited for localization because they do not suffer from drift. The position estimates accumulate error due to odometry being a dead reckoning method, but the velocities (position variations) remain the same regardless of the difference between ground-truth and odometry. The effect of drift can be seen in Figure 4.5, where the odometry from C_{wo} is compared with the ground-truth. There is some sway in the drift for x, which is due to the odometry's rotation model: It sometimes over-estimates the robot's turns, causing the robot position to retrocede in x. However, it is clear to see that the robot is drifting upwards in y. The drift varies between the datasets, as it depends on uncertainties and noise.



(a) Drift effect for the X coordinate

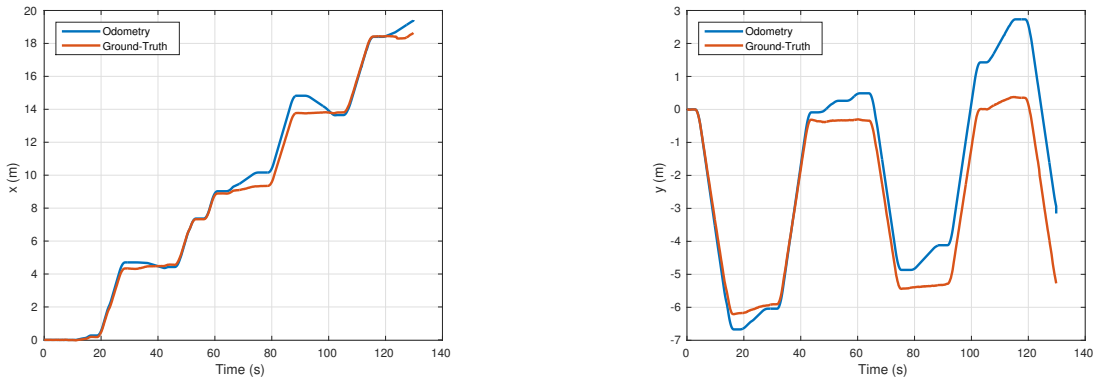
(b) Drift effect for the Y coordinate

Figure 4.5: Drift effect in the robot's odometry for the C_{wo} dataset

Figure 4.5 shows another important error: The ground-truth is delayed approximately 2 seconds relative to the odometry. The cause of this is unknown, possibly being due to some systematic error in the synchronization process.In order to correct this, an *offset vector* is concatenated to the beginning of the odometry x and y data vectors. The size of the offset vector is empirically defined, and all of its values are equal to the first value of the x and y vectors respectively. With this modification, the ground-truth and odometry in phase can be seen in Figure 4.6, once again for C_{wo}. Each triplicate set has its own offset size, which are: $A_{wo} = A_w = B_w = 9$ samples; $B_{wo} = C_{wo} = C_w = 10$ samples.

With the delay gone, we can properly validate the velocities and accelerations. This is done by comparing the measured quantities with their ideal values, and thus the ideal velocities and accelerations have to be calculated from the ground-truth. In discrete-time, this can be done through the finite difference method:

$$\dot{x}_{ideal} = \frac{x(t+1) - x(t)}{T} \tag{4.2}$$



(a) Odometry offset corrected for X coordinate (b) Odometry offset corrected for Y coordinate

Figure 4.6: Ground-truth and odometry in phase for the C_{wo} dataset

$$\dot{y}_{ideal} = \frac{y(t+1) - y(t)}{T} \tag{4.3}$$

$$\ddot{x}_{ideal} = \frac{\dot{x}(t+1) - \dot{x}(t)}{T} \tag{4.4}$$

$$\ddot{y}_{ideal} = \frac{\dot{y}(t+1) - \dot{y}(t)}{T} \tag{4.5}$$

where $t+1$ denotes the next measurement and T is the sampling time, in this case, 0.2 seconds. The same is done for the odometry vectors, obtaining the odometry velocity and acceleration in x and y. Figure 4.7 shows the comparison for the C_{wo} dataset. The expected and measured velocities are very similar: The odometry estimate follows the ground-truth closely, even in the case of sudden peaks and drops. However, there are errors, such as when the robot stops to turn and the drift effect makes it seem as if the robot was moving forward/backward. Another error is that there is a small bias in the peak velocities. These errors must be accounted for by adjusting odometry's covariance in the Kalman Filters. For the accelerations, except for occasional spikes caused by accentuated noise, the ideal and measured mean values are very close, which indicates that the measured accelerations vary around the ideal values according to a Gaussian distribution. Their behavior is also consistent with what happens in the real world: When the robot starts moving forward there is positive acceleration, and when it is stopping there is negative acceleration. Once again disconsidering random noise spikes, the same analysis holds true for A_{wo} and B_{wo}, and as such their comparisons can be found in Appendix A.

For the datasets with the absorbing plate, Figure 4.8 shows the comparison for A_w. There are sudden spikes in the x and y velocities/accelerations when the robot it at top speed, an effect which happens to a lesser degree in B_w and C_w. The possible causes for this are many, for example effector noise in the motors due to loose shafts, causing the encoders to spin a bit faster than they should. However, these spikes are short and do not affect the odometry measurements much, as evidenced by the overall small drift in x and y seen in Figure 4.9. Apart from these errors, the same properties observed for the

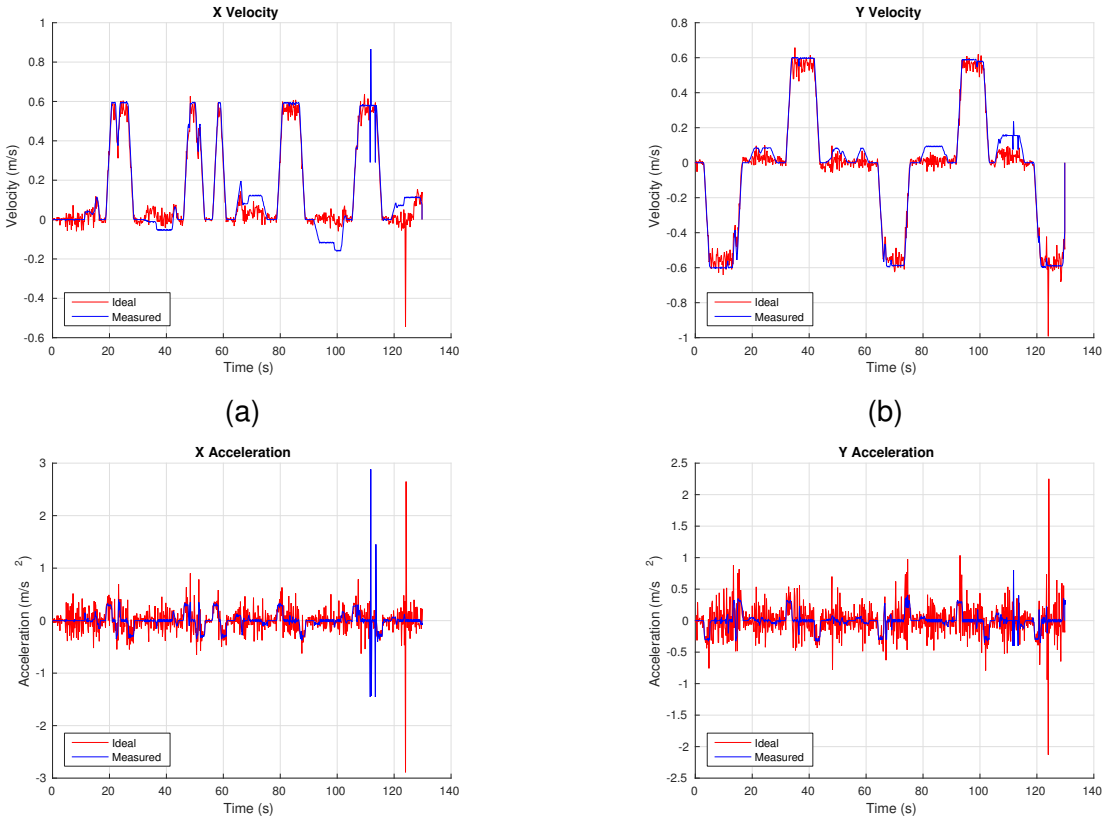


Figure 4.7: Ground-truth and odometry velocities/accelerations for the C_{wo} dataset

$(A, B, C)_{wo}$ datasets hold true here. Once again, the comparisons for B_w and C_w can be found in Appendix A.

Thus, we conclude that the odometry velocities and accelerations are valid measurements to be used in the Kalman Filters, as they approximate the ideal measurements and do not suffer from drift. The small biases and noise effects must be considered when tuning the odometry's covariance values, which in this case can not be approximated with the stationary dataset. This is because the encoders do not measuring anything with the robot stationary, and thus can not suffer from noise/uncertainty.

4.1.4 Inertial Measurement Unit

To validate the IMU measurements, first the gyroscope's angular velocity must be validated, as the IMU reports the robot's accelerations in the local frame (frontal and lateral motion), and angular position is needed to convert the ground-truth accelerations to the local frame and then properly compare the ideal and real measurements. However, the robot's motion restrictions must be considered. While the IMU capable of measuring rotation in 3 Degrees-of-Freedom, the only possible rotation for the robot is about its vertical axis. As the

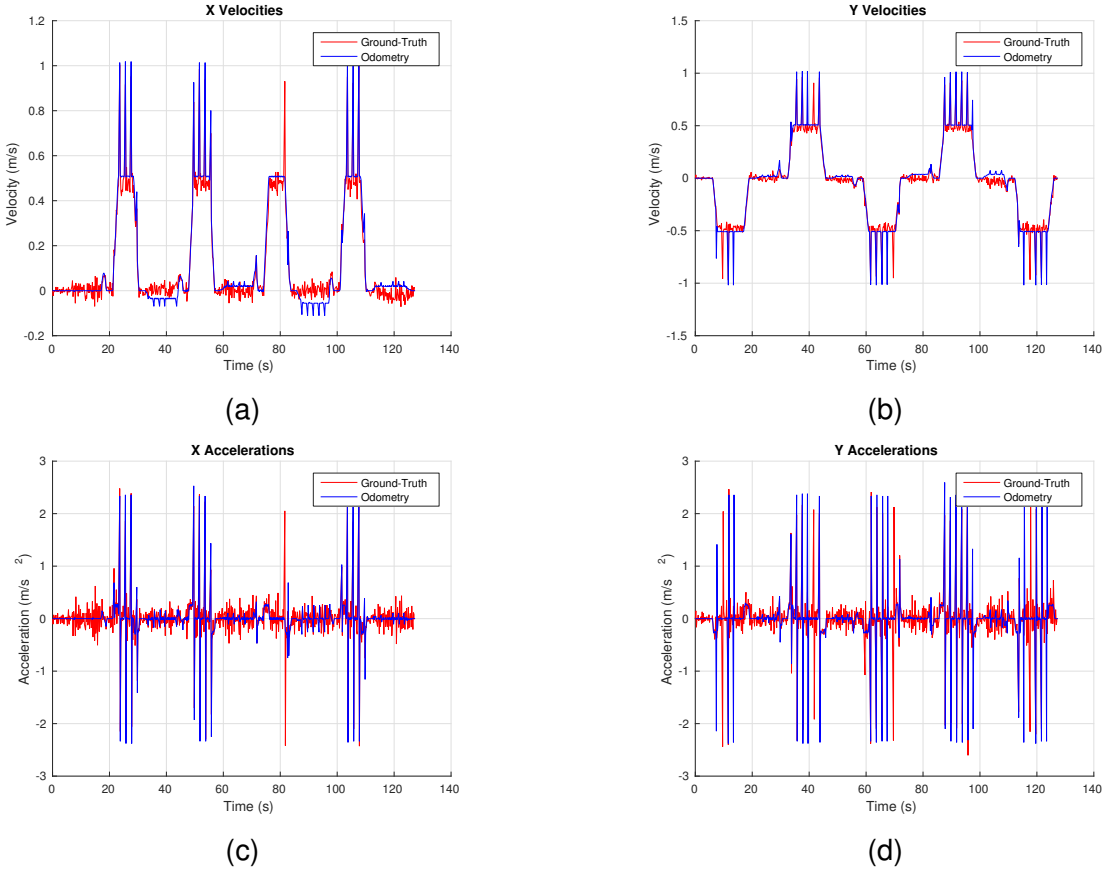
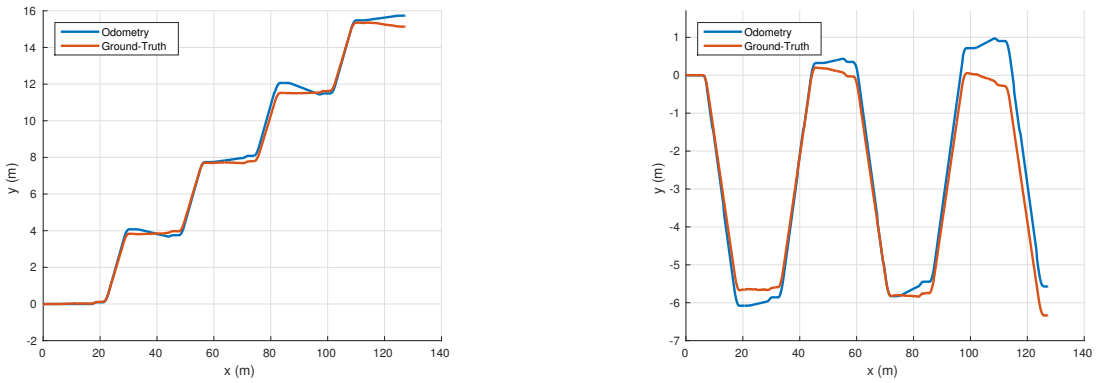


Figure 4.8: Ground-truth and odometry velocities/accelerations for the A_w dataset



(a) Drift effect for the X coordinate

(b) Drift effect for the Y coordinate

Figure 4.9: Drift effect in the robot's odometry for the A_w dataset

sensor is mounted upright on the robot, the only rotation possible is in the IMU's Z axis, and as such only the angular velocities in Z (ω_z) will be validated.

