Università degli Studi di Pisa

FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Wireless communication services to support teams of cooperating autonomous robots

Candidato

Carmelo Di Franco

Matricola 429608

Relatori

Prof. Giorgio Buttazzo

Prof.ssa Gigliola Vaglini

Prof. Luis Almeida

Anno Accademico 2012–2013

$\pi = 3.141592653...$: *The ratio of the circumference of a circle to its diameter. And this is just the beginning. It keeps on going. Forever. Without ever repeating. Which means that contained within this string of decimals is every single other number.*

Your birth date, combination to your locker, your social security number. It's all in there somewhere. And if you convert these decimals into letters, you would have every word that ever existed in every possible combination. The first syllable you spoke as a baby, the name of your latest crush, your entire life story from beginning to end. Everything we ever say or do... All of the world's infinite possibilities rest within this one simple circle. Now what you do with that information... What it's good for... Well, that would be up to you.

Acknowledgements

I would like to thank my Supervisor Giorgio Buttazzo, for believing in me and allowing me to live a wonderful experience abroad.

Professor Luis Almeida, and phd. Luis Oliveira for welcoming me kindly, for supporting me during these months, and helped me in my work.

I thank my friends and my girlfriend to be always by my side, both in good times and in the most difficult ones.

Finally, I thank my entire family, and especially my parents for always believing in me, for allowing me to study, for their love, their support, every day in my choices

Contents

Abstract

This work consists in investigating wireless communication protocols to provide services that are commonly required to support cooperation among autonomous robots. Particular attention will be dedicated to RF-based relative localization services that are infrastructure-free. The main idea is to research the joint use of RF-ranging with RSSI-based techniques to develop a system that has improved accuracy with faster response.

Beyond these services, the work will also address local state data sharing and global point-to-point communication on a volatile topology. Using a self-synchronization technique, the work will build upon previous efforts to track the topology and provide reservations (channels) on-demand, that route communications between nodes.

Chapter 1

Introduction

The use of robotic vehicles to perform tasks autonomously is becoming widespread due to both technological and scientific advances, for example, the miniaturization of electromechanical systems and new sensing and control paradigms. It is natural to imagine that soon, teams of vehicles will be fully autonomous and capable of carrying out challenging tasks. The use of autonomous vehicles requires coordination through the use of cooperation strategies because there are tasks that one vehicle alone could not perform due to both its partial knowledge about the task and limited resources.

1.1 Mobile Applications

Multiple-robot systems can accomplish tasks that cannot be achieved individually. That is why a cooperating team of mobile robots, joining together to accomplish a common objective with no human intervention, is an interesting possibility. This problem can be found in many robotic applications, either for military or civil purposes. Some examples are:

- environmental: eg. prevention of fire, tracking of toxic clouds
- military : eg. surveillance, demining areas ...
- healthcare: eg. search and rescue in catastrophic situations
- logitisics: eg. manufacturing, large volume transportation

Moreover, using multiple such units can increase the effectiveness of surveillance in a big building or outside, improve the rate of coverage in

search and rescue, reducing cost of equipment using cooperative sensing and inter-robot motion coordination. For example, When someone is lost during a snowstorm or under a snow avalanche, improving the rate of coverage in search and rescue is fundamental since the survival rate of a person under the snow drops drastically after 15 minutes. So using multiple units that autonomously navigate and start searching is really interesting. Another example is in a mine sweeping application, it is advisable to spread a team of robots with mine detecting capability and equip only a small portion of them with sweeping ability, thus reducing the cost of equipment. When mines are detected a robot with sweeping ability is informed to approach the specific spot.

For such cooperation one of the key factors is to know the positions of the robots, both absolute and relative.

1.2 What is Localization

Localization is fundamental in large number of applications. The term localization means identifying the position of an object within a reference system. Depending on the type of reference system we can have two types of localization: absolute or relative. In several applications you need to know the real position of an agent(e.g. search and rescue). In relative localization systems each robot attempts to determine the position of every other robot in the team, relative to itself.

In some situations, a possible solution is to build an infrastructure that enables every robot to know its own absolute position. But, building infrastructure is costly and it is probably unavailable in urgent scenarios. GPS may be a possible solution for outdoors but may be not available in indoor spaces and street canyons. A possible solution, which is considered in our work, is to derive relative positions from local communication using algorithms such as the Multi-Dimensional Scaling (MDS)(See Chapter. 7) which minimizes the dissimilarities of a connectivity matrix up to a rigid formation. Howerer, in order to implement such solution the robots must first collect inter-robot distance information.

One of the technologies used for obtaining distances with Radio-Frequency (RF) communication is Time-of-Flight (ToF) measurements, where one unit measures the time a message needs to reach the destination and return,thus

obtaining the distance that separates them. This method produces a distance that is accurate enough to be used for localization but is only possible to range one robot per ranging operation, thus making this method less responsive to fast robots dynamics and it is not scalable. Another possibility is RSSI based ranging that uses the signal strength of a received message to calculate the distance between the two nodes. This method produces faster measurements but it is not very accurate because RSSI is a measurement of signal strength , thus dependent of the propagation medium, antenna , and obstacles.

1.3 Proposal

This work consists in investigating wireless communication protocols to provide services that are commonly required to support cooperation among autonomous robots. Particular attention will be dedicated to RF-based relative localization services that are infrastructure-free. The main idea is to research the joint use of RF-ranging with ToF and RSSI-based techniques to develop a system that has improved accuracy with faster response.

In this work we propose to fill the gap between the RSSI and the ToF approaches. To accomplish that, we propose to use ToF ranging to estimate the log-distance path loss model. This model will increase accuracy in the transformation of RSSI measurements in distance values. The advantages to previous work are:

1. no need for any extra sensors, since all the data is captured from the transceiver module

2. no need for any a priori knowledge, the channel model is estimated online and there are no a priori localised anchor nodes

3. to support the high dynamics of RSSI with the improved precision of ToF

Beyond these services, the work will also address local state data sharing and global point-to-point communication on a volatile topology. Using a self-synchronization technique, the work will build upon previous efforts to track the topology and provide reservations (channels) on-demand, that route communications between nodes.

1.4 Structure of the dissertation

In chapter 2 we will take an overview of location systems. We will show the main methods of localization, and we will focus more on what we're going to use. Chapter 3 will briefly describe the related work that has already been done on this topic. We are showing some of the articles that were close to our work, showing strengths and problems that have not been addressed.

Chapter 4 will show the idea from which we want to start to create an algorithm that meets our specifications. We will show the main idea that underlies it (our online channel estimator model) and will outline a first version of the algorithm. In Chapter 5 we will study a series of digital filters that will help us in the correction of errors caused by measurements. Will be implemented some of them and will show the differences. Finally, the final form of the algorithm will be outlined.

In Chapter 6 and 7 we will implement the algorithm on a real environment. This implementation will allow us to be able to clearly visualize the results obtained. In Chapter 7 we will show an example of localization based on the algorithm MDS that allows you to create local maps.

Finally in Chapter 8 will discuss about the entire work and we will describe some future works and topic that can be explored.

Chapter 2

Localization Systems

2.1 Types of Localization

Location [1] systems provide a new layer of automation called automatic object location detection. Real world applications relying on such layer are many: location of products stored in a warehouse, location of medical personnel or equipment in a hospital, location of firemen in a building on fire, etc. Different applications may require different types of location information: physical/symbolic or absolute/relative.

2.1.1 Absolute Localization

In several applications you need to know the real position of a node. In these cases the localization system is composed of two fundamental components:

- *Mobile nodes*: correspond to the objects that you want to locate, their location is not known a priori and therefore, are free to move within the area where you installed the tracking system.
- *Anchor nodes*: are installed at known positions and stay there permanently. they are used as a reference point for the calculation of the absolute position. The area in which you want to install the tracking system will have to be completely covered by anchor nodes.

There are four different system topologies for positioning systems:

[1] "location", "localization", and "positioning" can be used interchangeably along the entire text.

1. *remote positioning*: the signal transmitter is mobile and several fixed measuring units receive the transmitter's signal. The results from all measuring units are collected, and the location of the transmitter is computed in a master station.

2. *self-positioning*: the unit receives the signals of several transmitters in known locations, and has the capability to compute its location based on the measured signals.

3. *indirect remote positioning*: if a wireless data link is provided in a positioning system, it is possible to send the measurement result from a self-positioning measuring unit to the remote side.

4. *indirect self-positioning*: the measurement result is sent from a remote positioning side to a mobile unit via a wireless data link.

2.1.2 Relative Localization

In relative localization systems each robot attempts to determine the position of every other robot in the team, relative to itself. For many team-oriented behaviours, it is this latter kind of localization that is most important. Consider, for example, a team of robots executing a formation behaviour: these robots need not know their latitude and longitude, but must know the relative position of their neighbours. Naturally, given a set of absolute position estimates for the robots, one can always derive relative positions. Sometimes building infrastructure is not feasible because is costly and it is probably unavailable in urgent scenarios and you can only measure relative distances between robots.

2.1.3 Range-Based and Range-Free Methods

There are two methods to locate a node: Range-Based and Range-Free Methods. In the Range-Based methods location discovery consists of two phases: Ranging Phase and Estimation Phase:

- *Ranging Phase*: where each node estimates its distance or angle from its neighbours

- *Estimation phase*: where nodes use ranging information and beacon

node locations to estimate their positions(we will focus on the Estimation phase in the last part of this work).