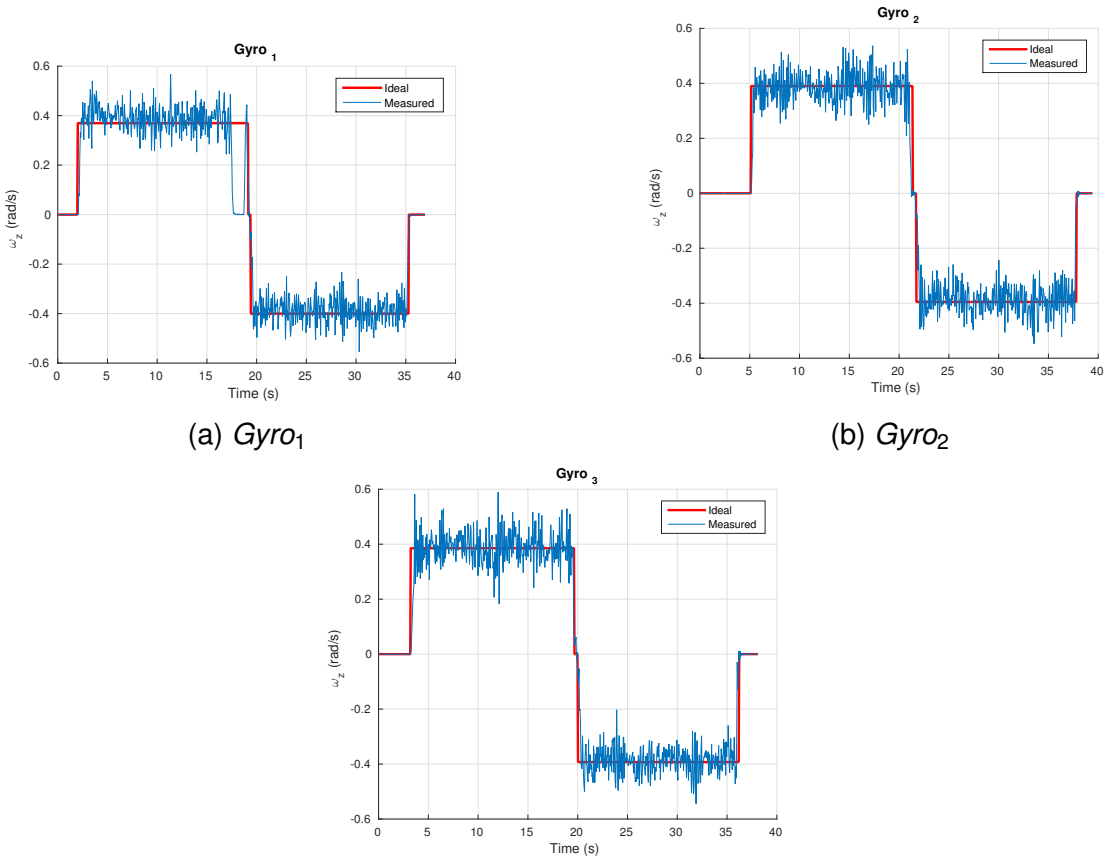
In order to do this, three distinct datasets were used, here denoted by $Gyro_1$, $Gyro_2$ and $Gyro_1$, where the robot spins once counter-clockwise and once clockwise, while the IMU data is collected in a rosbag. The times for both rotations (ΔT_{cc} and ΔT_c respectively) are later measured through the bag file's timestamps, and the instantaneous counter-clockwise and clockwise angular velocities (ω_z and $-\omega_z$) are calculated, with knowledge that the robot

spun 2π and -2π radians in ΔT_{cc} and ΔT_c. The times and angular velocities for the three datasets are presented in Table 4.5, as well as the variances for each variable along the datasets.

Table 4.5: Rotation times and angular velocities in z for the Gyro datasets

	$Gyro_1$	$Gyro_2$	$Gyro_3$	σ^2
ΔT_{cc} **(s)**	17.0	16.1	16.3	0.22330
ΔT_c **(s)**	15.7	15.9	16.0	0.02330
ω_z **(rad/s)**	0.37	0.39	0.38	0.00010
$-\omega_z$ **(rad/s)**	-0.40	-0.39	-0.39	0.00003

Figure 4.10 shows the comparison between the instantaneous ω_z and the measured angular velocities. As shown by the small variances of ω_z in Table 4.5, the sensor measurements are close to the ideal value, following a Gaussian distribution. This indicates the validity of the ω_z measurements and their suitability to be used in the Kalman Filters.



(a) $Gyro_1$

(b) $Gyro_2$

(c) $Gyro_3$

Figure 4.10: Comparison between instantaneous and measured ω_z for the gyro datasets

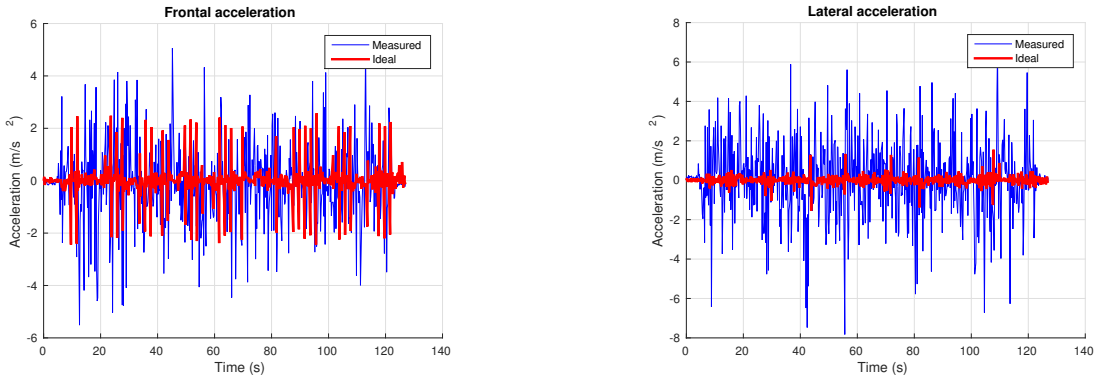
With the gyroscope validated, we can verify the frontal and lateral accelerations reported by the IMU in each of the $(A, B, C)_{wo}$ and $(A, B, C)_w$ datasets. The ideal values in this case are the frontal and lateral accelerations of the ground-truth in each dataset, found

with the double derivation of the expression for the local robot pose in Equation (2.7) (page 35):

$$\ddot{\xi}_L = \begin{bmatrix} a_{front} \\ a_{lat} \\ \ddot{\theta} \end{bmatrix} = R(\theta)\ddot{\xi}_G = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} \tag{4.6}$$

where $\ddot{x}$ and $\ddot{y}$ are the accelerations in x and y, found by derivating the ground-truth positions twice through the finite differences formula. The orientation θ can be found by integrating the angular velocities reported by the gyroscope in each dataset. Thus, the direct application of Equation (4.6) with θ and ($\ddot{x}$,$\ddot{y}$) gives us the frontal and lateral accelerations for the ground-truths.

Unfortunately, the IMU acceleration measurements do not fare well. Figure 4.11 compares the ground-truth and measured frontal/lateral accelerations for the A_{wo} dataset. As the results for all datasets are similar, the other comparisons can be found in Appendix A. There is a great amount of noise in the measurements, due to the sensibility of the accelerometers. As the robot is a traversing the uneven terrain of the parking lot, the IMU vibrates along with the robot's body, causing short acceleration bursts. The sensor was placed in a sponge base to reduce this effect, but it is still very significant. To be used in the Kalman Filters, these accelerations would have to be represented with very large covariances, effectively nullifying their effect on the pose estimation, or else they can potentially impair the pose estimation. As such, the IMU's frontal/lateral accelerations are unsuitable to be used in the sensor fusion.



(a) (b)

Figure 4.11: Comparison between ideal and measured frontal/lateral accelerations for the A_{wo} dataset

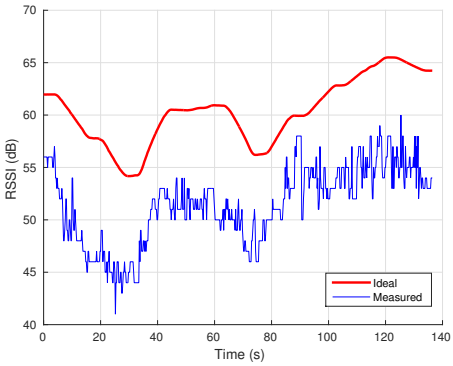
4.1.5 Received Signal Strength Indicator

The main contribution of this project is the implementation of a localization system that uses the received signal strength (or RSSI) of radio-frequency transmissions to improve a robot's pose estimation. Localization in GPS works similarly: The pseudo-ranges between satellites and the GPS receiver are calculated with the time-of-flight of the satellite carrier wave. As the distances between satellites and ground receiver are very large, this time-of-flight is measurable. With the beacon setup discussed previously, the waves travel too fast for the time differences to be captured, and thus the property of signal strength path loss must be exploited instead, which is defined by the Free-Space Path Loss model (Equation 2.41, page 59). These measurements are notorious for being highly sensible to environmental conditions, and as such their suitability for localization must be verified.

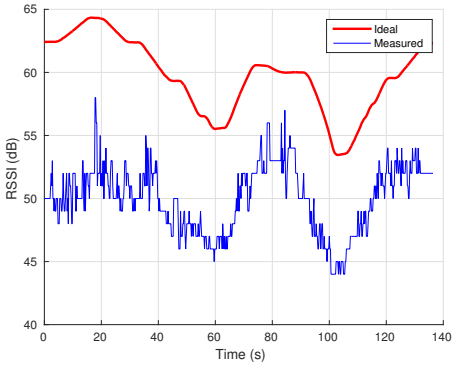
This can be done by comparing the ideal Path Loss (in decibels) from the Free-Space model to the actual measurements reported by the robot's XBee receiver. As before, the ideal measurements can be found by using the ground-truth trajectories of the six datasets. With the known beacon locations, the straightforward application of Equation (2.41) gives us the expected RSSI values for each beacon-receiver pair, which are then compared with the actual measurements. Figure 4.12 shows this comparison for the A_{wo} dataset, while the comparison for B_{wo} and C_{wo} can be found in Appendix A. The A_{wo} comparison was chosen here because it depicts the most relevant characteristics of using the XBee receiver without the absorption plate.

As can be seen in Figures 4.12a and 4.12b, the measured RSSI follows the shape of the ideal measurements, albeit with much noise and a semi-constant bias, which is caused by attenuation in the electromagnetic signals due to environmental conditions such as destructive interference from other sources of radio signals (i.e. Wi-Fi, Bluetooth). However, the biggest problem with this sensor can be seen in Figure 4.12c. The received strength for the third beacon is maintained at an almost constant value, as if the robot was moving in a circle with constant radius around B_3, possibly due to multi-path interference. This is extremely detrimental to localization, because the pose estimate is restricted to this circle. As this happens due to uncertainty, it affects each dataset differently.

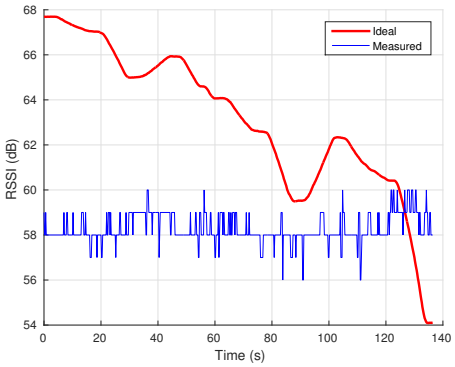
The use of the absorption plate is fundamental in eliminating this error. As seen in Figures 4.13, A.11 and A.12 for A_w, B_w and C_w respectively, the multi-path effect is gone. The plate works as a shield, absorbing electromagnetic signals that are reflected by the ground and arrive at the antenna from below, ensuring only the strongest signal is detected by the antenna. Apart from preventing multi-path interference, the plate also reduces the noise of the RSSI and improves its uniqueness with different distances, coherently with the results reported in [15]. However, the biases are still present in the measurements, and thus, the idea of estimating the biases iteratively in the Extended Kalman Filter was conceived,



(a) Beacon 1



(b) Beacon 2

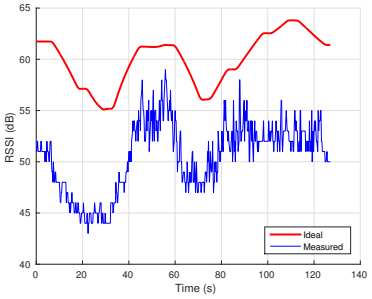


(c) Beacon 3

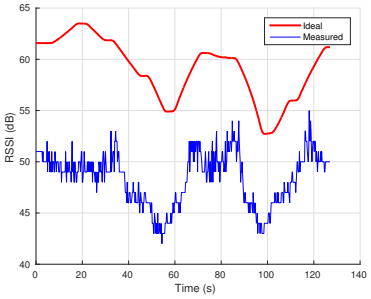
Figure 4.12: RSSI value comparison for the three beacons for the A_{wo} dataset

resulting in the Augmented EKF seen in Section 3.2.4. The unbiased RSSI measurements will be discussed in Section 4.2.2, along with the results in localization.

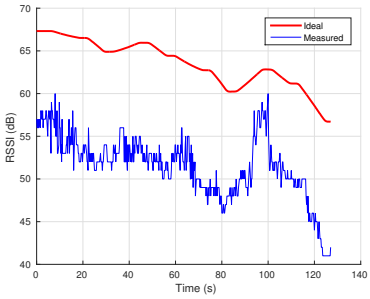
In conclusion, the received signal strength indicator is an adequate measurement to be used in localization, as long as the necessary steps are taken to reduce noise, ensure its uniqueness and correct the biases. The effect of the absorbing plate can be further verified when approximating the covariances for each beacon with the stationary dataset. As seen in Figure 4.14, the signal attenuation is a bit smaller (the values are higher in decibels), and the spread of the measurements for all beacons is lower with the plate, endorsing the conclusions drawn previously.



(a) Beacon 1

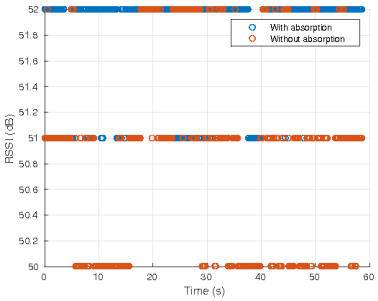


(b) Beacon 2

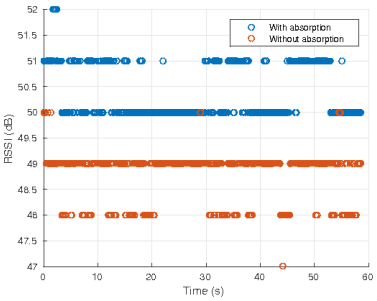


(c) Beacon 3

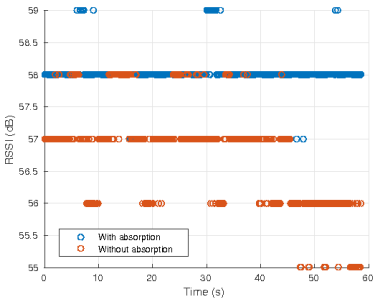
Figure 4.13: RSSI value comparison for the three beacons for the A_w dataset



(a) Beacon 1



(b) Beacon 2



(c) Beacon 3

Figure 4.14: Effect of the absorbing plate on the RSSI values

4.2 Robot Trajectory Estimation

With the sensors validated, their data can be used to recover the robot's trajectory with the Extended Kalman Filters defined in Sections 3.2.3 and 3.2.4. The trajectories for the six datasets are recovered and then compared with the ground-truth, and the Root-Mean-Square Error (RMSE) is used to determine the quality of localization:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|(x,y)_{i,est} - (x,y)_{i,gt}\|^2} \tag{4.7}$$

where $\|(x,y)_{i,est} - (x,y)_{i,gt}\|$ is the Euclidean norm of the estimated trajectory relative to the ground-truth, and n is the number of pose estimates recovered. The effect of the different sensors in the pose estimation is also of interest, specially in the case of the RSSI. To test this, the covariance of each sensor is individually set to an absurdly high amount, effectively eliminating its influence on the filter. As such, it is possible to see if that particular sensor's absence improves or damages the quality of localization.

As the problem to be solved is robot localization in environments with little or no GPS availability, let us specify the GPS trajectories as the baseline for our experiments: The localization system is successful if it produces a higher quality trajectory. Apart from the root error, another important metric for this project is the *average deviation*, which relates to the repeatability of the solution. This is the reason for the triplicate experimental setup: We analyze the mean errors for the three trajectories with and without plate, and evaluate if the results are statistically significant through the average deviation:

$$deviation = \frac{\Sigma|x - \mu|}{n} \tag{4.8}$$

where x is a set of measurements, μ is the set's mean value and n is the number of measurements in the set. The deviation ranges from 0 to infinity: A lower (close to 0) deviation means that the results are more repeatable, as they are closer to each other (and so more statistically significant), meanwhile a higher deviation indicates a greater disparity between the results. For brevity, the subsections below present only the most relevant/best results, while the others can be found in Appendix A.

4.2.1 Extended Kalman Filter

In order to use the Extended Kalman Filter as it is described in Section 3.2.3, first the covariance and uncertainty matrices Q and R must be defined. The covariance matrix is constructed with the values for each sensor previously discussed, except in the case of

the odometry, as the covariances could not be approximated with the stationary dataset and thus had to be determined empirically. As the RSSI has two distinct covariances depending on the hardware setups (i.e. with and without the plate), two Q matrices are defined. These matrices are fine-tuned by repeatedly running the EKF, resulting in:

$$Q_{wo} = diag(B_1, B_2, B_3, \dot{x}_{odom}, \dot{y}_{odom}, x_{gps}, y_{gps})$$

$$= \begin{bmatrix} 4012.3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1549.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4779.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 26.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 67.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 18.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 89.1 \end{bmatrix} \tag{4.9}$$

$$Q_w = \begin{bmatrix} 320.17 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 516.17 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 158.02 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 26.20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 67.30 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 18.10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 89.10 \end{bmatrix} \tag{4.10}$$

The states uncertainty matrix R is defined empirically and once again fine-tuned by running the EKF iteratively, being the same for all datasets:

$$R = diag(x, \dot{x}, y, \dot{y}, \theta)$$

$$= \begin{bmatrix} 10 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 10 \end{bmatrix} \tag{4.11}$$

The EKF's input vector u is composed of the odometry's frontal and lateral accelerations, and the gyroscope's angular velocity. Figure 4.15 shows the trajectory obtained by the EKF using the C_{wo} dataset, which was selected for being the only dataset without the plate that had consistent beacon measurements. With the Q_{wo} matrix defined above, the pose estimation is almost instantly pulled towards the erroneous GPS measurements, but are then corrected by the odometry and RSSI. The sawtooth effect we see on the figure is because of the different update rate of the GPS, which pulls the trajectory back every 5 iterations. A big deviation occurs after the third turn, caused by a sudden spike in the received

signal strengths. After that, the filter is capable of following the ground-truth with a fixed error.

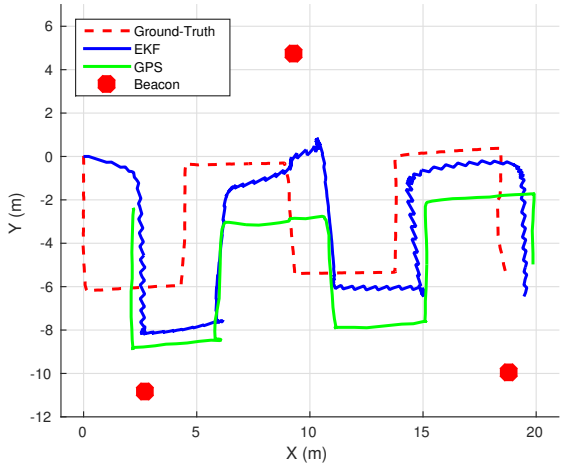


Figure 4.15: EKF trajectory for C_{wo} using all sensors

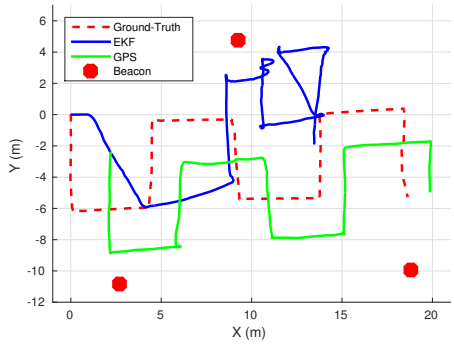
To observe the effect of the GPS in pose estimation, we can increase its covariance by a large amount and see how its absence affects the filter. Figure 4.16a depicts the trajectory without GPS, which is completely incoherent due to the biased RSSI measurements. In this case, the proposed alternative sensors actually hamper the localization solution. Increasing the RSSI covariances gives us Figure 4.16b, where the filter's pose estimation follows the GPS closely. Finally, eliminating the effects of odometry gives us Figure 4.16c, which is similar to the trajectory estimated with all sensors.

Although only the trajectories for C_{wo} are shown here, the localization errors for the A_{wo} and B_{wo} datasets are presented in Table 4.6, needed in order to calculate the Extended Kalman Filter's deviation. The best results for each dataset are presented in bold and the baseline is the GPS error, as discussed previously.

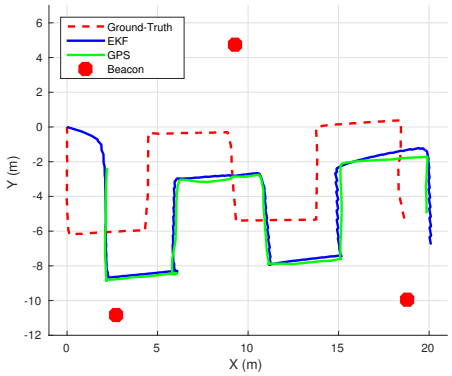
Table 4.6: EKF localization errors for the $(A, B, C)_{wo}$ datasets

	A_{wo} (m)	B_{wo} (m)	C_{wo} (m)	Mean Error (m)	Deviation (m)
GPS	2.739	1.728	2.891	2.453	0.483
All Sensors	1.994	2.984	2.005	**2.328**	0.437
Odom+RSSI	5.599	3.386	4.716	4.567	0.787
Odom+GPS	2.588	2.960	2.733	2.760	**0.133**

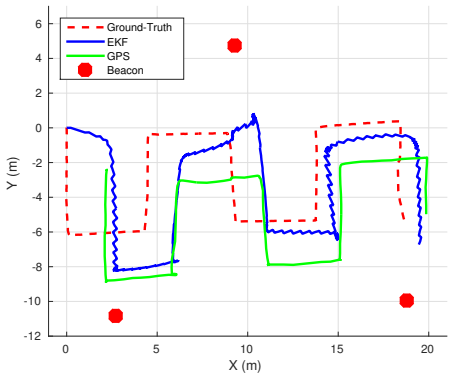
In regards to the datasets with the absorbing plate, Figure 4.17 shows the estimated trajectory for the A_w dataset. The received strength measurements now violently pull the estimation towards a specific point between the beacons, an effect which is corrected every 5 iterations by the GPS. The distortion caused by the RSSI here is much stronger due to the lower covariances associated with the absorbing plate, which if elevated result in a much more coherent trajectory, as seen in Figure 4.18b. If the GPS' covariance is elevated



(a) No GPS



(b) No RSSI



(c) No Odometry

Figure 4.16: EKF trajectory estimation for C_{wo} without GPS, RSSI and odometry

instead, we have the worst possible outcome for this dataset in Figure 4.18a, as the RSSI is free to pull the trajectory downwards. If the odometry is eliminated, once again we have a similar result to the full sensor EKF. Although its mean error is not the lowest, the deviation for the filter with odometry and GPS (i.e. without RSSI) endorses the consistency of this localization solution variation: Among these solutions, this is the most reliable.

Table 4.7 shows the errors and deviations for the datasets with the absorbing plate.

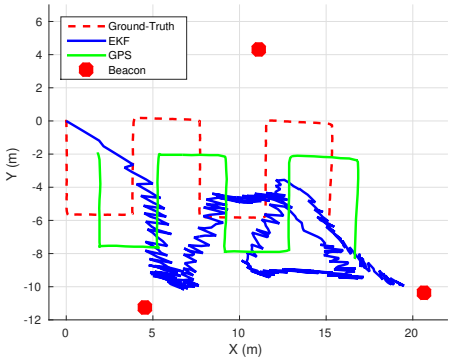
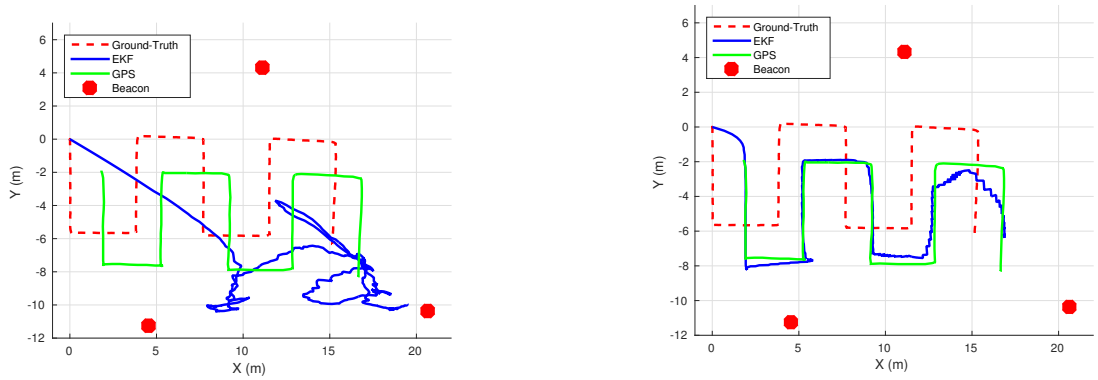


Figure 4.17: EKF trajectory for A_w using all sensors



(a) No GPS (b) No RSSI

Figure 4.18: EKF trajectory estimation for A_w without GPS and RSSI

Table 4.7: EKF localization errors for the $(A, B, C)_w$ datasets

	A_w	B_w	C_w	Mean Error (m)	Deviation (m)
GPS	2.650	2.226	2.270	2.382	0.179
All Sensors	6.236	5.877	5.669	5.927	0.206
Odom+RSSI	8.863	8.330	8.125	8.439	0.282
Odom+GPS	2.487	2.100	2.224	**2.270**	**0.144**

In light of these results, we conclude that the RSSI values are beneficial to the $(A, B, C)_{wo}$ datasets while detrimental to $(A, B, C)_w$. This indicates that the absorbing plate is harming localization, as the covariances for the RSSI with the plate are lower. More specifically, the biased measurements have a stronger influence on the EKF estimation, thus corrupting it. Compared to our baseline, the EKF improves localization for the datasets without the plate, as evidenced by the lower mean error. However, its deviation is also higher, indicating that this method is not consistent due to the stochastic nature of the RSSI measurements. For the datasets with the plate, using only the GPS and odometry measurements is more beneficial, once again endorsed by the low deviation.

4.2.2 Augmented Extended Kalman Filter

The results obtained in the previous Section contradict the results from Section 4.1.5, in which it was seen that the absorbing plate improves the uniqueness of the received signal strength, reduces noise and avoids the problem of multi-path interference. However, the effect of attenuation in the RSSI is present in the Extended Kalman Filter, and as such the Augmented Extended Kalman Filter must be used in order to estimate and correct this. The same covariance matrices Q_{wo} and Q_w for the EKF are used here, and the R matrix is augmented with the three beacon bias states:

$$R = diag(x, \dot{x}, y, \dot{y}, \theta, \beta_1, \beta_2, \beta_3)$$
$$= \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1250 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 1250 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 & 1250 \end{bmatrix} \quad (4.12)$$

The same analysis from Section 4.2.1 will be performed here, for the datasets with and without the plate: First the best estimation with all sensors will be presented, and then the individual influence of each sensor will be evaluated by removing it from the filter through the covariance values. Finally, the errors and deviations will be presented and the localization solution will be compared with the baseline.

Figure 4.19 shows the trajectory estimated with the A-EKF, once again for the C_{wo} dataset. A notable improvement in the shape can be seen, as the sawtooth effect is almost gone. However, the filter is dominated by the GPS measurements. Increasing their covariance results in Figure 4.20, where the odometry dominates and its drift can be seen. It is interesting to evaluate the effect of the RSSI in this particular case, and thus we increase the covariances of both RSSI and GPS. The result is a trajectory only containing odometry, similar in shape to Figure 4.20 but with higher error, indicating that the received signal strength has a beneficial effect here. Table 4.8 shows the errors and deviations for all configurations.

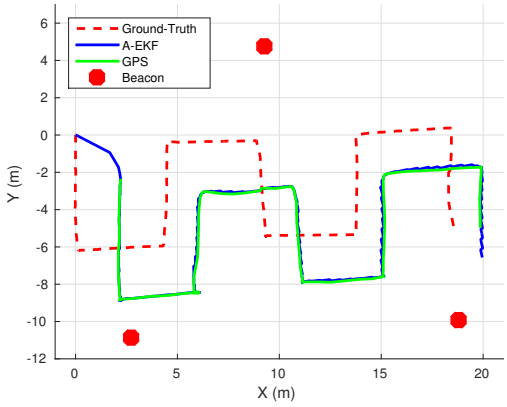


Figure 4.19: A-EKF trajectory for C_{wo} using all sensors

For the $(A, B, C)_w$ datasets, Figure 4.21 shows the trajectory for A_w with the A-EKF and all sensors. As before, there is improvement in the trajectory but with heavy reliance on the GPS measurements. Increasing the GPS covariance gives us the best result for this project, seen in Figure 4.22, which follows the ground-truth almost perfectly. To verify the effect of the RSSI, we use the filter only with odometry, resulting in a slightly more erroneous

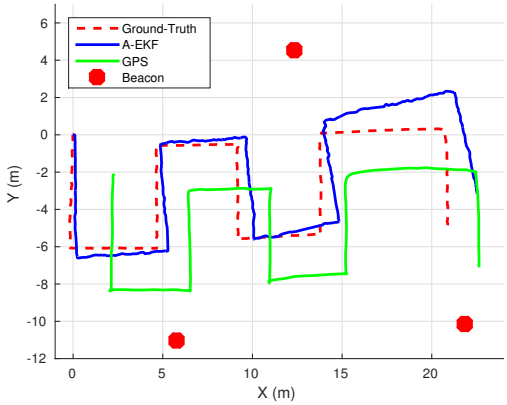


Figure 4.20: A-EKF trajectory estimation for C_{wo} without GPS

Table 4.8: A-EKF localization errors for the $(A, B, C)_{wo}$ datasets

	A_{wo}	B_{wo}	C_{wo}	Mean Error (m)	Deviation (m)
GPS	2.739	1.728	2.890	2.452	0.483
All Sensors	2.743	2.969	2.876	2.853	**0.079**
Odom+RSSI	1.116	2.536	1.108	**1.587**	0.633
Odom Only	1.190	2.623	1.237	1.683	0.626

trajectory. This means that the beacons are indeed important in this solution. Table 4.9 presents the errors and deviations for these configurations.