Range-free solutions estimate the location of sensor nodes by, either, exploiting the radio connectivity information among neighbouring nodes, or exploiting the sensing capabilities that each sensor node possesses. You need a large number of anchors to estimate the sensor location. The advantages are that you can use cheap sensor hardware and you can have low computational power, but the disadvantage is that you have less accuracy than range based methods.

2.2 Measurable quantities for localization

There are several methods for estimating the distance between two robots. These methods use various types of measurement. In general, measurements involve the transmission and reception of signals between hardware components of the system. You can measure the following physical quantities:

- time - Phis. q. second
- Intensity - Phis. q. decibel
- Phase(Interferometry) - Phis. q. meter
- Angle of Arrival - Phis. q. Radiant

2.3 Systems based on time

These systems can measure the time it takes for a signal to propagate from a transmitting station to a receiving station. Radio waves propagate in the air with a velocity slightly less than that of light ($c = 299792468m/s$) and given that the light takes $3.3ns$ to travel a meter, in order to obtain an accuracy of 30 cm should have a system able to detect time intervals up to a nanosecond. Such precision is available on Ultra Wide Band (UWB) devices. The UWB systems are the most precise but also more expensive. Instead in other systems, to make the measurements, it was decided to use signals slower compared to radio waves: ultrasound. Although Cheaper than UWB systems, these systems present problems over long distances just for the fact

of using the ultrasound and this results in the need to install more anchor nodes.

2.3.1 Time of arrival

With Time-of-arrival (ToA),The distance from the mobile target to the measuring unit is directly proportional to the propagation time. The one-way propagation time is measured, and the distance between measuring unit and signal transmitter is calculated. This method gives better results with long distances because we work at the speed of the light.

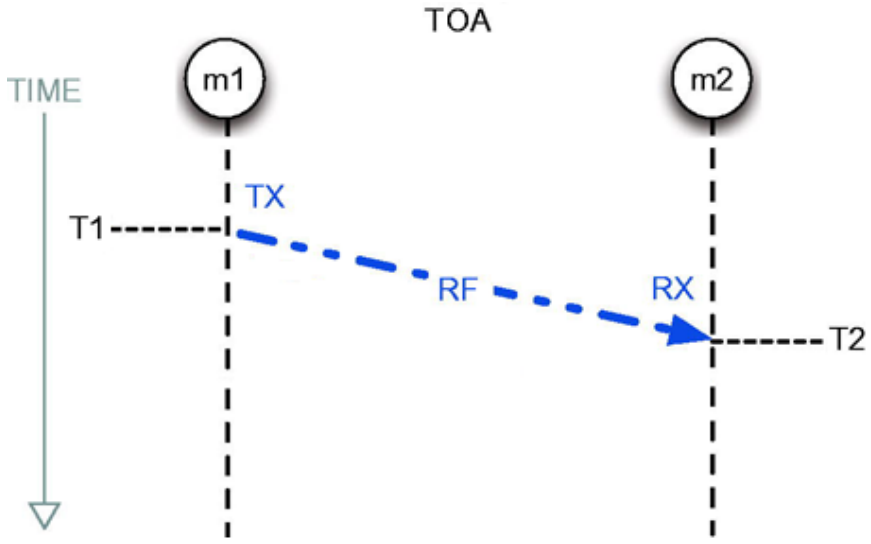


Figure 2.1: Diagram showing the the ToA process

However it requires that all transmitters and receivers in the system are precisely synchronized. The receiving unit will measure the time with his internal clock that must be perfectly synchronized with the internal clock of the transmitting unit. For example if you have a synchronization error of $100ns$ you will have a distance uncertainty that is 30 meters.

2.3.2 Time difference of arrival

Time-Difference-of-Arrival (TDoA) is a technique for measuring the propagation time of a signal when you have available devices capable of transmitting signals with different speeds of propagation. A node is equipped with sensors that can transmit both radio waves (RF) and ultrasound (U.S.). It trans-

mits a radio signal and immediately after an ultrasonic signal. Since sound travels much more slowly than radio waves, nodes will receive first the RF signal and only after a certain time will also get the U.S. signal.

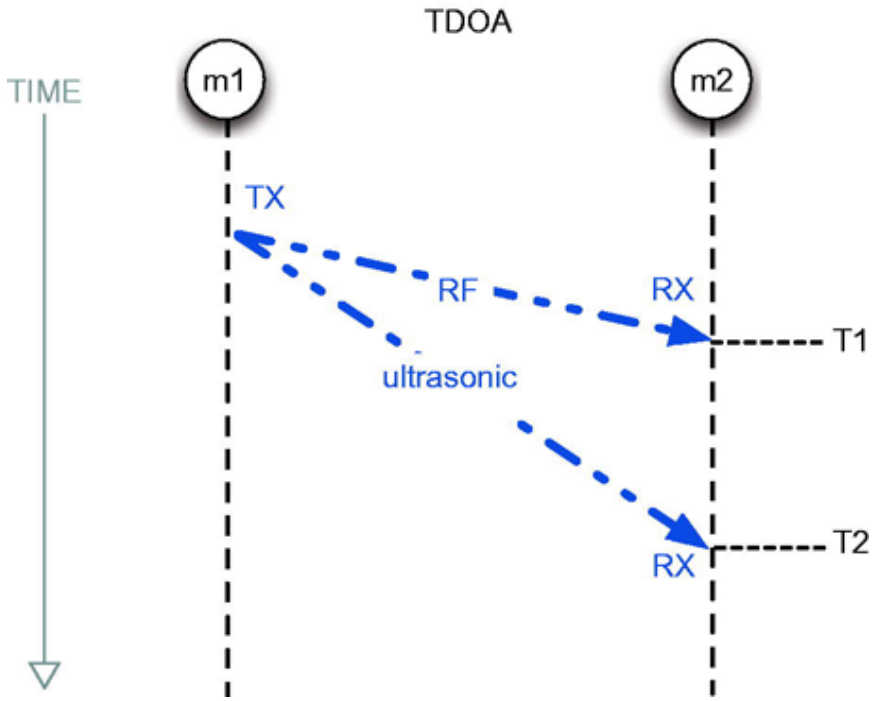


Figure 2.2: Diagram showing the TDoA process

In 2.2 you can see a TDoA example. At instant t_0 node m_1 sends the two signals. At t_1 the node m_2 receives the RF signal and at t_2 the U.S. signal. Whereas the RF signal propagation is much smaller than U.S. propagation ($t_1 - t_0 << t_2 - t_1$), we can measure the U.S. propagation time simply like a difference between t_2 and t_1. The distance will be

$$distance = (t_2 - t_1) \times v \tag{2.1}$$

where v is the velocity of the ultrasound signal. Similar to ToA or any other time-based methods, synchronization must exist in order for different time measurements to be accurate but since TDoA does not use the distance between the transmitter and the receiver, the transmitter is not required to be in sync with the sensor. However, it is expensive because it takes additional hardware. In 2.3 the figure an example of a device with an ultrasonic sensor together with the RF interface.

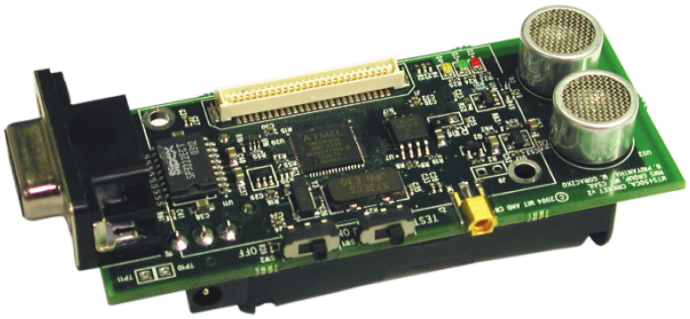


Figure 2.3: Device Cricket MCS410CA. You can see on the right the ultrasound transmitter and receiver.

2.3.3 Time of Flight

ToF ranging calculates the distance between units by measuring the time the signal needs to reach the receiver and come back at the transmitter. Since we will work with the NanoLOC Devices (See Section A.3) we will describe Time of flight directly from NanoLOC Data-sheets.

Ranging in the nanoLOC chip uses two types of transmissions, which are Data packet and hardware Acknowledgements, to obtain two types of time measurements:

- TX Propagation Delay: This delay is the time for a data or acknowledgement packet to be transmitted from one station to another. As the speed of a signal propagating through the air is known (the speed of light), the time in which a packet is sent from one station to another can be used to calculate the distance between the stations.

- Processing Delay : This delay is the time required to process a received data packet and generate and transmit a hardware acknowledgement packet to the sending station. This also is a known value and is used as part of the ranging calculations. These time measurements are accumulated and with a ranging formula used to obtain a ranging distance between two nanoLOC nodes.

The chip also offers two ranging modes: Normal Ranging Mode and Fast Ranging Mode. These are briefly discussed below

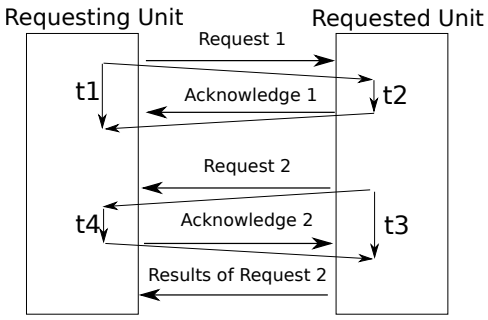


Figure 2.4: ToF - Ilustration of the ranging process

Normal Ranging Mode

Normal ranging mode uses a symmetrical ranging methodology that means that the measurement from the transmitter to the receiver is mirrored by a measurement from the receiver to the transmitter (ABA to BAB) (Fig. 2.4).

Ranging measurements between two stations in normal ranging mode are obtained using the following formula:

$$distance = \frac{(t_1 - t_2) + (t_3 - t_4)}{4} \tag{2.2}$$

where:

- t_1 is the propagation delay time of a round trip between the transmitter and the receiver
- t_2 is the processing delay in the receiver
- t_3 is the propagation delay time of a round trip between a receiver and a transmitter
- t_4 is the processing delay in the transmitter

Fast Ranging Mode

Fast ranging mode uses the same ranging methodology as normal ranging mode, except that it is not symmetrical. Only one set of measurements are used (ABA). This increases the speed at which ranging values can be determined, but without the additional validity of the second measurement in normal ranging mode.

Ranging measurements between two stations in fast ranging mode are obtained using the following formula:

$$distance = \frac{t_1 - t_2}{2} \tag{2.3}$$

2.4 Systems based on Signal Strength

2.4.1 Received Signal Strength Indicator (RSSI)

Evaluating the intensity of a received signal it is possible to estimate the distance of the station that has transmitted. In the case of sensor networks and WiFi networks we can use RSSI that indicates an estimate of the received signal power in dBm. To correlate the power of a signal received with the distance from which it was sent, we can use the Friis equation:

$$P_R = P_T * \frac{G_T G_R \lambda^2}{(4\pi)^2 d^\alpha} \tag{2.4}$$

where:

- P_R: is understood to be the available power at the receive antenna terminals in Watt
- P_T: is understood to be the power delivered to the transmit antenna
- G_T: is the gain of the transmitting antenna
- G_R: is the gain of the receiving antenna
- λ: is the wavelength, $\lambda = c/f$ where c is the speed of the light and f is the frequency
- d: is the distance in meters
- α is the path loss exponent end is environment-dependent.

Since RSSI is usually in dBm, we have to convert it with the following equation:

$$P[dBm] = 10 \log(P[W] \times 10^3) \tag{2.5}$$

assuming G_T and G_R equals to 1, and considering a maximum output power P_T equals to 1 mW, $lambda = c/f = 0,12277$ (freq: 2441,75 MHz)

Table 2.1: Some α values in specific environments

Environment	α
Free Space	2.0
Retail store	2.2
Grocery store	1.8
Office, hard partitions	3.0
Office, soft partitions	2.6
Metalworking factory,line of sight	1.6
Metalworking factory, obstructed line of sight	3.3

$$\begin{aligned} P_R[dBm] = 10\log(P[W]\times 10^3) &= 10\log\left(P_T * \frac{G_T G_R \lambda^2}{(4\pi)^2 d^\alpha}\times 10^3\right) \\ &= 10\log\left(P_T * \frac{9.5459 * 1 * 1}{d^\alpha}\times 10^3\right) = \\ &= 10\log\left(9.5459 * P_T * 10^3\right) - 10\log\left(d^\alpha\right) = \\ &= \rho_0 - 10\alpha\log\left(d\right) = \end{aligned} \tag{2.6}$$

where ρ_0 is the transmitted power, in dBm. From 2.6 we can derive the distance from the transmitting node that sent the message. Unfortunately RSSI value is not very reliable, since it depends on the environment and on the reflected waves; hence the computed distance is not very accurate. In table 2.1 you can see how the path loss exponent changes in different environments.

2.5 Systems based on Phase

2.5.1 Interferometry

If a node has two antennas separated by a distance d, on which the radio waves arrive with an incident angle θ, we can see that the signal must travel different distances to reach both antennas. This difference results in a phase difference $\Delta\Theta$ of the received signal between the two antennas. With this formula we can calculate the direction of arrival of the signal:

$$\theta = \sin^{-}1\left(\frac{\lambda\Delta\Theta}{2\pi d}\right) \tag{2.7}$$

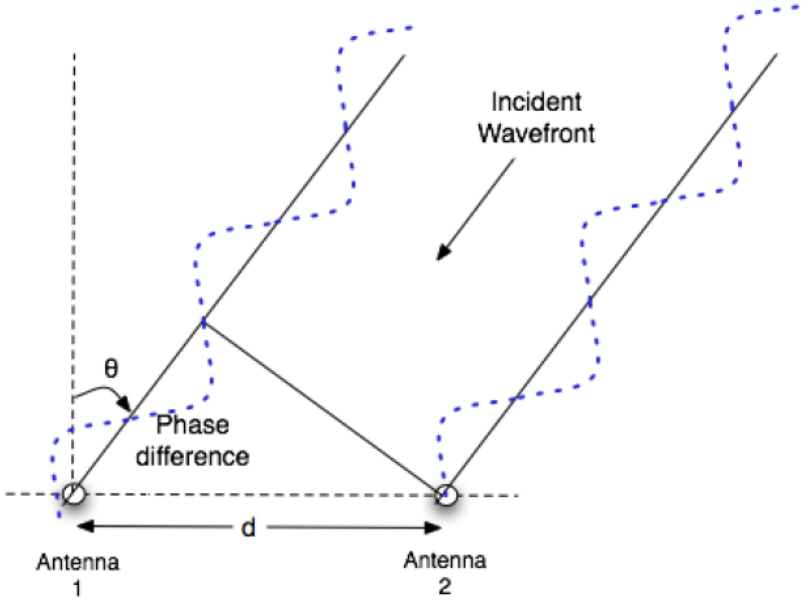


Figure 2.5: Principle of interferometry fig:interferometria

where λ is the wavelength of the signal. This principle is based on some tracking systems, such as Radio Interferometric Positioning System (RIPS): they provide good accuracy but problems can arise if the signal reaches its destination from more paths through various reflections. Indeed in the case in which the signal is received through multiple paths (multipath) it can be difficult to understand what is the right phase difference to use in calculations.

2.6 Systems based on Angles

2.6.1 Angle of arrival

With Angle of arrival (AoA) method, location is derived from the intersection of several pairs of angle direction lines. It rquires estimating relative angles between neighbours. It uses directional antennas or array of antennas and no time synchronization is needed. The disadvantage is that it requires additional hardware and is expensive to deploy in large sensor networks.

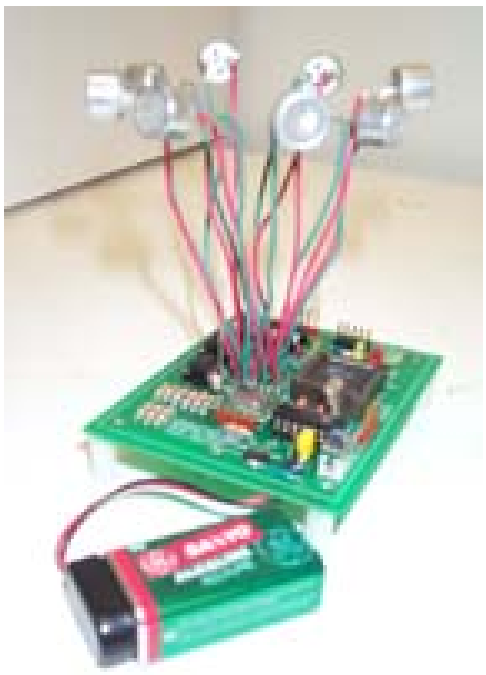


Figure 2.6: First generation Medusa node prototype

Chapter 3

Measuring Distances

Measuring distance between wireless nodes is a topic that has been widely explored by many authors. Some focus on time-based techniques, some focus on signal strength techniques, and others on hybrid approaches.

3.1 Time-based techniques

The most common time based techniques rely on one-way Time-of-arrival (ToA) measurements, Time-Difference-of-Arrival (TDoA) andTime-of-Flight (ToF) measurements [1]. However, ToA and TDoA require global time synchronisation, since the measurement is unilateral.

On the other hand, ToF eliminates the need for global clock synchronisation. In order to do that, instead of measuring the time of one-way trip, it measures the time that a message needs to go to the receiver and return to the transmitter. Since some local processing needs to be done on the receiver before sending the reply, the processing time has to be very well known, thus it is usually done in hardware. Adding to that, since the ranging operation is between two units, it needs a long time to range several units, thus **it may not accommodate fast moving robots**.

3.2 Signal Strength techniques

The signal strength based techniques, as the name implies, obtain range estimations from the strength of the received RF signal (see section 2.4) . In open space and without interference there is a predictable relationship

between RSSI and distance, however, in the presence of interference, reflection, and refraction, this relationship is no longer accurate. Despite that, most of the current wireless transceivers possess the capability of measuring the RSSI intrinsically. Therefore, if the application only requires a coarse localization, either for navigation or topology estimation purposes, the RSSI can still be very useful. In order to obtain ranging data from RSSI, some researchers use anchor-free RF only localization methods without previous knowledge, such as in [2],[3] , where RF-based localization is performed. In [2],[3] , the authors do not consider a propagation model and all localization is performed considering the "distance in the RSSI space", i.e., not an estimate of relative physical distance.

Other researchers rely on channel models to estimate real distance based on RSSI, some using a priori channel measurements [4], and others performing online channel estimation, either based on anchor nodes [5] or based on external sensors. However, a priori data may be unavailable or unreliable, i.e. either there is no previous knowledge or there were severe changes to the environment; estimations based on measurements between anchor nodes are not compatible with unknown environments; and estimations performed with external sensors require extra equipment.