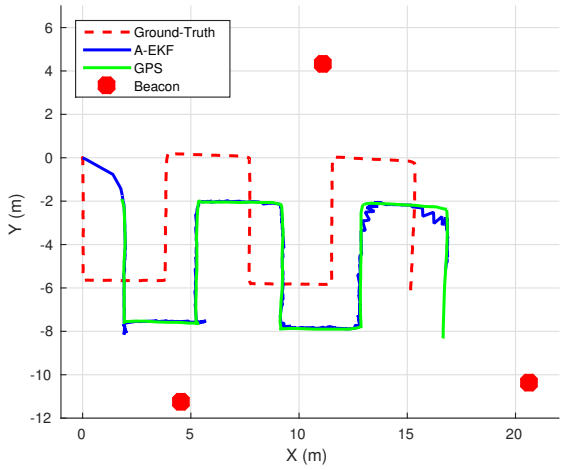


Figure 4.21: A-EKF trajectory for A_w using all sensors

Table 4.9: A-EKF localization errors for the $(A, B, C)_w$ datasets

	A_w	B_w	C_w	Mean Error (m)	Deviation (m)
GPS	2.650	2.226	2.270	2.382	0.179
All Sensors	2.716	2.233	2.523	2.491	0.172
Odom+RSSI	0.343	0.405	0.490	**0.413**	0.052
Odom Only	0.620	0.589	0.531	0.580	**0.033**

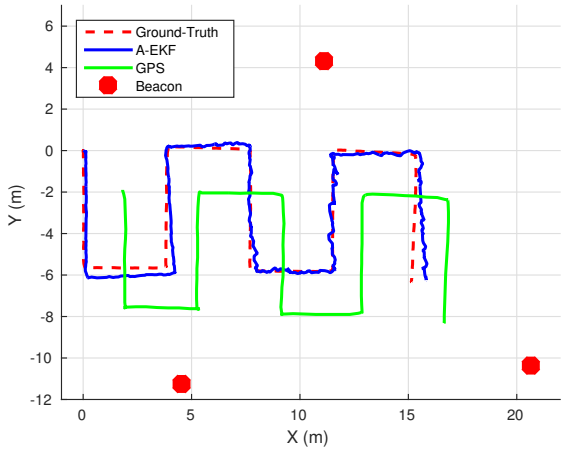


Figure 4.22: A-EKF trajectory for A_w without GPS

Thus, the received signal strength measurements can be used with the Augmented Extended Kalman Filter to improve localization, relative to the stand-alone USB GPS. The best results are obtained using the absorption plate, coherently with the conclusions on Section 4.1.5. This can be confirmed by comparing the errors for the Augmented EKF in Tables 4.8 and 4.9, where the mean error of the best configuration (odometry+RSSI) is reduced by 33.37%. The most statistically significant results (i.e. with the lowest deviations) are respectively the A-EKF with odometry only, and the A-EKF with odometry and RSSI, although both deviations are very similar.

Comparing the two Kalman Filters, the received strength measurements are detrimental to the pose estimation in the EKF because of the attenuation present in the electromagnetic signals. As such, they can only be used properly with the A-EKF, where the received signal strength biases are estimated and corrected. Figure A.17 in Appendix A shows the unbiased RSSI for each beacon, and we can see that they are centered around the ideal values. However, the bias and pose estimations are intrinsically connected, and as such the RSSI values still need to be fused with another sensor to produce a good pose estimate and in turn a good bias correction. If the position estimation contains error, this affects the bias correction and vice versa, accumulating error in each iteration and causing the filter to diverge. Furthermore, the deviations for both filters tell us that even when the error is higher, their estimations are more consistent than using only the GPS.

4.2.2.1 Error Boundedness of the Augmented Extended Kalman Filter

Typically, localization systems affected by drift fuse a measurement with bounded maximum error (i.e. GPS), to ensure that the estimation error will not grow past a certain threshold [28, p. 122]. As such, an interesting analysis to be made here is if the RSSI measurements act as a bounding agent to the drifting odometry in the Augmented Extended Kalman Filter. The received signal strength is not a dead-reckoning measurement, and as

such there is no evidence suggesting that its error is unbounded. However, as concluded in Section 4.2.2, the RSSI needs to be fused with another sensor in the Augmented EKF for its bias to be corrected. If the other sensor has a high noise factor, the bias correction will be wrong, and thus the error boundary for the RSSI would grow accordingly.

To test this, we can use the stationary dataset's received signal strength data for B_1, B_2 and B_3 with the absorbing plate. As this dataset was recorded with the robot stationary at the initial position, let us assume that the ground-truth now is always the origin of our coordinate frame, $(x_G, y_G) = (0, 0)$.

Since the robot is standing still, the odometry must also report $(x, y) = (0, 0)$. We fuse these constant odometry measurements with the three RSSI in the Augmented EKF, using the same covariance values as before. Thus, any error present in the A-EKF pose estimation is due to the beacons alone. Figure 4.23 shows the result for this experiment. Excluding an outlier, the estimations are distributed around the ground-truth with error in the order of micrometers. This result demonstrates that the A-EKF with RSSI has its error bounded to that area.

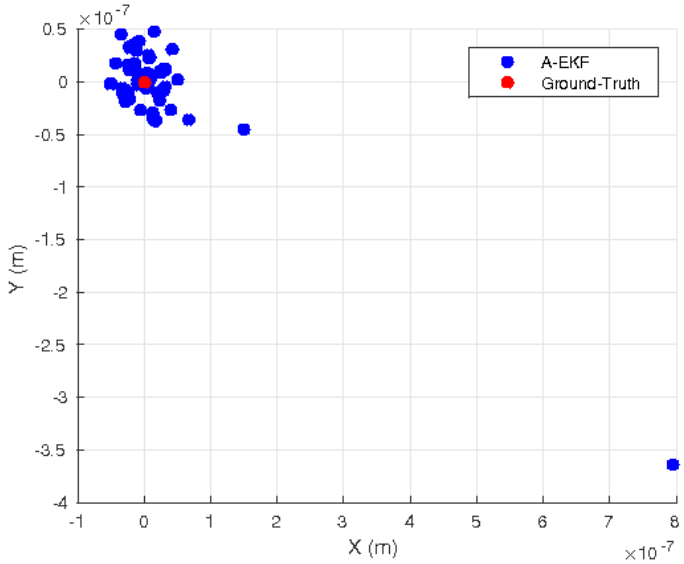


Figure 4.23: Error distribution for the A-EKF with perfect odometry

However, odometry measurements are not perfect and suffer from drift, as verified in Section 4.1.3. To simulate this, we can add a random normally distributed variable to the odometry's x and y, multiplied by a noise factor:

$$x_{odom} = 0 + R_{norm} \times n_f \tag{4.13}$$

$$y_{odom} = 0 + R_{norm} \times n_f \tag{4.14}$$

where R_{norm} is the random normally distributed numbers and n_f is the noise factor. This way we can control the influence of the odometry's noise in the A-EKF estimation. Figure 4.24 shows the result for several values of noise factor.

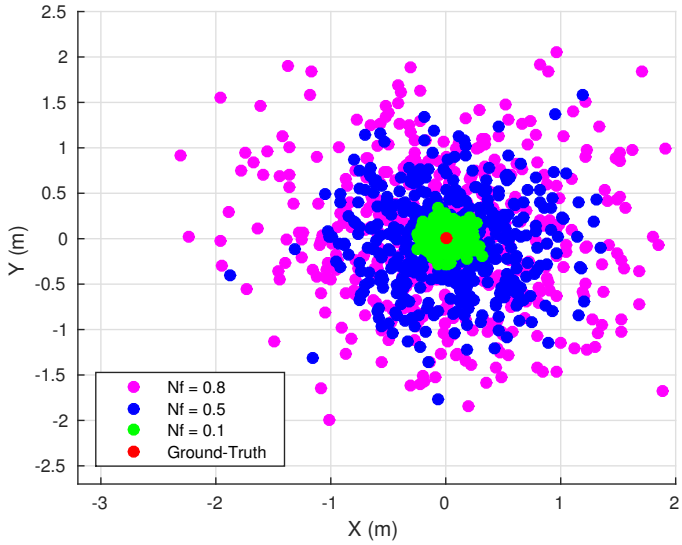


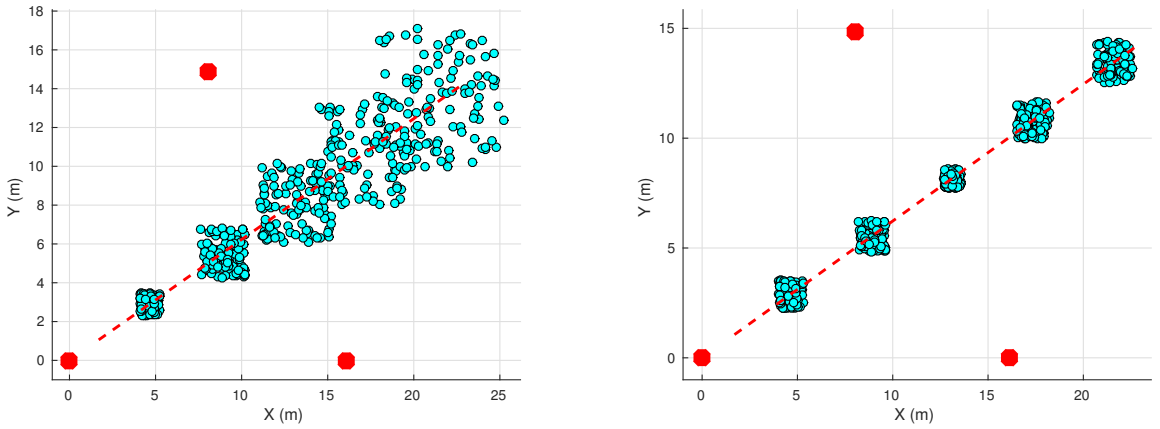
Figure 4.24: Error distribution for the A-EKF with different noise factors

As we can see, the error boundary for the localization solution grows with the noise factor. For the odometry, its error is unbounded because of drift, and so the error for the fused RSSI and odometry in the A-EKF is unbounded as well. Thus, a drifting measurement such as odometry increases the error boundary indefinitely, and the RSSI in this case does not act as a bounding agent. Nevertheless, the received signal strengths still have a positive effect on the pose estimation by restricting the growth rate of the odometry, as seen in Figure 4.22.

4.3 Localization error in an uncontrolled environment

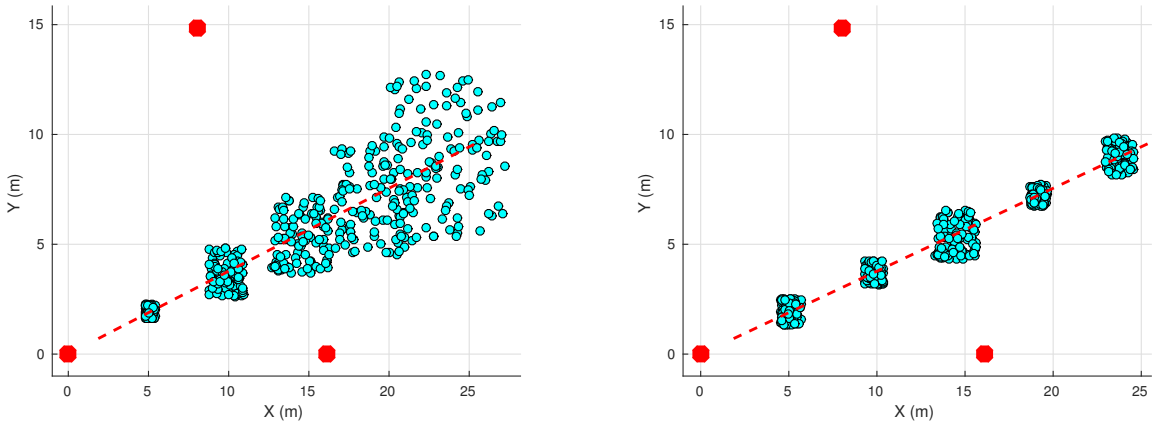
The second experiment was concerned with evaluating the effectiveness of the localization system in different environments. To this end, once again six datasets were collected, three in the parking lot and three in the garden environment, all of them with the absorbing plate equipped. The robot would travel in a straight line for 28 meters, and once again all of its sensors would be recorded. As the RTK GPS did not work in the garden, the ground-truth had to be mathematically derived, knowing that the robot performed a straight line trajectory marked on the ground.

The results for the parking lot can be seen in Fig. 4.25, and the results for the garden environment can be seen in Fig. 4.26. The error propagation for the odometry and A-EKF in each environment are compared. As expected, the odometry estimation error grew much faster relative to the A-EKF. Although unbounded, the proposed localization



(a) Odometry (b) A-EKF

Figure 4.25: Comparison between odometry and A-EKF errors for the parking lot



(a) Odometry (b) A-EKF

Figure 4.26: Comparison between odometry and A-EKF errors for the garden environment

system helped to reduce the rate of error growth. Furthermore, the A-EKF solution worked consistently in both environments, evidenced by comparing Fig. 4.25b and Fig. 4.26b, indicating that it could be successfully used in the HRATC arena. These results were verified on all datasets.

5. LITERATURE REVIEW

This Chapter presents a literature review of mobile robot and wireless sensor network localization. We discuss recent works in the areas, reviewing their localization solutions and the challenges involved. The works related to mobile robot localization with RF signal strength are also discussed, as this is the theme of this project. As the literatures for mobile robot localization and wireless sensor networks are vast, only a few interesting and relevant works were selected.

5.1 Mobile Robot Localization

In the field of mobile robotics, vastly different approaches have been attempted in order to solve the localization problem: from single sensors to probabilistic fusion, vision-based localization, multiple hypothesis tracking, etc. A solution is seldom completely adequate to different applications. The requirements, operating conditions and operating environment dictate the localization system to be used. For example, water quality monitoring with autonomous boats on lakes do not require highly accurate solutions, as usually there are few obstacles to avoid. However, water quality monitoring on an urban waterway requires more accuracy, as there are nearby walls and obstacles for the boat to collide. Thus, the requirements of the localization solution change with the operating environment, for example. This makes it difficult to define a single state-of-the-art for mobile robot localization, as there is no unified metric to evaluate and compare all different systems and applications. Each one has advantages and disadvantages, and a solution that is highly accurate in one specific application may be suited only to that application.

Localization methods based on the Bayes Filter have been widely used in mobile robotics, in some cases becoming the *de-facto* algorithms for localization, with the most famous example being the Extended Kalman Filter. While the EKF has been used extensively in pose estimation for mobile robots, it has also found usage in error correction, to eliminate the effect of external disturbances in challenging environments.

The Unmanned Surface Vehicles (USV), or autonomous boats, are good examples of robotic platforms whose localization suffers greatly due to environmental effects. Their operating environments (i.e. rivers, lakes, ocean) have considerable challenges not found on ground robots. For example, wind blowing on the boat's body can push it sideways on the water's surface. This violates two assumptions of the localization systems proposed in this thesis: First that the robot can only move forward/backward and rotate, and second that external disturbances do not change the robot's pose. Yet, the Kalman Filters are widely popular in USV localization, due to their optimal nature and their capacity to correct biases

and external disturbances. For example, the works of Tuna *et al* [31] and Dunabin and Grinham [10] use linear Kalman Filters for the localization of autonomous boats. The second uses a Kalman Filter for sensor fusion between a GPS and a digital compass, where the pose of a boat is estimated with mean error of 2.9 meters. The first implements a coupled filter architecture in simulation, where measurements from the global positioning system and an inertial navigation system are passed through individual Kalman Filters and then combined for a final pose and velocity estimation. The GPS filter is used to estimate pose and velocity according to the GPS, and the Inertial Navigation System filter is used for estimating and correcting the pose error. Mean errors of less than 1 meter are achieved [31, p. 6].