3.3 Hybrid approaches

In this section present two works that make use of hybrid approaches:

1. Comparison of hybrid localization schemes using rssi,toa,tdoa [6]
2. A data fusion technique for wireless ranging performance improvement [7]

The first work presents a simulation study of non-hybrid and hybrid localization techniques using RSSI, ToA, and TDoA location dependent parameters. The assumed scenario here is a situation where the targeted mobile is connected to different anchors from which it is able to get different parameters. In Fig. 3.1 we can see an example of a generic heterogeneous scenario. Simulations(Fig. 3.2) have revealed that when ToAs and/or TDoAs have high accuracy, the use of RSSIs is either marginal or not necessary. Nevertheless, RSSIs are very important and may enhance positioning accuracy

in cases where no sufficient number of ToAs or/and TDoAs is available or when their precisions are not accurate.

In the second paper , a hybrid approach fusing RSSI and rToF measurements is used. Here the authors proposed a data fusion algorithm to combine both techniques assuming the channel parameters to be estimated in advance.

One of the most limiting factors in the first work is that the authors use anchors to improve the localization. Moreover, Time-based techniques require a long time to range one robot, and RSSI allows several receivers to "range" one transmitter simultaneously, thus making RSSI appealing for applications with mobile robots where the dynamics of the movements are not negligible. Unlike the second work, which assumes the channel parameters to be estimated in advance, our approach assumes no prior knowledge and estimates the channel parameters in real time.

In our work we explore using the higher accuracy of ToF measurements to improve the accuracy of a faster RSSI-based distance estimator by recurrent online re-calibration.

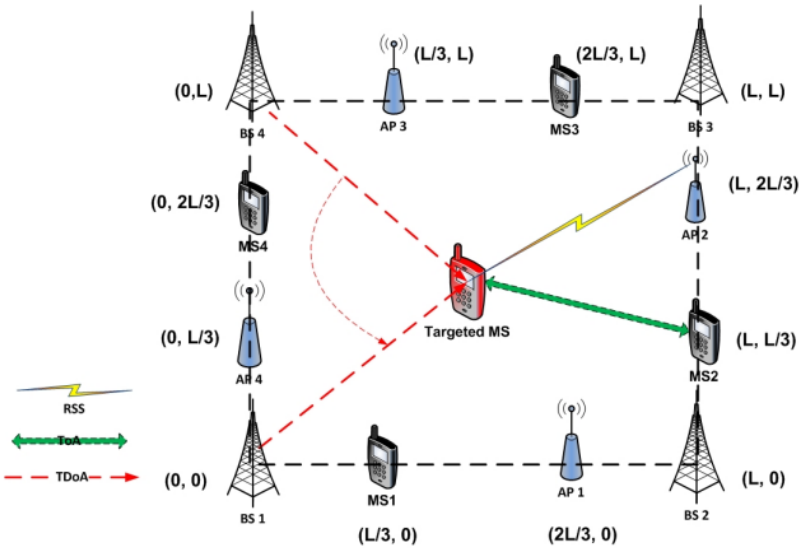


Figure 3.1: Heterogeneus generic Scenario

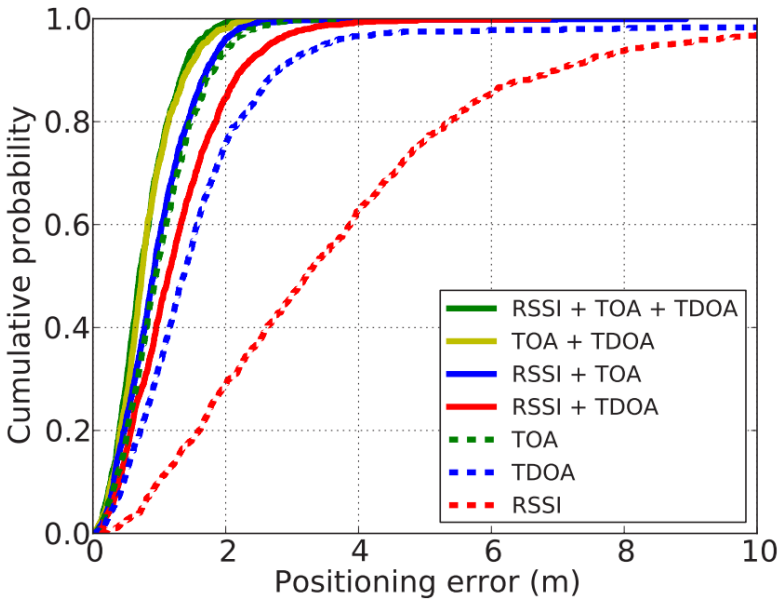


Figure 3.2: CDFs of positioning error for different non-hybrid and hybrid schemes using ML technique

Chapter 4

The Adaptive Radio-Frequency Ranging Algorithm

As written in the Introduction, one of the main issues in the RSSI method is the high dependency with the environment.

To get distance from RSSI we need to know some parameters from equation (4.11), namely the reference RSSI value ($RSSI_0$) at the respective reference distance(d_0) and the path loss exponent (α).

In order to calibrate the Channel model, we need to estimate $RSSI_0$ and α. To do this we need to collect $RSSI_d - distance$ pairs and for the Estimation we can use the Maximum-Likelihood Estimator (MLE) or other similar methods. After getting the parameters of the channel model we can consider RSSI accurate enough for a certain interval of time. The length of this interval can be for example a given time, or the interval after the robot has moved a certain distance(Fig. 4.1). Thus we will have a periodic online Channel Model estimation that will give us an enhacement of the RSSI method performance.

The main steps to develop our approach are:

- Preliminary estimation and evaluation of the channel model
- Design and refining the Algorithm for the Adaptive Radio-Frequency Ranging
- Evaluating the Algorithm with experiments

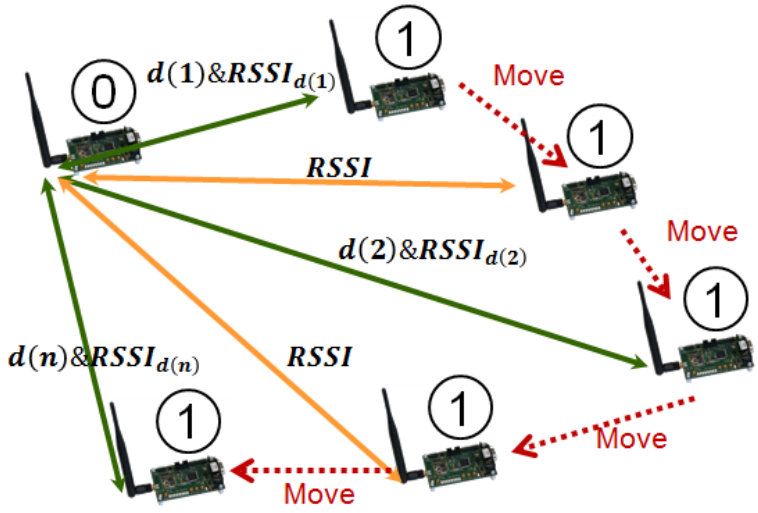


Figure 4.1: Fusing ToF and RSSI for adaptive Radio-Frequency Ranging

4.1 Least Squares Estimator (LSE): Theory

The method of Least Squares is a standard approach to the approximate solution of overdetermined systems, i.e., sets of equations in which there are more equations than unknowns. "Least squares" means that the overall solution minimizes the sum of the squares of the errors made in the results of every single equation. The method of least squares assumes that the best-fit curve of a given type is the curve that has the minimal sum of the deviations squared (least square error) from a given set of data. The least squares criterion has important statistical interpretations. If appropriate probabilistic assumptions about underlying error distributions are made, least squares produces what is known as the maximum-likelihood estimate of the parameters. Even if the probabilistic assumptions are not satisfied, years of experience have shown that least squares produces useful results. The computational techniques for linear least squares

4.1.1 Problem Statement

Suppose that the data points are $(x1, y1), (x2, y2), ..., (xn, yn)$ where x is the independent variable and y is the dependent variable. The fitting curve $f(x)$ has the deviation (error) d from each data point, i.e., $d_1 = y_1 - f(x_1), d_2 = y_2 - f(x_2), ..., d_n = y_n - f(x_n)$. According to the method of least squares, the best fitting curve has the property that:

$$S = d_1^2 + d_2^2 + ... + d_n^2 = \sum_{i=1}^{N} d_i^2 = \sum_{i=1}^{N} [y_i - f(x_i)]^2 = is\ minimized. \quad (4.1)$$

hence the name "least squares". In practical cases generally $f(x)$ is parametric: in this way the problem is reduced to determining the parameters that minimize the distance of the points from the curve. Of course, to obtain a single optimized curve and not a bundle, you need a number of experimental points greater than the number of parameters which determine the curve (the problem usually known as overdetermined). In general the experimental data obtained would exhibit a distribution governed by certain analytical relationships. Then it is useful to parametrize the theoretical curve and determine the parameters so as to minimize S.

4.1.2 Solution of the linear case

Let $f(x)$ be a linear function of the parameters:

$$f(x) = p_1 f_1(x) + p_2 f_2(x) + \cdots + p_k f_k(x) \quad (4.2)$$

where p_i are k parameters with $k \ll n$, and n is the number of known points. You can rewrite $f(x)$ through the linear system oversized

$$Ap \approx y \quad (4.3)$$

where

$$A = \begin{bmatrix} f_1(x_1) & \dots & f_k(x_1) \\ \vdots & \vdots & \\ f_1(x_n) & \dots & f_k(x_n) \end{bmatrix}, p = \begin{bmatrix} p_1 \\ \vdots \\ p_k \end{bmatrix}, y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad (4.4)$$

The problem of minimizing S leads back therefore to minimizing the norm of the residue

$$\|r\| = \|Ap - y\|, \|r\|^2 = \|Ap - y\|^2 = ([Ap]_1 - y_1)^2 + \cdots + ([Ap]_n - y_n)^2 \quad (4.5)$$

$$= \sum_{i=1}^{n} (f(x_i) - y_i)^2 = S \quad (4.6)$$

where $[Ap]_i$ is the i-th component of the resulting vector $y = Ap$.