In [21], Matos *et al* develop an aquatic capsule for ocean Search-and-Rescue operations. The authors implemented a three-stage system for localization, depending on the operating scenario of the capsule. The first stage uses a highly accurate GPS and is the main form of localization, with error of approximately 5 mm. The second stage is when heading information is needed, and thus the initial pose estimate is fused with data from an IMU, which is also beneficial in situations where GPS measurements are unreliable or not available. Finally, the third state introduces an Extended Kalman Filter for yaw estimation and correction, when more accuracy is needed.

In the area of probabilistic robot localization, the Extended Kalman Filter is one of the most used variations of the Kalman Filter, as most robot movement models are nonlinear. However, the implementations of the EKF for different robot types can almost never be reused, as the sensors and the movement model change. For example, the filters implemented in this thesis could not be reused in the pose estimation of a drone, as our filters assume 2D movement. In light of that, Moore and Stouch [22] propose a ROS package with the infrastructure for modular EKF implementation, called *robot_localization*[1]. This package implements a generalization of the filter, where any number of sensors can be fused, specified by the user. The filter assumes the states are the robot's pose and velocities in 3D, and the package allows the user to input the Q and R matrices as ROS parameters in an YAML file. In spite of facilitating code reuse, the generalized structure of the EKF in this package leads to two limitations. First is regarding the state-space itself, as the Augmented EKF implementation seen in this thesis would not be possible, since the state-space of the package can not be changed. To do this, the source code would have to be rewritten, defeating the purpose of generalization. Second is regarding the sensor measurement models. The package assumes that the robot's sensors can produce measurements in the form of the states, and thus the measurement model Jacobian H is an identity matrix. There is no way to use alternative measurements directly, as is the case of the RF received signal strength measurements proposed in this thesis.

As discussed in Section 2.1.3.6, an interesting application of the Kalman Filters is for the estimation and correction of external disturbances. This is done in the work by

[1] http://wiki.ros.org/robot_localization

Caccia *et al* [6], where a two-stage localization system for an autonomous boat is developed. First, a linear Kalman Filter estimates the boat's pose using a linear motion model and a GPS. Second, an Extended Kalman Filter is used with non-linear models of surge and yaw dynamics, to correct the previous pose estimate in relation to these effects. This circumvents the effect of external disturbances in the pose associated with water dynamics (i.e. water currents, eddies, waves).

In general, the localization methods based on the Bayes Filter (namely the Extended Kalman Filter) are widely used for pose estimation and have been used in creative ways with autonomous boats, to compensate for water dynamics for example. One common thread between USVs and other robotic platforms is that they often depend on GPS to limit the error of the pose estimation of the filters. For example, a wheeled robot may use odometry, which suffers from drift. To bound the odometry error, these measurements are fused with an absolute positioning system. However, as discussed in Section 2.1.2.3, there are several situations where GPS measurements are not available, and as such there is a strong trend towards developing localization systems that are independent from this sensor. This is an important issue especially for the boats, since they do not have many suitable sensors for pose estimation and thus rely heavily on GPS. The disadvantage of the Extended Kalman Filter is that it is a dead-reckoning method, capable only of position tracking. As such, the estimation error has a negative effect, causing the filter to drift if the error is not bounded.

5.2 Wireless Sensor Network Localization

The localization of unknown nodes in wireless sensor networks has been a topic of study for many years. They have been successfully used in indoor localization applications such as inventory management and asset tracking, where the objective is to know the location of products in warehouses or equipment in hospitals, for example [19]. Wireless Sensor Networks are becoming pervasive due to the widespread usage of wireless devices, and as such have the potential to be used for mobile robot localization in scenarios where conventional sensor measurements are not available.

Liu *et al* [19] survey the different techniques and systems for performing indoor localization in WSNs. The challenges of this process are discussed, namely the difficulty on modelling radio propagation in indoor environments due to multi-path interference, low availability of line-of-sight and environmental parameters such as dynamism and the presence of reflective surfaces. According to the survey, there are three major categories of localization algorithms, inside which there are subclasses of methods based on the property exploited. The *distance-based* methods are of interest here, as they locate the target through the distances between it and the wireless nodes. The distances may be obtained with time

measurements (i.e. Time-of-Arrival, Time-Difference-of-Arrival) or based on the Received Signal Strength Indicator (also known as Signal Attenuation-based methods). Both distance estimation methods have disadvantages: There are strict synchronization requirements in some time-based algorithms, and all of them require hardware able to measure time differences in the order of picoseconds. In the case of RSSI-based algorithms, the path-loss models often disagree with the actual measurements due to interference and shadowing effects, which can be circumvented with multiple sources of measurements or by employing pre-processing techniques. The authors evaluate the solutions proposed by several works according to six criteria: Accuracy, precision, complexity, robustness, scalability and cost. For example, Battiti *et al*[5] propose a localization solution based on a *Multi-Layer Perceptron*[2], a form of Artificial Neural Network, with three input layers, eight hidden layers and two output neurons. With five sources of received signal strength, a mean error of 3 meters is achieved, while an error of 1.5 meters can be obtained by increasing the number of samples in neural network's training set. In contrast, the proposed multi-layer perceptron architecture with eight hidden layers presents a restriction in the number of parameters that must be stored for the network, possibly prohibiting its use in computers with low memory.

Although RSSI-based techniques usually have worse results in localization, they are popular because of the low cost and ease of implementation. Sichitiu *et al* [26] propose a distributed algorithm for wireless node localization based on inaccurate distances. The algorithm works by each node receiving a transmission packet from a neighbor node, from which the receiving node derives its RSSI. This measurement is used for estimating the position of that node. If the position has improved in regards to the last estimate, the node broadcasts this position to its neighbors, which in turn will refine their own estimates based on this new and improved position. As such, this becomes a collaborative localization system, where each node contributes to the convergence of localization for the whole network. The authors test this method in simulation, using 35 unknown and 8 known nodes spread over an area of one square kilometer, and each node has a transmission range of 300 meters. After obtaining the RSSI, the nodes are capable of self-localization with error of 25 meters. We have too keep in mind that this error is in the case of long-range distance estimation. Compared to the maximum range, 25 meters represents an error of approximately 8.3%. A limitation of this algorithm is that it only works reliably outdoors, as the walls and objects of an indoor environment introduce non-linearities, noise and other uncertainties in the RSSI measurements.

Another example is by Alippi and Vanini [2], where the authors propose the localization of an unknown WSN node through the construction of a topological map for the network, which is used to construct a path-loss model to be used with a Minimum-Least-Squares algorithm for position estimation. An interesting contribution is that the construction of the topological map happens iteratively, and as such this system can be used with semi-static

[2]https://en.wikipedia.org/wiki/Multilayer_perceptron

wireless networks. In other words, the localization system tolerates slow movements on the known nodes. The authors test the system in an outdoor environment with 20 known nodes, spread over an area of 500 m^2 in a football field, and a mean error of 2.3 meters is achieved. The authors identify the main causes for error in the RSSI measurements, namely multi-path interference due to signal reflection in the soil, imperfections on the path-loss model used and slight differences in antenna orientation. In light of that, the authors conclude that the RSSI measurements are unsuitable to be used in applications that require errors smaller than 2.3 meters. The identified causes of error can be considered as limitations of this work, since they can be overcome with hardware modifications to the antennas, as will be discussed in Section 4.1.5. Another limitation is the elevated number of beacons needed for the reported estimation error.

5.3 Related Works

Using WSN localization as an alternative positioning method in mobile robots seems logical, since robots have a requirement for wireless communication and as multiple-robot applications are becoming ever more popular. However, this area of application is not widely explored, and the received signal strength is not considered a suitable measurement for localization. As such it is difficult to find works that directly relate to this project.

An example can be found in the work by Awad *et al* [3], where RSSI-based localization is discussed and two methods for distance estimation are proposed. The first consists on linear and exponential regressions, and the second involves using a Neural Network to predict the optimal curve that describes the characteristics of RSSI with distance. The two methods are used to perform the multi-lateration of a real mobile robot in a controlled indoors environment, with mean errors of 0.5 and 1 meter respectively. The authors also identify several parameters that need to be considered when developing an RSSI-based localization system, namely: Transmission power, frequency of the radio wave, antenna characteristics, the suitability of the localization algorithm and the quality of the known node positions. The most important is the selection of the adequate transmission power, as a power too high leads to decreased sensitivity to distance changes with different RSSI values.

Cheng *et al* [7] propose a robot localization algorithm for use in indoor environments, based on a mix of RSSI and Time-Difference-Of-Arrival measurements. First, the robot's received signal strength is measured in relation to fixed nodes and filtered through an Iterative-Recursive Weighted Average Filter. This data is then fitted to a polynomial curve, which is used to predict the distance between receiver and transmitter. Finally, the robot pose is estimated by merging the distance information with TDOA measurements through a Maximum Likelihood Estimator. The authors performed tests in an indoor environment with a wheeled robot, obtaining a mean error of approximately 0.5 meters.

In [25], Raghavan *et al* develop a localization system for indoor mobile robots using Bluetooth received strength measurements. Once again, a study on important characteristics of RSSI was performed, and the authors devised a modified multi-lateration algorithm in which gradient descent is used to find the position with minimal estimation error. The output of this multi-lateration is then used in a Particle Filter (PF) algorithm, and a mean error of 0.427 meters is observed, evaluated with a differential drive robot and three Bluetooth dongles in known positions. This result is interesting because no other sensor data is fused in the particle filter, and the RSSI measurements are the only sensor data used. As for limitations, the authors report that the presence of obstacles in front of the transmission path have an effect on localization, but we can also mention that there is a range restriction due to the short communication distances of the Bluetooth protocol. The authors use class 2 USB dongles, with a reported maximum range of 10 meters.

However, the results found in [25] are for indoor robots. The work that most closely approximates the goal of this project, solving the localization problem in an outdoor environment, can be found in [15]. Graefenstein and Bouzouraa propose a solution to outdoor robot localization by fusing RSSI from an XBee radio, odometry and laser rangefinder measurements in a Particle Filter. To improve the uniqueness of the received signal strength measurements and to lessen the effects of interference and noise, the authors thoroughly developed a special antenna with an absorbing plate at its base, which greatly improved the quality of the RSSI measurements. To estimate the distances between receiver and transmitter, the Free-space Path Loss model is used, assuming that there are no shadowing effects on the operating environment. The authors test the algorithm with a lawnmower robot in a garden using 11 known RF nodes, and a mean error of 0.32 meters is achieved with the Particle Filter, using 2000 particles. The hardware modifications to the antenna proved fundamental in reducing the mean error of the localization system, and as such these modifications will be employed here. The disadvantages of this solution are the heavy computational burden of the Particle Filter with 2000 particles, as well as the elevated number of beacons needed. The authors do not detail the test environment, only stating that it is an outdoors garden with 140 square meters, approximately 11.83 meters in length considering a perfectly square area.

Table 5.1 presents a brief comparison between the work in this project and the related works. The "localization system" row relates to what methods are used for distance and pose estimation respectively. The best result for our work is presented, which will be discussed in Chapter 4.

Each related work share similarities to ours in different aspects, but the most similar ones are [15] and [25], while the least similar is [7]. The work of Graefenstein and Bouzouraa is the main reference for this project, since it is the most similar in hardware, environment and application. The advantage in our work is that we use fewer known nodes, a simpler absorption plate and a computationally cheaper pose estimation algorithm compared to the

Table 5.1: Comparison between related works

Reference	Localization System	Environment	Sensors Used	# of known nodes	Mean Error (m)
[3]	Regression/Neural Net. + Multi-Lateration	Indoors	RSSI	9	0.5 (Reg.) 1 (N.N.)
[7]	Iterative-recursive filter + Regression + Maximum Likelihood Estimator	Indoors	RSSI TDOA	6 RSSI 2 TDOA	0.5
[15]	Free-Space Path Loss + Particle Filter	Outdoors	RSSI Encoders Laser rangefinder	11	0.32
[25]	Multi-Lateration with Gradient-Descent + Particle Filter	Indoors	RSSI	3	0.427
Proposed Method	Free-Space Path Loss + Augmented Extended Kalman Filter	Outdoors	RSSI Encoders IMU	3	0.413

Particle Filter, allowing for the use of this solution in embedded computers, for example. A comparison of the computational complexities for the PF and for the solution proposed here is presented in Chapter 6. The disadvantage is that [15] solves global localization with the Particle Filter, while only position tracking is tackled here. As such, Graefenstein-Bouzouraa's solution does not suffer from drift and is more robust to uncertainty. Raghavan *et al* [25] also achieve interesting results, attaining a similar mean error to ours while using only RSSI and the same number of known nodes in an indoor environment, which is notably more difficult for WSN localization. The limitations are the use of Bluetooth as the wireless communication protocol, notorious for its short range, and that breaks in line-of-sight between receiver and transmitter have an effect on localization.

6. CONCLUSIONS

This project proposed a localization system to solve the problem of trajectory estimation of a mobile ground robot in environments with little or no GPS signal available. The system uses three Radio-Frequency transmitters which communicate with a receiver on the robot, and the Received Signal Strength Indicator for each beacon is used along with other sensors in the Extended and Augmented-Extended Kalman Filter algorithms. To evaluate the quality of localization with these algorithms, a robot was constructed and several datasets of its sensors were collected in an outdoor urban environment, while the robot performed a predefined trajectory. The robot's pose is obtained in the Kalman Filters by fusing the received signal strengths, linear velocities obtained through encoder measurements, angular velocity from an Inertial Measurement Unit and global position observations from a regular GPS device. A stand-alone Real-Time Kinematics module is used to log the robot's ground-truth, and a mean localization error of 41.3 cm. The localization system is tested on two areas, a parking lot with clear sky-view and a garden with tall trees and a nearby building, presenting consistent behavior between the areas, indicating that the proposed localization system can be used successfuly in the absence of GPS.

The overall best result, with error of 34.3 cm (Table 4.9, column A_w), was achieved with the Augmented Extended Kalman Filter using only odometry and the enhanced RF beacons, independently from the GPS measurements. The best mean error for the three datasets with the RF enhancements was 41.3 cm, an improvement of 82.66% over the baseline error (i.e. GPS only). The statistical significance of this result is also verified by its low deviation. However, special care must be taken in placing the beacons at accurately known positions, as different position errors between them negatively affect localization. In regards to hardware, the use of the absorbing plate in the RF receiver is crucial, as it improves the quality of the RSSI measurements. Even so, the signal strength is highly attenuated by environmental conditions, and thus it can only be used successfully with the A-EKF algorithm, in which their biases and uncertainties are corrected. As this correction depends on the quality of localization (and vice-versa), the received signal strength can not be used alone. Furthermore, the pose estimates will eventually diverge due to the solution's unbounded error, as seen in Section 4.2.2.1. Other sensors must be fused to improve the pose estimation (and by consequence the RSSI correction) and bound the estimation error.

6.1 Contributions

The main contribution of this project is the development of a localization system which can be used independently from GPS, while at the same time not relying on expen-

sive sensors such as LIDAR and rangefinders. Instead, it uses a property of electromagnetic signal propagation to perform beacon-based localization with low-cost radios. Furthermore, it does not need a known map of the environment and thus can be used for localization in open, feature-less environments. Apart from its uses in the HRATC competition, this system could be applied to outdoor robots such as autonomous boats and agricultural robots, as Radio-Frequency transmissions tend to suffer less interference in outdoor rural environments. However, it could possibly work indoors, as RSSI-based localization for indoor environments is a studied field of Wireless Sensor Networks.

This project borrows the RF hardware modifications proposed by Graefenstein and Bouzouraa [15], albeit in a much more simplistic way. In their work, the authors thoroughly designed an antenna for localization, with specific radiation patterns and a special absorbing plate. Here, the radios had simple Wi-Fi router antennas and the receiver on the robot had an improvised absorption plate, thus being much simpler and easier to implement. Nevertheless, we confirm that these modifications were adequate and necessary for improving the characteristics of the received signal strength. Comparing the results, a localization error of 32 cm is achieved in [15] with 11 beacons and a Particle Filter, an algorithm with a high computational load. The best localization error here is 34.3 cm, but using a third of the number of beacons and a less costly algorithm, whose complexity compared to the Particle Filter will be discussed below. As demonstrated here, three beacons are sufficient to approximate the error obtained with the Particle Filter. Another advantage of our solution is that it has no need of a map nor distinct environmental features (i.e. obstacles), two requirements of the Particle Filter seen in [15]. Thus, our localization system can be used in wide, open and featureless environments.

Compared with [15], our localization solution with the Augmented Extended Kalman Filter is computationally cheaper, while at the same time being more computationally complex than the Particle Filter. To understand this paradox, let us consider the complexities of both algorithms. As seen in Section 2.1.3.4, the Kalman Filters are dominated by matrix inversions and multiplications and thus work in quadratic time, or $\mathcal{O}(n^2)$, where n is the number of states in the state-space. The Particle Filter has several stages, of which the most costly is the *resampling* stage. In the best-case scenario, this stage runs in linear time, or $\mathcal{O}(M)$, where M is the number of particles being used [30, p. 87][17].

Considering that a well-conditioned Particle Filter was used by Graefenstein and Bouzouraa, we can see that their solution is less computationally complex than the Augmented EKF presented here. However, the Particle Filter suffers with the *curse of dimensionality*: In order to converge, the PF typically requires a great number of particles, in the order of thousands. Thus, its computational load is larger than the A-EKF's load, as the number of particles M is much larger than the number of states n. This conclusion is further cemented by considering the number of states of the A-EKF, $n = 8$, and the number of particles used in [15], $M = 2000$.

6.2 Limitations

The localization solution presented here is limited by the characteristics of the received signal strength measurement when used with the Augmented Extended Kalman Filter. The RSSI and pose estimation in the A-EKF are strongly coupled due to the co-dependency of the pose estimation and bias correction: Errors in pose estimation affect the bias correction and vice-versa. For this reason, another sensor must be fused in order to restrain the pose estimation and keep the RSSI measurements centered around the true value. While this benefits localization, the coupling effect eliminates the potential for the RSSI to bound the estimation error, as seen in Section 4.2.2.1.

Even though our solution achieves a similar mean error and has a lighter computational load compared to [15], Graefenstein and Bouzouraa's solution is more complete, as the algorithms presented here address the localization problem of position tracking, where the initial position is known. The Particle Filter solves the global localization problem without requiring an initial position, is more robust to noise and uncertainty and does not suffer from drift.

Our solution is also dependant on the robot's environment. Atmospheric conditions, multi-path interference and noise affect received signal strength, corrupting the pose estimation and, in our case, the bias correction with the A-EKF. Depending on the level of corruption, the coupling effect may cause the filter to diverge, even with the hardware modifications to the robot receiver's antenna. Accurately positioning the RF beacons is also a problem. It can be a challenge if the environment has no landmarks clearly visible on Google Maps, where the beacons can be placed. In this case, a preliminary step of manually measuring the local coordinates of the beacons must be taken, which increases the deployment time of the localization solution.

There are two limitations specific to this project's methodology. First, only the triangular beacon setup was tested, and as such there is no evidence that our localization system works in other configurations. For example, if this solution was to be used in an autonomous boat operating on a beach, the beacons would have to be laid out in a straight line on the shore. Secondly, the parking lot where the experiments were run was a well-behaved and controlled environment, without people or cars passing by, and without much nearby buildings and trees. As such, there is no guarantee that our solution would work in the HRATC arena which inspired this project. The original HRATC arena is currently a construction site, and thus it was not possible to test our system in this location.

Finally, the operating area of our localization system is limited to the common area within the communication range of the RF modules. To illustrate, let us consider that the maximum range of the modules describe radii, where the circumferences are the beacons' communication areas. A requirement of our solution is that all RF beacons must be con-

nected to the receiver on the robot, and thus the robot must remain within the communication radius of each beacon, limiting the operating area to the intersection between the three circumferences as seen in Figure 6.1. A possible solution would be to add more beacons, but a disadvantage is that the state-space and measurement vector of the Augmented Extended Kalman Filter increase by one with each new beacon, making the filter more computationally costly and difficult to implement. The theoretical operation area could be calculated considering the maximum range specified in the radio modules' datasheet.

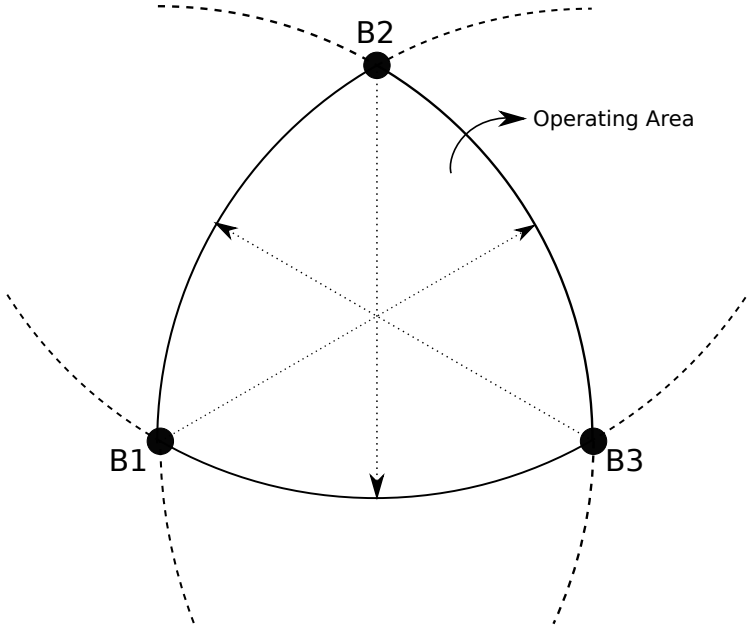


Figure 6.1: Operating area relative to the communication range of the RF beacons

6.3 Future Work

Firstly, the setup that produces the best localization must be tested in HRATC's arena or in a similar environment, to see if in fact this problem is solved for the competition. To do this, a method for acquiring the robot's ground-truth in GPS-deprived environments must be implemented. Apart from that, the Kalman Filters were tested in an off-line manner, with pre-recorded datasets. The next step in this front is to implement them in a ROS package and test them in real-time on the field. This package could potentially be refined and released to the academic community as an alternative localization solution. The implementation of a Particle Filter using the measurements seen here is also an interesting possibility.

Secondly, the coupling problem of the A-EKF can be further explored. As the pose estimation and bias correction are co-dependant, a possible solution is to remove the bias correction from the Kalman Filter and pre-process the RSSI data and correct it before feeding

it to an Extended Kalman Filter. The use of pre-processed data in a Kalman Filter results in a so-called *loosely coupled* filter [29].

Due to this coupling effect, the RSSI needs to be fused with other sensors on the A-EKF in order to work, eliminating its error boundedness. As the Kalman Filter is a dead-reckoning method, eventually the estimations will drift from the ground-truth, and a sensor with bounded error (normally a GPS) must be used to reset the estimation error. Apart from decoupling the A-EKF to restore the RSSI boundedness, as discussed before, other measurements with bounded error could be explored, such as pose estimation through QR codes.

The Trouble robot can be improved by exchanging some of its sensors for better counterparts. For example, the IMU could be replaced by its newer version or by a different model with more precise accelerometers, as the frontal and lateral accelerations from the UM6 could not be used here. This improves the localization quality and at the same time potentially reduces the total cost of the localization system. Changing the mobile base is possibility too, as the setup described here is not dependant on the Pioneer 3-AT base: For example, the XBee receiver could be mounted onto an autonomous boat for bathymetrical surveying, or applications which need a finer degree of localization accuracy. Using this solution on indoor robots is also interesting, as they are GPS-deprived environments.

In regards to the XBee modules, a detailed study of the received signal strength's decay with distance must be performed, in order to better understand the characteristics of this measurement, as well as how its error grows with distance and if it is indeed bounded. Another problem with the modules is that the 2.45 GHz frequency band in which they operate is very crowded, as it is widely used by Wi-Fi, Bluetooth and other commercial applications. This is one of the sources of destructive interference which contribute to the attenuation of the received strength seen in Figure 4.13. Thus, the selection of an RF module with a different frequency would benefit this project. Using lower frequencies are indicated, as the electromagnetic waves are less sensitive to environmental effects and are able to more easily pass through obstacles.

Setting aside the received signal strength, the other properties of electromagnetic signal propagation could be explored, such as Time-of-Flight and Time-Difference-of-Arrival. The advantage of using these measurements is their robustness from environmental conditions in small distances, the finer resolutions possible and the fact that their distance measurement models are simpler. However, there is an instrumentation challenge, as measuring the Time-of-Flight of waves travelling small distances at the speed of light is no trivial task. In some cases, there is also a strict requirement for synchronization between the radios.

The optimization problem of accurately determining the beacon positions could be integrated with the localization system. One interesting idea is to expand the augmented state-space of the A-EKF to include additional states for the (x,y) position of each beacon. The filter then takes care of estimating the optimal positions, as it does with the RSSI biases.

The problem is that the number of states jumps to 14 considering 3 beacons, resulting in a difficult Kalman Filter to implement. The beacon positions could also be estimated in a step prior to localization, with an optimization algorithm.

REFERENCES

[1] Adept Technology Inc. “Pioneer 3-AT”. Captured in: http://www.mobilerobots.com/Libraries/Downloads/Pioneer3AT-P3AT-RevA.sflb.ashx, 19-12-2016.

[2] Alippi, C.; Vanini, G. “A RSSI-based and calibrated centralized localization technique for Wireless Sensor Networks”. In: IEEE International Conference on Pervasive Computing and Communications Workshops, 2006, pp. 1–5.

[3] Awad, A.; Frunzke, T.; Dressler, F. “Adaptive distance estimation and localization in WSN using RSSI measures”. In: IEEE Euromicro Conference on Digital System Design Architectures, Methods and Tools, 2007, pp. 471–478.

[4] Barclay, L. “Propagation of Radiowaves”. Stevenage, Hertfordshire, UK: Institution of Engineering and Technology, 2003, 3rd ed., 482p.

[5] Battiti, R.; Nhat, T. L.; Villani, A. “Location-aware computing: A neural network model for determining location in wireless LANs”, Technical Report, University of Trento, Trento, Italy, 2002, 17p.

[6] Caccia, M.; Bibuli, M.; Bono, R.; Bruzzone, G. “Basic navigation, guidance and control of an unmanned surface vehicle”, *Autonomous Robots*, vol. 25–4, 2008, pp. 349–365.

[7] Cheng, L.; d. Wu, C.; z. Zhang, Y. “Indoor robot localization based on wireless sensor networks”, *IEEE Transactions on Consumer Electronics*, vol. 57–3, 2011, pp. 1099–1104.

[8] Digi International. “XBee/XBee-PRO S2C Zigbee user guide”. Captured in: https://www.digi.com/resources/documentation/digidocs/pdfs/90002002.pdf, 20-12-2017.

[9] Dudek, G.; Jenkin, M. “Computational Principles of Mobile Robotics”. New York, New York, USA: Cambridge University Press, 2010, 2nd ed., 391p.

[10] Dunbabin, M.; Grinham, A. “Experimental evaluation of an autonomous surface vehicle for water quality and greenhouse gas emission monitoring”. In: IEEE International Conference on Robotics and Automation, 2010, pp. 5268–5274.

[11] El-Rabbany, A. “Introduction to GPS: The Global Positioning System”. Norwood, Massachusetts, USA: Artech House, 2002, 1st ed., 194p.

[12] Emlid. “Reach: RTK GNSS module for precise navigation and UAV mapping”. Captured in: https://emlid.com/reach/, 20-12-2017.

[13] Figueiras, J.; Frattasi, S. “Mobile Positioning and Tracking: From Conventional to Cooperative Techniques”. Hoboken, New Jersey, USA: Wiley Publishing, 2010, 1st ed., 298p.

[14] Gebhard, H.; Weber, G. “NTRIP, version 1.0”, Technical Report, Federal Agency for Cartography and Geodesy (BKG), Frankfurt, Germany, 2003, 50p.

[15] Graefenstein, J.; Bouzouraa, M. E. “Robust method for outdoor localization of a mobile robot using received signal strength in low power wireless networks”. In: IEEE International Conference on Robotics and Automation, 2008, pp. 33–38.

[16] Hassanzadeh, I.; Fallah, M. A. “Design of augmented extended Kalman filter for real time simulation of mobile robots using simulink”. In: IEEE International Symposium on Mechatronics and its Applications, 2009, pp. 1–6.

[17] Hol, J. D.; Schon, T. B.; Gustafsson, F. “On resampling algorithms for particle filters”. In: IEEE Nonlinear Statistical Signal Processing Workshop, 2006, pp. 79–82.

[18] Larsen, T. D.; Hansen, K. L.; Andersen, N. A.; Ravn, O. “Design of Kalman filters for mobile robots; evaluation of the kinematic and odometric approach”. In: IEEE International Conference on Control Applications, 1999, pp. 1021–1026.

[19] Liu, H.; Darabi, H.; Banerjee, P.; Liu, J. “Survey of wireless indoor positioning techniques and systems”, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 37–6, 2007, pp. 1067–1080.

[20] Madhavan, R.; Amory, A.; Prestes, E.; Guedes, R.; Bergamin, A.; Neuland, R.; Mantelli, M.; Kindin, D.; Rodrigues, F. “The 2017 humanitarian robotics and automation technology challenge [humanitarian technology]”, *IEEE Robotics & Automation Magazine*, vol. 24–4, 2017, pp. 127–129.

[21] Matos, A.; Silva, E.; Cruz, N.; Alves, J. C.; Almeida, D.; Pinto, M.; Martins, A.; Almeida, J.; Machado, D. “Development of an unmanned capsule for large-scale maritime search and rescue”. In: IEEE OCEANS, 2013, pp. 8.

[22] Moore, T.; Stouch, D. “A generalized extended kalman filter implementation for the robot operating system”. In: International Conference on Intelligent Autonomous Systems, 2016, pp. 335–348.