We can minimize $\|r\|$ deriving $\|r\|^2$ in respect of each p_m and setting the derivative equal to 0:

$$\frac{d\|r\|^2}{dp_m} = \sum_{i=1}^{n} 2(\sum_{j=1}^{k} a_{ij}p_j - y_i)a_{im} = 0 \tag{4.7}$$

This is equivalent to the following system:

$$(Ap - y)^T A = 0 \tag{4.8}$$

Hence the vector p that minimizes S is the solution of the equation:

$$A^T Ap = A^T y \tag{4.9}$$

This latter equation is called normal equation. If the rank of A is complete then $A^T A$ is invertible and therefore:

$$p = (A^T A)^{-1} A^T y \tag{4.10}$$

where the matrix $(A^T A)^{-1} A^T$ is called pseudo-inverse.

4.2 Channel model Estimation with LSE

In 4.1 we talked about a method, LSE, that is a standard approach to the approximate solution of overdetermined systems. For a good estimate we need to collect a large number of RSSI-distance pairs separated as much as possible; for that purpose we can either rely on the movement of the robots, or force them to move in order to collect more diversified data. Let's consider the Channel Model Equation:

$$RSSI_d = RSSI_0 - 10\alpha \log\left(\frac{d}{d_0}\right) \leftrightarrow d = d_0 \times 10^{(RSSI_0 - RSSI_d)/(10\alpha)} \tag{4.11}$$

Referring to (4.4) we want to estimate $RSSI_0$ and α by knowing n known RSSI-distance pairs ($RSSI_1$,d_1),($RSSI_2$,d_2),..., ($RSSI_n$,d_n). If we assume

$d_0 = 1$ and define A and b as:

$$A = \begin{bmatrix} 1 & -10\log d_1 \\ \vdots & \vdots \\ 1 & -10\log d_1 \end{bmatrix}, b = \begin{bmatrix} RSSI_1 \\ \vdots \\ RSSI_N \end{bmatrix} \tag{4.12}$$

We can solve the equation system in the equation (4.13) and obtain the estimated parameters $\widehat{RSSI}_0$ and $\widehat{\alpha}$.

$$\widehat{X} = \begin{bmatrix} \widehat{RSSI}_0 \\ \widehat{\alpha} \end{bmatrix} = (A^T A)^{-1} A^T b \tag{4.13}$$

4.3 Preliminary estimation and evaluation of the channel model

To evaluate the channel-model estimation method we did two experiments, one in an open space and one in an indoor environment. We programmed two nanoLOC devices, naming them as *nodeA* and *nodeB*.We connected *nodeA* via UART to a PC. Then we put *nodeB* at the distance of one meter. Every step for 10 steps:

1. STOPPED: *nodeB* is stopped
2. RF-RANGING: *nodeA* collects one hundred values of RSSI-distance pairs repeating the RF-Ranging operation.
3. MOVING: *nodeB* moves away a meter from *nodeA* and sends messages to it
4. RSSI: *nodeA* collects one hundred RSSI values
5. return to step 1

Note that the values collected in the two operations(RF-RANGING and RSSI) include failures and need to been filtered. Then we used the RSSI-distance pairs to solve the equation (4.13).

From the two experiments we obtained the following data-sets:

$$Dataset1 = \begin{bmatrix} Timestamp & RealDistance & MeasuredDistance(RF) & RSSI \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{4.14}$$

and

$$Dataset2 = \begin{bmatrix} Timestamp & RSSI \\ \vdots & \vdots \end{bmatrix} \tag{4.15}$$

$Dataset1$ includes all the measurements during the RF-Ranging Operation while the $nodeB$ is always stopped at a precise position. $Dataset2$ includes all the measurements during the $nodeB$ movement while remove broadcasting a packet. $NodeA$ will get only the RSSI measurement.

4.3.1 Outdoor Experiment: results

This Experiment was made in an open space, in the university garden. With this experiment we want to see how the channel estimation works in the best scenario. We used all the measurements from $dataset1$ to estimate the channel model ($RSSI_0$,α). In figure 4.2 you can see two curves. The red curve is the estimated model with the coloured dots that are the $MeasuredDistance - RSSI$ pairs. To better understand the figure, the points are plotted with different colors for every meter.

The black curve is the real model. To estimate it, we used the black dots that are $RealDistance - RSSI$ pairs. Note that in an open space the measurements are really close to the real model. This leads to have two curves that are very similar and we have a very good approximation.

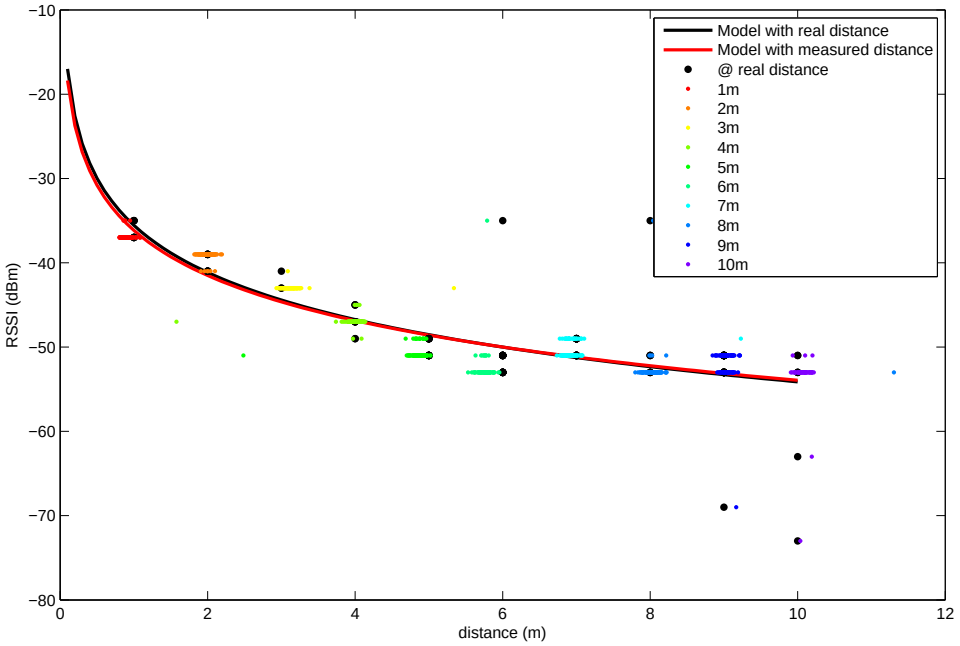


Figure 4.2: Collected data outside: Different colours for each meter $(\bar{d}, R\bar{S}SI)$; black points represent real distance$(d, R\bar{S}SI)$; lines represent models using MLE with all points

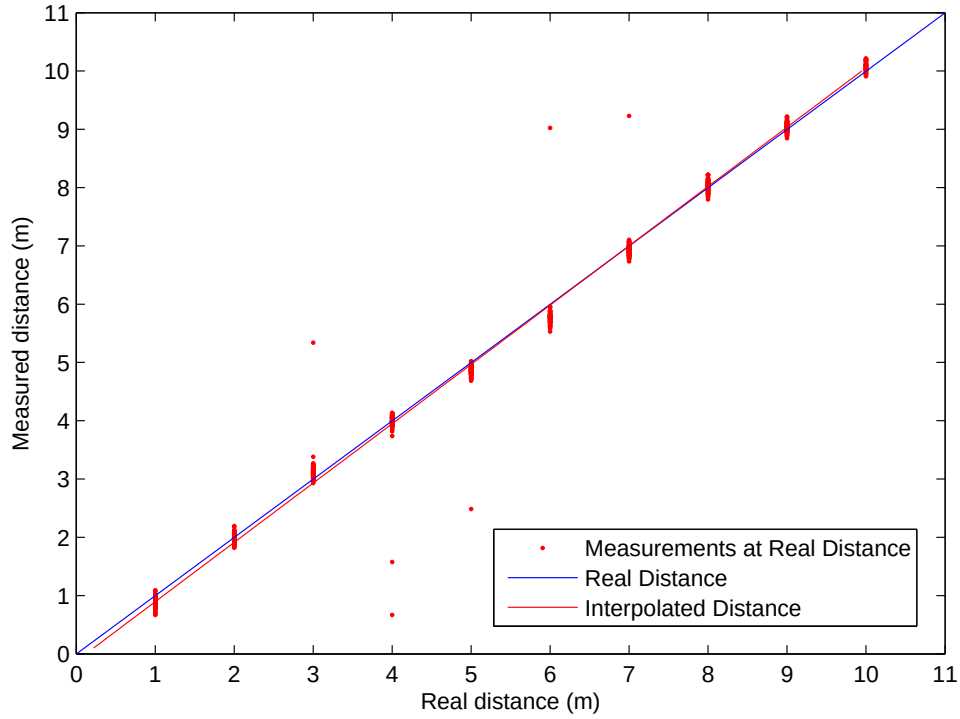


Figure 4.3: Collected data outside: measured distances with ToF ranging.

In figure 4.3 we can see a comparison between measured distances and real distance. The blue line is the bisecting line where every measured distance is equal to the real distance. The red line is an interpolated line that show how good are the measured values. In this outdoor experiment the slope of the line is 0.9825 and the offset is 0.1209.

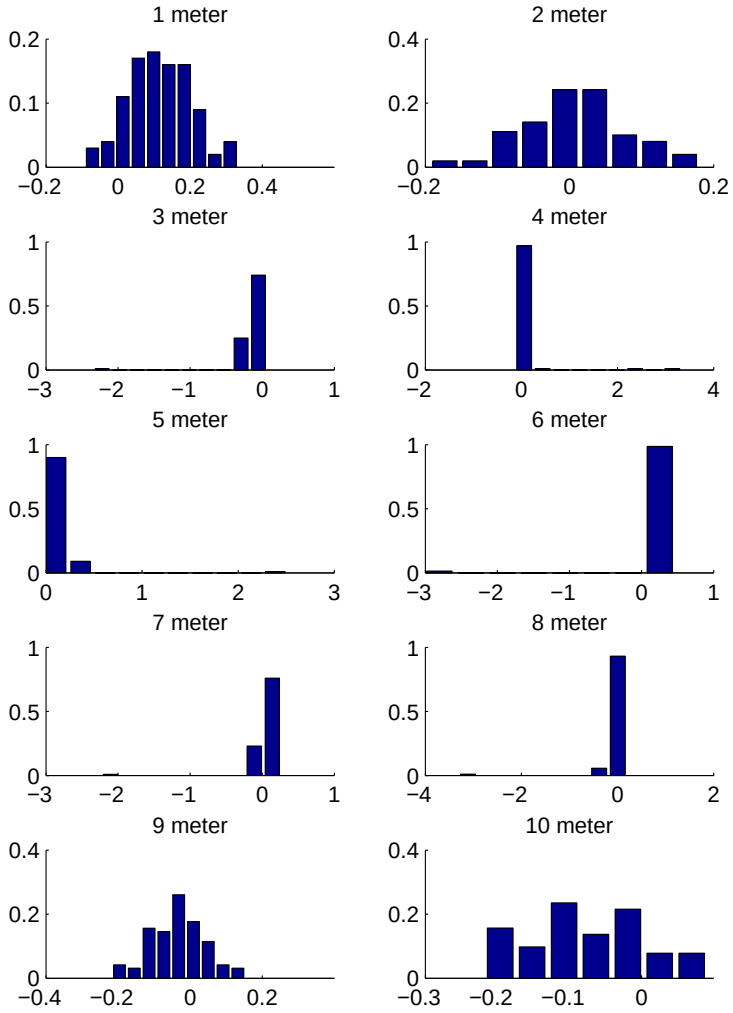


Figure 4.4: Collected data outside: histogram of the distance error for each meter.

In figure 4.4 we can see the histograms of the collected data at every

meter. The measured data has a Gaussian noise. In figure 4.5 we can see how the bias and the Standard Deviation (Std.) change at every meter.

Finally we can say that, in an outside environment the measurements present a Gaussian Noise with mean 0.0278 and Std. 0.2212

$$\text{Measurements Noise : } W \sim \mathcal{N}(\mu, \sigma^2)$$

$$\mu_d = 0.0278 \; and \; \sigma_d = 0.2212 \tag{4.16}$$

$$\mu_\rho = 0.0417 \; and \; \sigma_\rho = 1.9043 \tag{4.17}$$

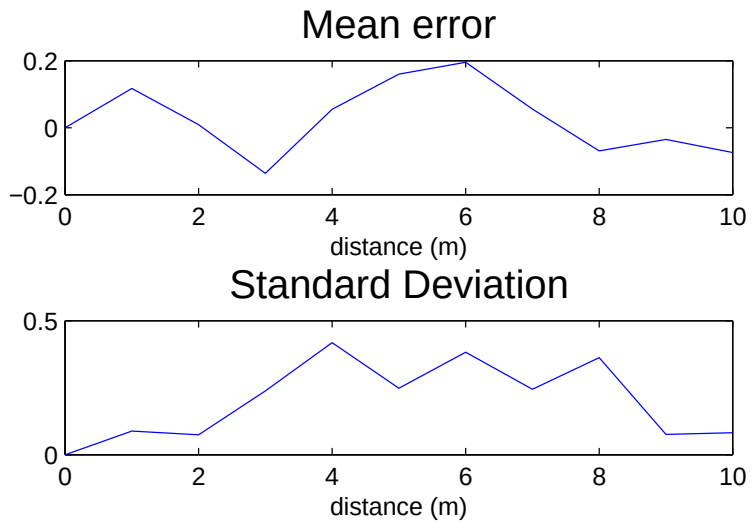


Figure 4.5: Collected data outside: Mean error and Standard deviation of the Measured distance.

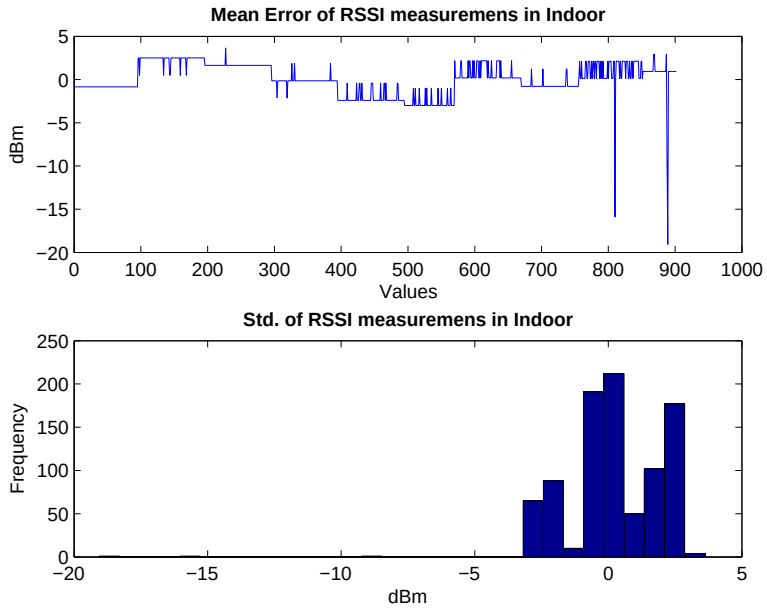


Figure 4.6: Collected data outside: Mean error and Standard deviation of the Measured RSSI.

4.3.2 Indoor Experiment

In the second experiment we repeated the first experiment but in an indoor environment(a corridor in the building).

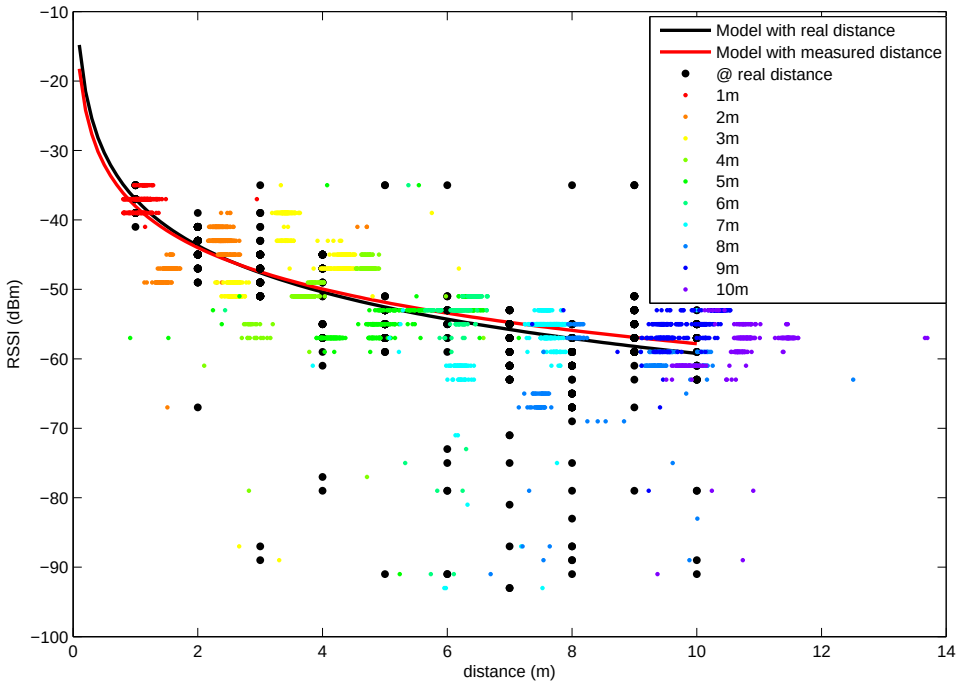


Figure 4.7: Collected data indoor: Different colours for each meter $(\bar{d}, R\bar{S}SI)$; black points represent real distance$(d, R\bar{S}SI)$; lines represent models using MLE with all points

In figure 4.7 we can see that now the measured values are more spread and there are a lot of outliers. The measured model and the real model are different.