[23] Nahavandi, S. “Trusted autonomy between humans and robots: Toward human-on-the-loop in robotics and autonomous systems”, *IEEE Systems, Man, and Cybernetics Magazine*, vol. 3–1, 2017, pp. 10–17.

[24] Quigley, M.; Conley, K.; Gerkey, B. P.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A. Y. “ROS: an open-source robot operating system”. In: IEEE ICRA Workshop on Open Source Software, 2009, pp. 1–6.

[25] Raghavan, A. N.; Ananthapadmanaban, H.; Sivamurugan, M. S.; Ravindran, B. “Accurate mobile robot localization in indoor environments using bluetooth”. In: IEEE International Conference on Robotics and Automation, 2010, pp. 4391–4396.

[26] Sichitiu, M. L.; Ramadurai, V.; Peddabachagari, P. “Simple algorithm for outdoor localization of wireless sensor networks with inaccurate range measurements”. In: IEEE International Conference on Wireless Networks, 2003, pp. 300–305.

[27] Siciliano, B.; Khatib, O. “Springer Handbook of Robotics”. Secaucus, New Jersey, USA: Springer-Verlag New York, Inc., 2007, 2nd ed., 2227p.

[28] Siegwart, R.; Nourbakhsh, I.; Scaramuzza, D. “Introduction to Autonomous Mobile Robots”. Cambridge, Massachusetts, USA: MIT Press, 2011, 2nd ed., 472p.

[29] Sirtkaya, S.; Seymen, B.; Alatan, A. A. “Loosely coupled Kalman filtering for fusion of visual odometry and inertial navigation”. In: IEEE International Conference on Information Fusion, 2013, pp. 219–226.

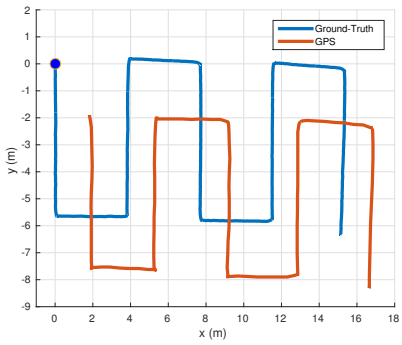
[30] Thrun, S.; Burgard, W.; Fox, D. “Probabilistic Robotics”. Cambridge, Massachusetts, USA: MIT Press, 2005, 1st ed., 672p.

[31] Tuna, G.; Arkoc, O.; Koulouras, G.; Potirakis, S. “Navigation system of an unmanned boat for autonomous analyses of water quality”, *Elektronika ir Elektrotechnika*, vol. 19–8, 2013, pp. 1–5.

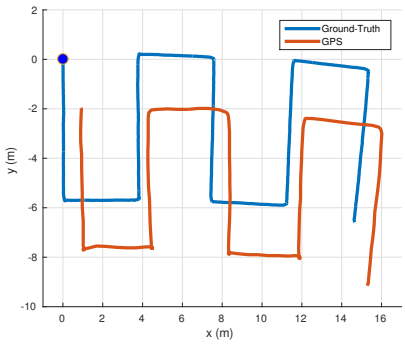
[32] Zeltkevic, M. “Forward and backward euler methods”. Captured in: http://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node3.html, 14-12-2017.

APPENDIX A – ADDITIONAL RESULTS

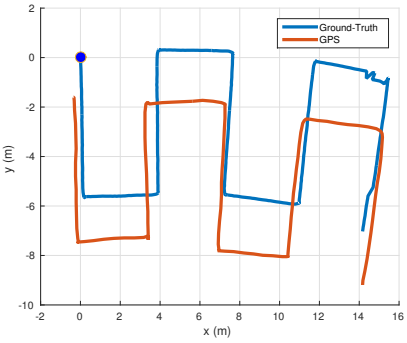
- Global Positioning System Validation (Section 4.1.2)



(a) A_w



(b) B_w



(c) C_w

Figure A.1: GPS and ground-truth trajectories for $(A, B, C)_w$

- Odometry Velocity and Acceleration Validation (Section 4.1.3)

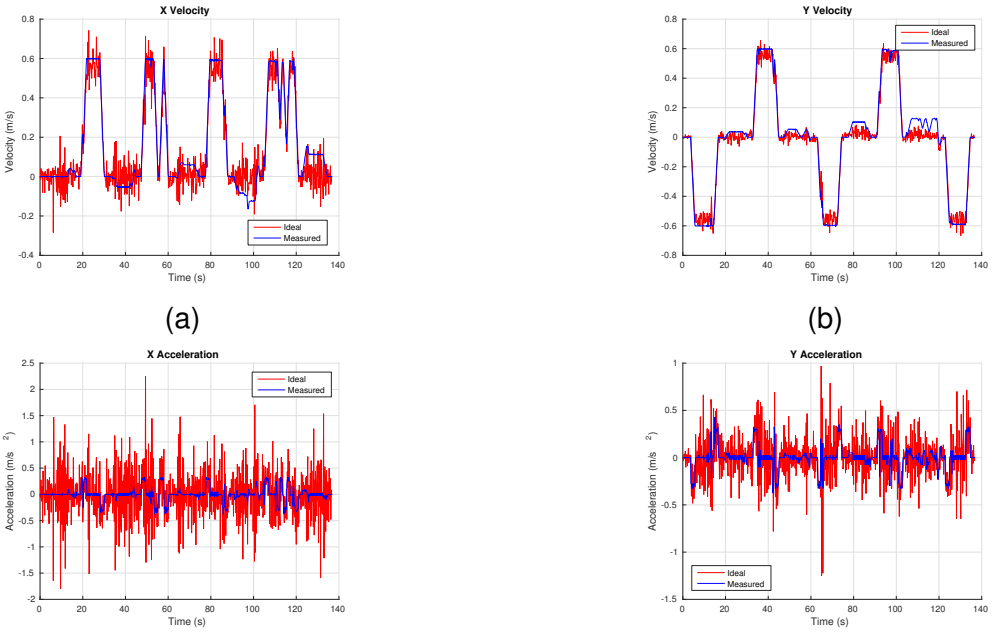


Figure A.2: Ground-truth and odometry velocities/accelerations for the A_{wo} dataset

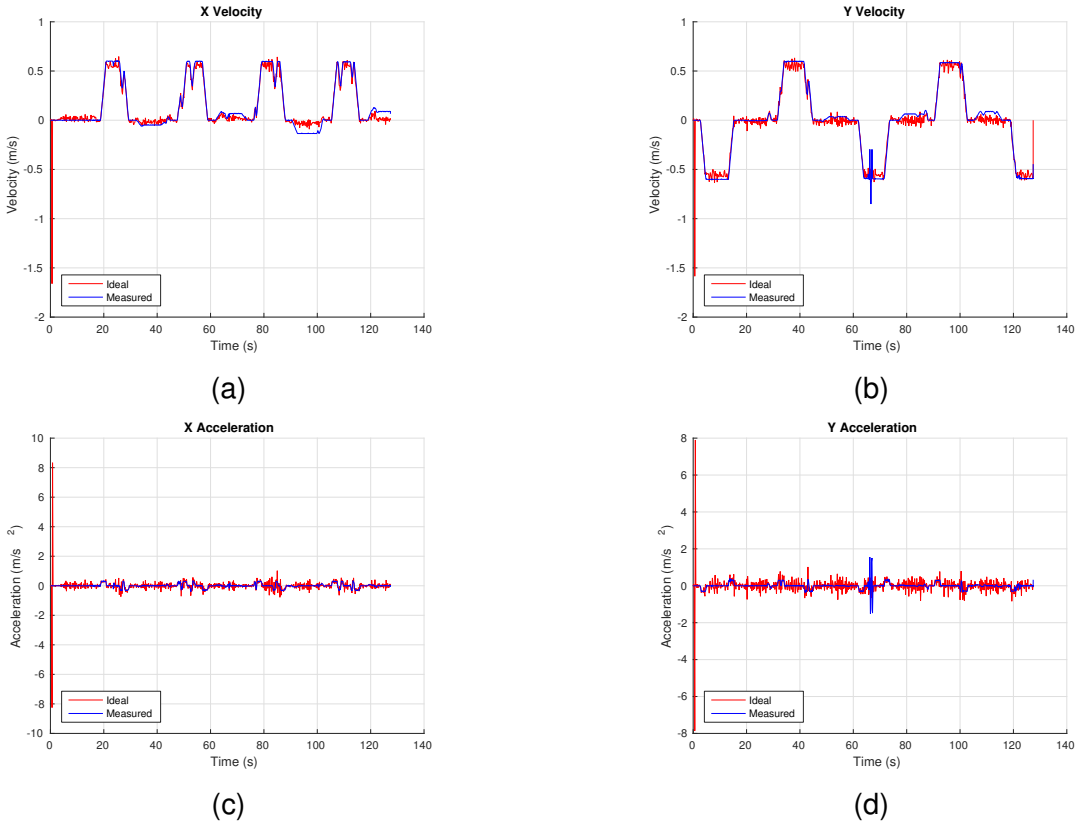


Figure A.3: Ground-truth and odometry velocities/accelerations for the B_{wo} dataset

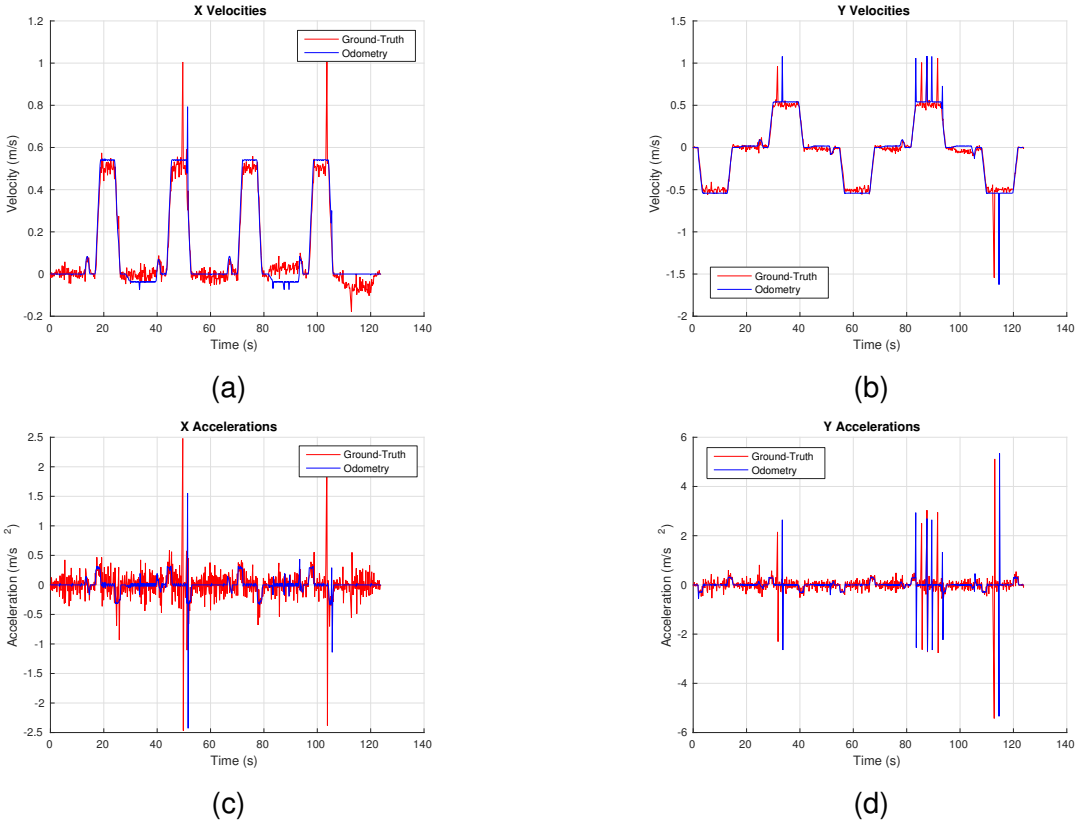


Figure A.4: Ground-truth and odometry velocities/accelerations for the B_w dataset

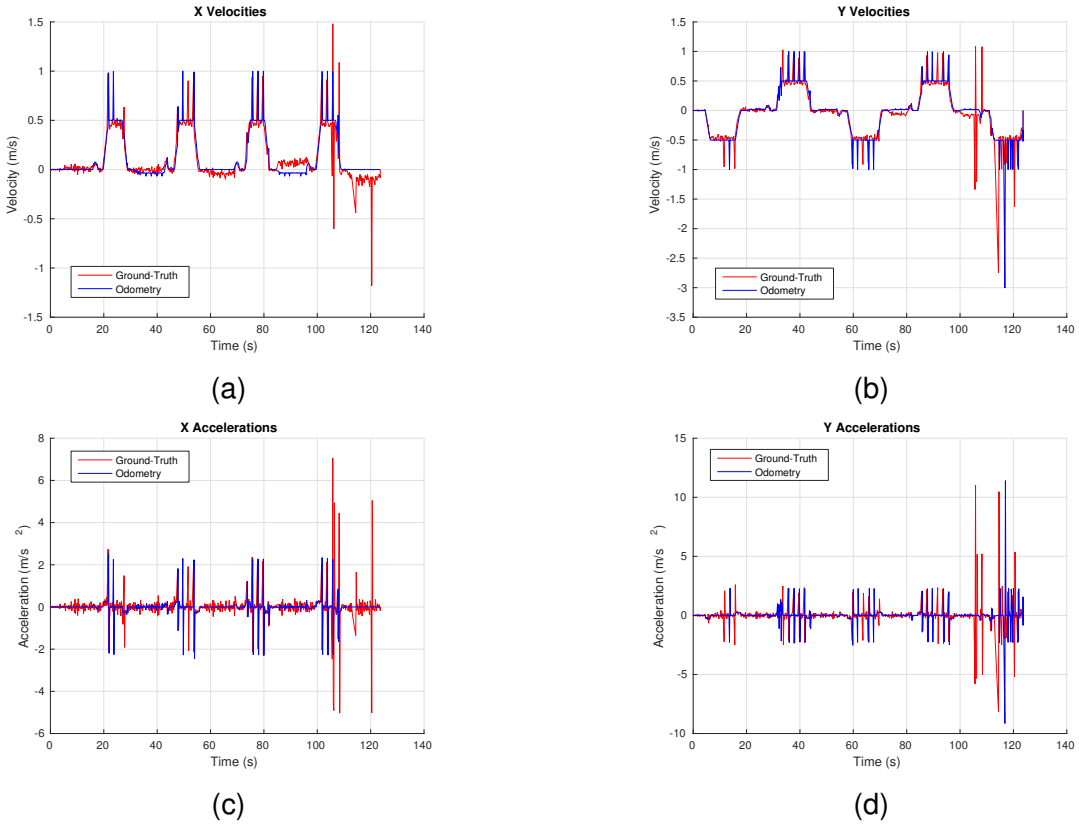
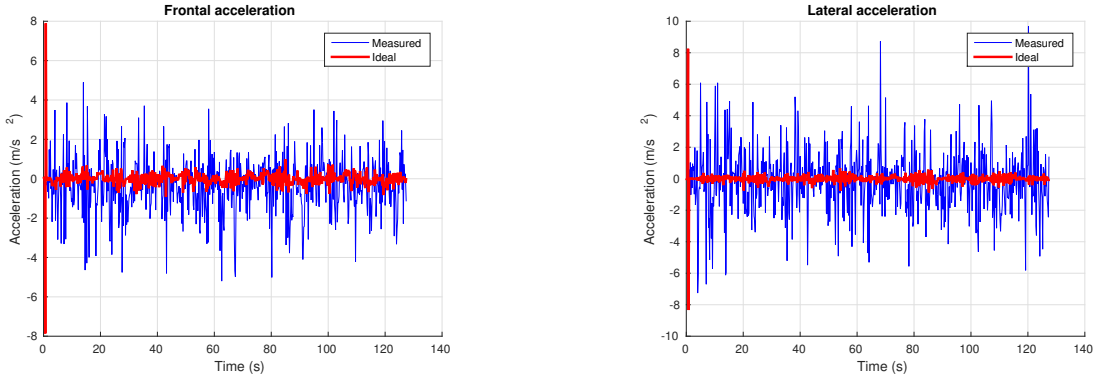