$Rssi0$ is -38.0277 and α is 1.9789

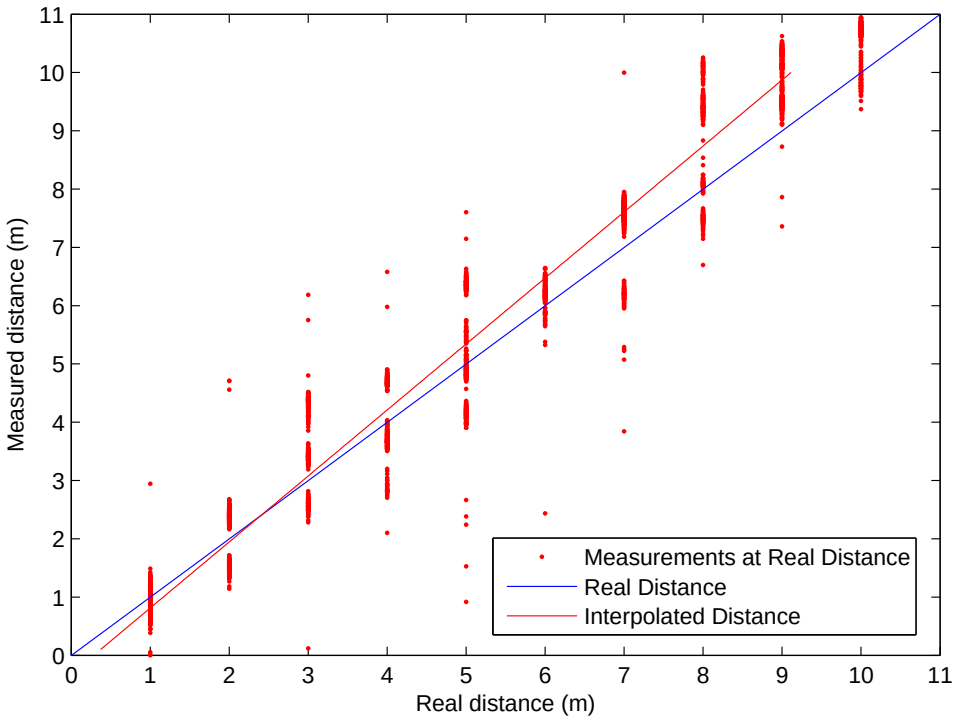


Figure 4.8: Collected data indoor: measured distances with ToF ranging.

In figure 4.3 we can see that now some measured values are far from the real distance and the slope of the red line is 0.8831 with an offset of 0.2824. So an indoor environment we have underestimated distance.

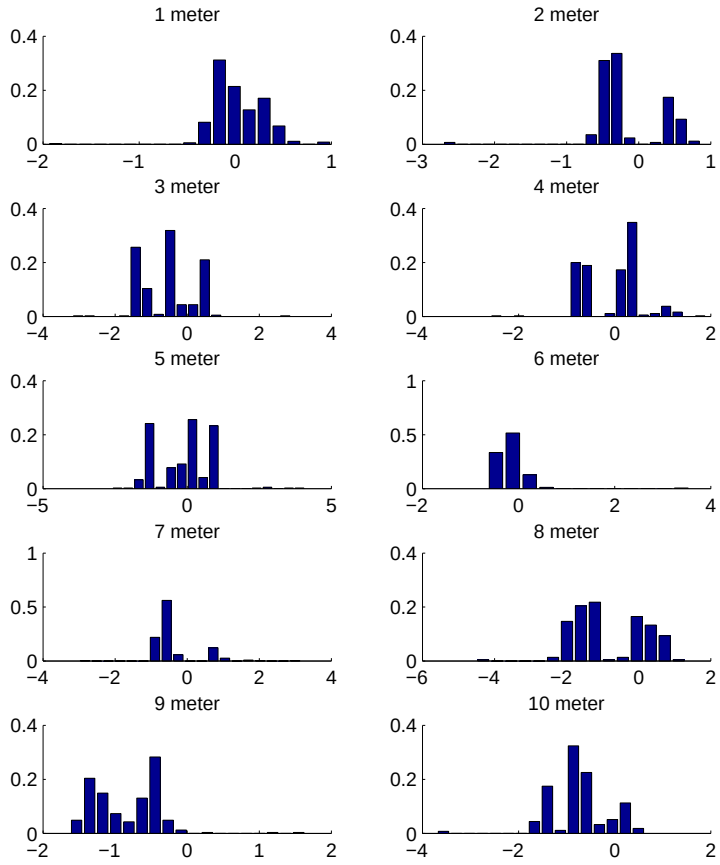


Figure 4.9: Collected data indoor: histogram of the distance error for each meter.

As in the outdoor experiment, we can see in figure 4.9 the histograms at every meter.

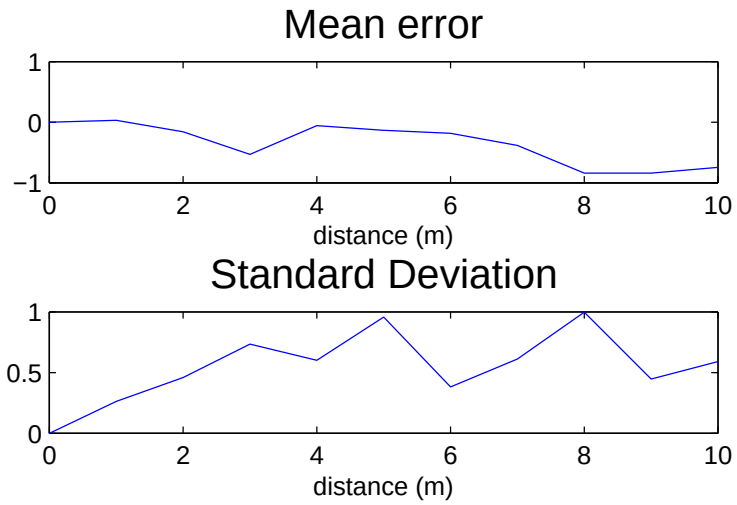


Figure 4.10: Collected data outside: Mean error and Standard deviation of the Measured distance.

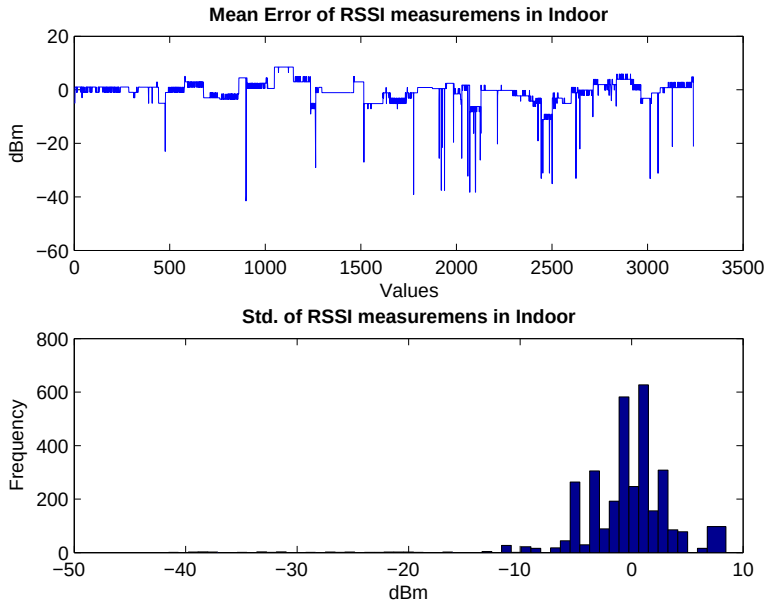


Figure 4.11: Collected data outside: Mean error and Standard deviation of the Measured RSSI.

Finally we can say that, in our indoor environment the measurements have an noise that can still be approximated with a Gaussian Noise with mean -0.3842 and Std. 0.6039

$$\text{Measurements Noise} : W \sim \mathcal{N}(\mu, \sigma^2)$$

$$\mu_d = -0.3842 \quad \sigma_d = 0.6039 \tag{4.18}$$

$$\mu_\rho = -0.6086 \quad \sigma_\rho = 4.4322 \tag{4.19}$$

4.4 Fixed-points MLE Model Estimator

The outdoor/indoor experiments gave us a clear idea about how good is the measured model and the measurements. In the previous sections we saw that the Mean error of the ToF measurements is relatively low and this gave to us a Measured Channel model close to the Real Channel model(calculated with real distances). However, our proposal is to create an algorithm that automatically update the measured model when we have new ToF measurements.