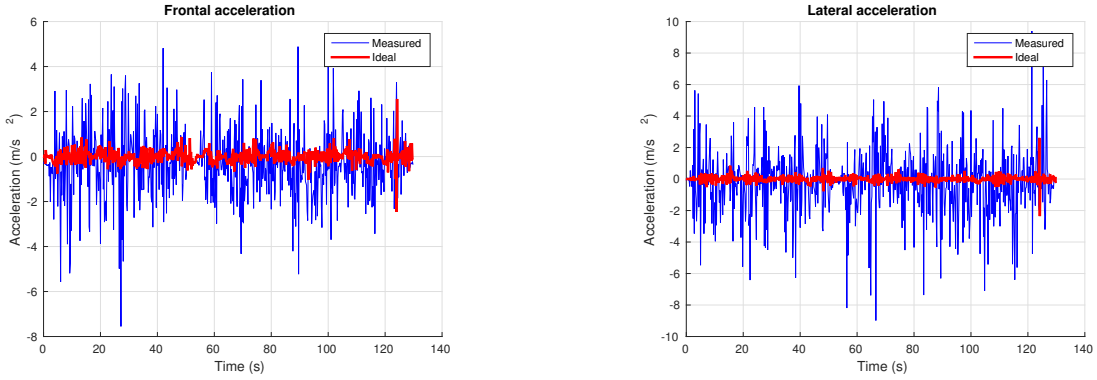
Figure A.5: Ground-truth and odometry velocities/accelerations for the C_w dataset

- Inertial Measurement Unit Validation (Section 4.1.4)



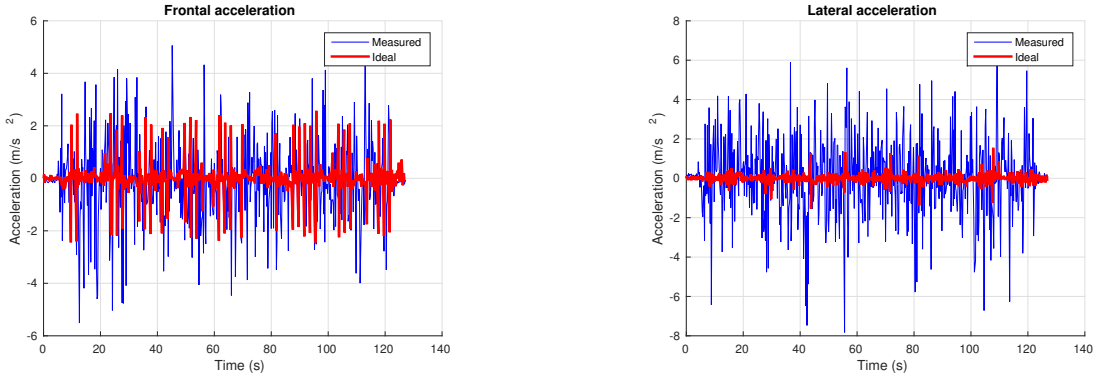
(a) (b)

Figure A.6: Comparison between ideal and measured frontal/lateral accelerations for the B_{wo} dataset



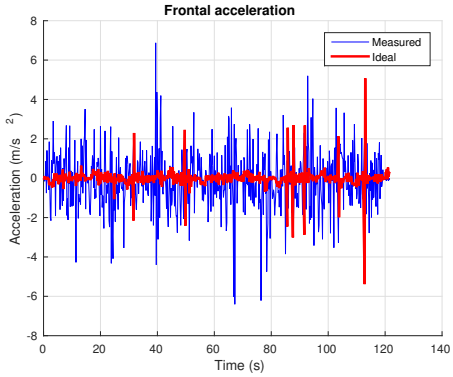
(a) (b)

Figure A.7: Comparison between ideal and measured frontal/lateral accelerations for the C_{wo} dataset

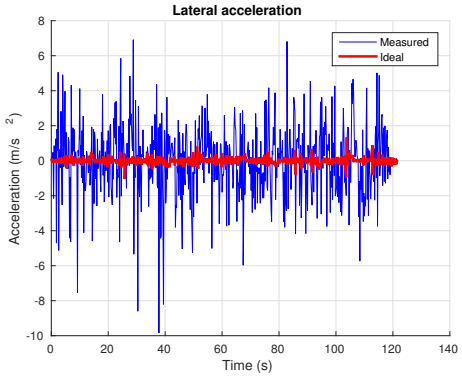


(a) (b)

Figure A.8: Comparison between ideal and measured frontal/lateral accelerations for the A_w dataset

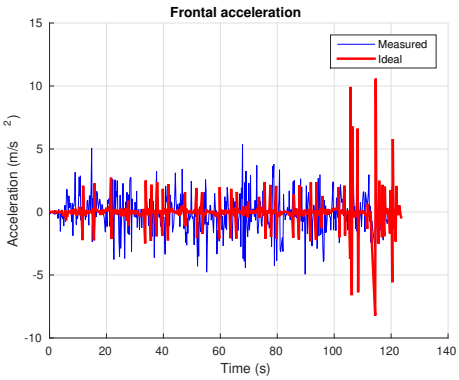


(a)

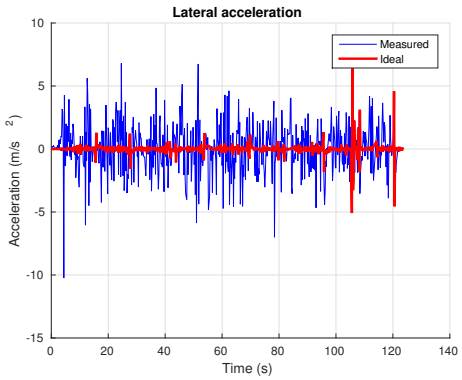


(b)

Figure A.9: Comparison between ideal and measured frontal/lateral accelerations for the B_w dataset



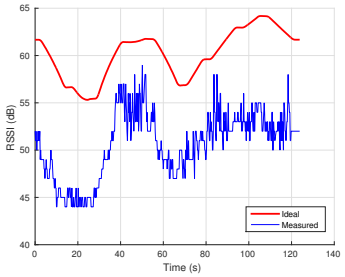
(a)



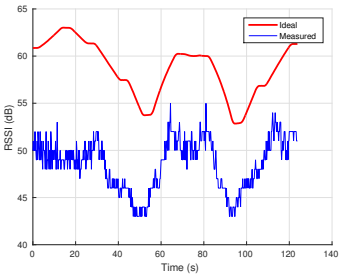
(b)

Figure A.10: Comparison between ideal and measured frontal/lateral accelerations for the C_w dataset

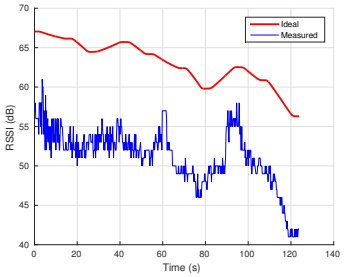
- Received Signal Strength Indicator (Section 4.1.5)



(a) B_1

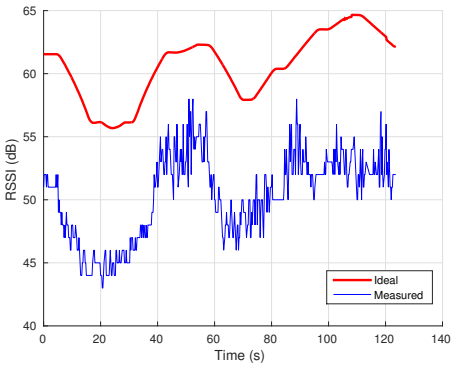


(b) B_2

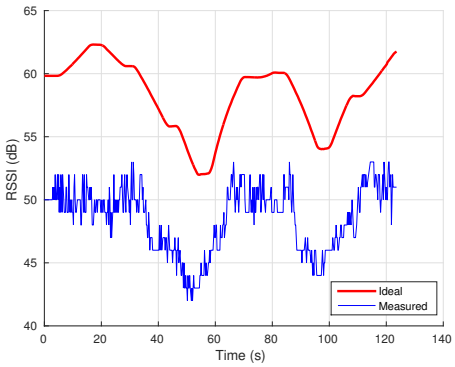


(c) B_3

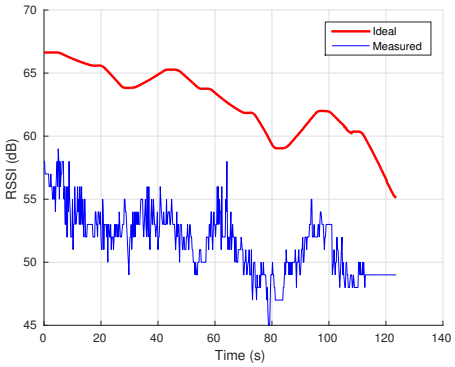
Figure A.11: RSSI value comparison for the three beacons for the B_w dataset



(a) B_1



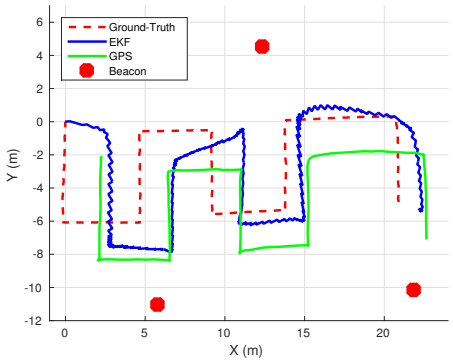
(b) B_2



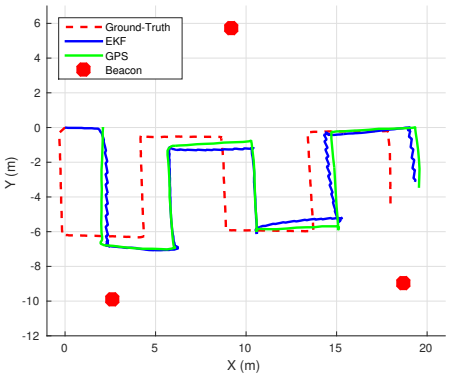
(c) B_3

Figure A.12: RSSI value comparison for the three beacons for the C_w dataset

- Extended Kalman Filter Trajectory Estimation (Section 4.2.2)

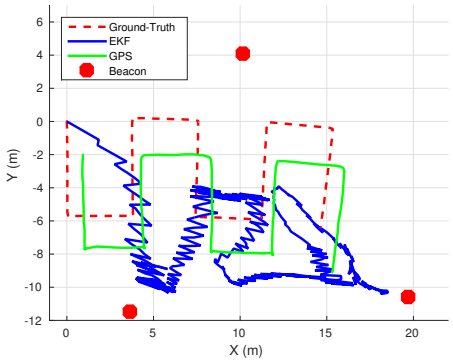


(a) A_{wo}

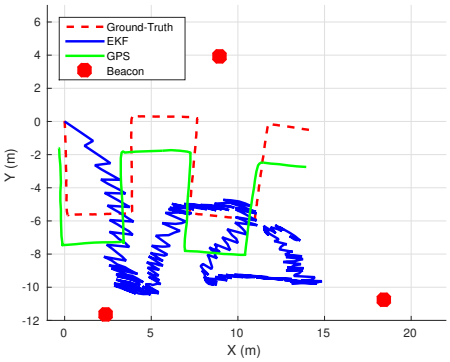


(b) B_{wo}

Figure A.13: EKF trajectory estimation for A_{wo} and B_{wo} with all sensors



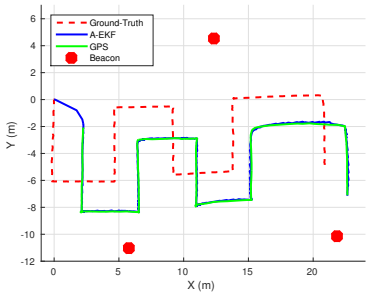
(a) B_w



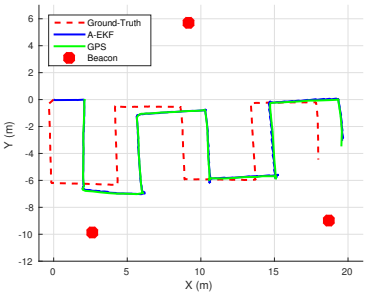
(b) C_w

Figure A.14: EKF trajectory estimation for B_w and C_w with all sensors

- Augmented Extended Kalman Filter Trajectory Estimation (Section 4.2.2)

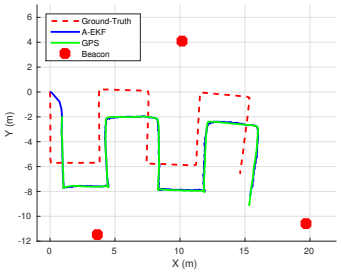


(a) A_{wo}

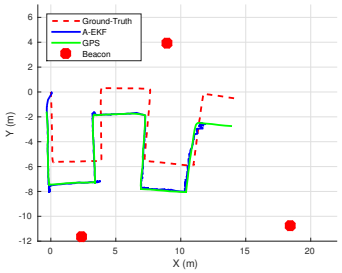


(b) B_{wo}

Figure A.15: A-EKF trajectory estimation for A_{wo} and B_{wo} with all sensors

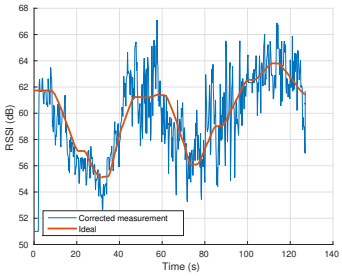


(a) B_w

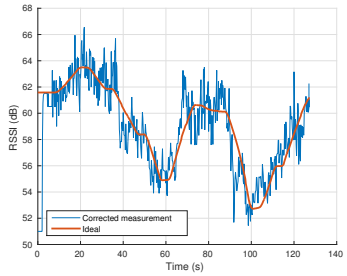


(b) C_w

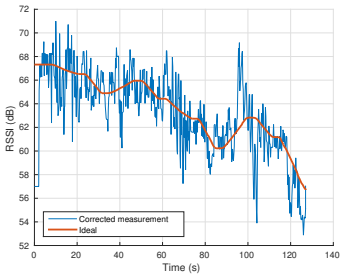
Figure A.16: A-EKF trajectory estimation for B_w and C_w with all sensors



(a) B_1



(b) B_2



(c) B_3

Figure A.17: RSSI measurements with corrected bias through the A-EKF

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br