We want to use an algorithm like the Maximum-Likelihood Estimator (MLE), but using only 1 new value instead of n. For that purpose, we define a vector of predefined n log-separated distances ($g_{1\times n}$) and create the matrices $A_{(n+1)\times 2}$ and $b_{(n+1)\times 1}$ (considering $d_0 = 1$) (4.12). The first n lines represent the previously estimated model $\widehat{x_{t-1}}$, and the $n+1$ point represents the new measurement $\widehat{d}_t, \widehat{RSSI}_t$. Then we run a Maximum-Likelihood Estimator (MLE) (4.13) to obtain the new channel model $\widehat{x}_t$. This allows us to run MLE using a fixed number of samples (n + 1), and at the same time to fuse the new knowledge with previous knowledge, where n defines the weight of the new measurement.

$$A_t = \begin{bmatrix} 1 & -10\log(g(1)) \\ 1 & -10\log(g(2)) \\ \vdots & \vdots \\ 1 & -10\log(g(n)) \\ 1 & -10\log\widehat{d}_t \end{bmatrix}, b_t = \begin{bmatrix} RSSI_{0,t-1} - 10\alpha_{t-1}\log(g(1)) \\ RSSI_{0,t-1} - 10\alpha_{t-1}\log(g(2)) \\ \vdots \\ RSSI_{0,t-1} - 10\alpha_{t-1}\log(g(n)) \\ \widehat{RSSI}_t \end{bmatrix} \tag{4.20}$$

$$\widehat{X_t} = \begin{bmatrix} \widehat{RSSI}_{0,t} \\ \widehat{\alpha}_t \end{bmatrix} = (A^T A)^{-1} A^T b \tag{4.21}$$

t	$RSSI_0$	α	$\widehat{RSSI}$	$\widehat{d}$
0	-30	3	-	-
1	-30.3532	2.9707	-43	2.2701
2	-30.7338	3.0510	-57	4.0232
3	-30.8802	2.9855	-55	7.3878
4	-31.6590	2.7187	-53	9.7562

Table 4.1: first 4 steps of the fixed-MLE

4.4.1 An example

To clarify our model estimator we wrote a basic example: At $t = 0$ we initialize our model with:

$$\widehat{X_{t|t=0}} = \begin{bmatrix} \widehat{RSSI_{0,t=0}} \\ \widehat{\alpha_{t=0}} \end{bmatrix} = \begin{bmatrix} -30 \\ 3 \end{bmatrix} \tag{4.22}$$

that are common values in a ideal environment. Now we define the vector $g_{1\times n}$ using $n = 10$ distances equally log-distributed:

$$g = \begin{bmatrix} 1.0000 \\ 1.2915 \\ 1.6681 \\ 2.1544 \\ 2.7826 \\ 3.5938 \\ 4.6416 \\ 5.9948 \\ 7.7426 \\ 10.0000 \end{bmatrix} \tag{4.23}$$

In Table 4.1 we re-assume the first 4 steps of a simulation. At step zero we start with the initial values. At every step we obtain a measured RSSI-distance pair, and we used it to modify the channel model.

In figure 4.12 we can see how the model changes and tries to get closer to the new measured value. At step 1 the measurement is close to the ideal model and the model does not change a lot. At step 2 the measuredRSSI is lower than the model curve and the model adapts to it and so on to the others steps. Changing the number of n points we can give more or less

weight to the goodness of the measurements. If we decrease the number of points, the new measurement will be more relevant. $n = 10$ seems to be a good trade-off between reactivity and quality of the measurements.

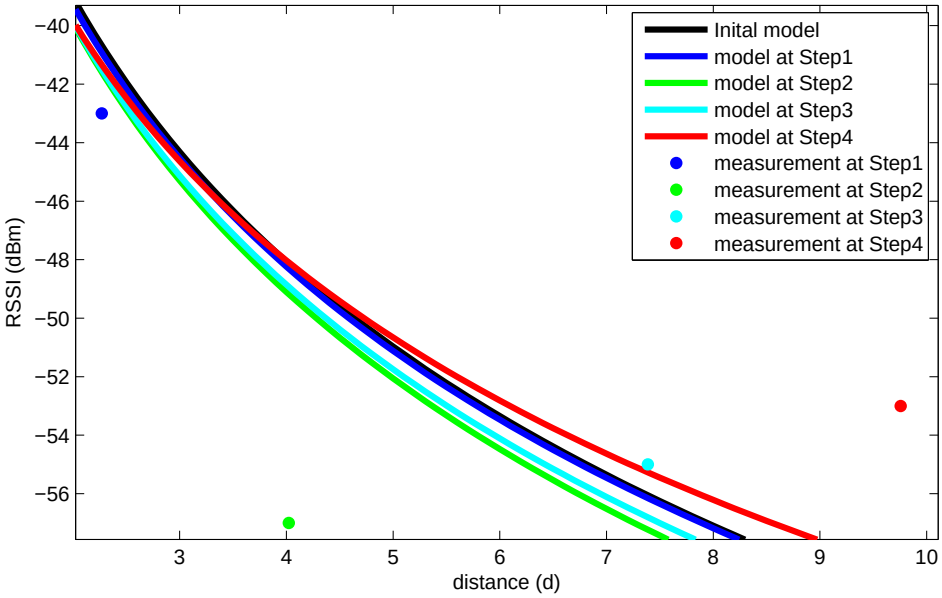


Figure 4.12: First 4 steps of the 1point-MLE (with $n = 10$ points).The Initial channel model (black) adapts itself every step(blu,green,cyan,and red) to new RSSI measurements.

4.5 Defining the Algorithm

In the past section we designed our Model estimator. Now we can arrange an algorithm to get distance using only RSSI and the improved model. In Figure 4.13 you can see the first version of the Algorithm. Basically:

- when the node can start a RF-ranging operation, it acquires a Distance-RSSI pair and uses it to calculate the model. After that it returns a filtered distance.
- If the node cannot start a RF-ranging (because it takes time, or simply it cannot do so), it uses the RSSI value from an arriving packet to calculate a filtered distance using the model measured in the past with distance-RSSI pair.

Once we have designed our algorithm, the next steps are to study and design filters that aim at eliminating eliminate the outliers in the measure-

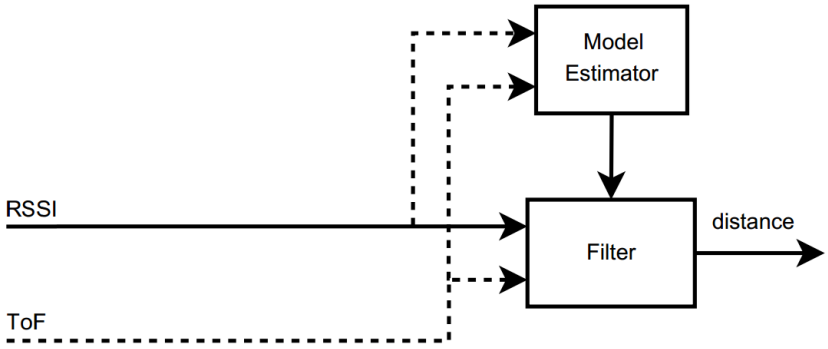


Figure 4.13: Adaptive Radio-Frequency Ranging Algorithm

ments and filters that can improve the estimation of the distance. One other issue is to evaluate how long has to be the period between RF-Ranging operations. For Example we can perform a RF-ranging operation whenever data is available, every second, etc. In Chapter 5 we will study and implement several filters and we will evaluate performance with some experiments. We will post-pone the issue regarding the rate of RF-Ranging operation to Chapter 6 where we implement the Algorithm on real robots.

Chapter 5

Filtering Distance Estimantes

5.1 Why use the word "Filter"?

We can define a filter as the process of finding the "best estimate" from noisy data amounts to "filtering out" the noise. To understand why it is important to filter measurements we will do a motivating example.

Imagine you have a robot that needs to know the distance from another robot B. The first thing the robot does, is to make a *measurement*. Measurements are usually denoted by z. From experience it might be known that the sensor is not so good, in other words the measurements are noisy. You might know that the sensor has an error that is most of the time zero, but sometimes (with a certain frequency)there is a certain range of error. You can model that as a Random variable with a Gaussian probability density function. Now the best you can do is to maintain an estimate or *belief* about your current position. The robot then moves toward B by putting some known voltage on its motors which causes a translation in the direction of B. This motor control data (its actions) is usually denoted by u. We can relate u to the robot's state (because we know that a high voltage leads to a large translation..) and we can summarize this relationship in a matrix C. Thus, Cu describes the effect of the robot's actions to its state. Therefore, the robot can use the previously measured state at time $t-1$ and Cu to make a prediction about the next state that it is about to measure, i.e. its *expectation*. Again: The robot has two sources of information to estimate its own state: 1) its own actions under the assumption that the robot knows how they change the world and 2) its measurements. What is needed is a

tool to combine these two stochastic quantities together to achieve an optimal state estimation. A Kalman filter is an example of such a tool, and in the following we are going to cover the necessary knowledge to understand the Kalman filter equations and to implementing them in our algorithm.

The key idea is to represent uncertainty explicitly, using the calculus of probability theory. Probabilistic approaches are typically more robust in the face of sensor limitations, sensor noise, environment dynamics, and so on. They often scale much better to complex and unstructured environments, where the ability to handle uncertainty is of even greater importance. In fact, certain probabilistic algorithms are currently the only known working solutions to hard robotic estimation problems

This family of filters is collectively called Gaussian Filters. Gaussian techniques all share the basic idea that beliefs are represented by multivariate normal distributions. Multivariate normal distributions are characterized by density functions of the following form:

$$p(x) = det(2\pi\Sigma)^{-\frac{1}{2}} exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \tag{5.1}$$

The commitment to represent the posterior by a Gaussian has important ramifications. Most importantly, Gaussians are uni-modal, that is, they possess a single maximum. Such a posterior is characteristic of many tracking problems in robotics, in which the posterior is focused around the true state with a small margin of uncertainty. On the contrary, Gaussian posteriors are a poor match for many global estimation problems in which many distinct hypotheses exist, each of which forming its own mode in the posterior. However, these advantages come at a price. Traditionally, the two most frequently cited limitations of probabilistic algorithms are *computational inefficiency*, and a *need to approximate*.

In the previous chapter we have shown how RSSI and ToF are merged in order to continually update the channel model and be able to measure the distance between two robots directly through RSSI. In this chapter we want to filter our measurements so as to minimize the error. To do this, we will describe Linear Kalman Filter, Extended Kalman Filter, and Unscented Kalman Filter theory. Then we will show the implementations of these filters: we will do a comparison between them, showing which provides the best results and explaining why.

5.2 The Kalman Filter

Probably the best studied technique for implementing Bayes filters is the Kalman Filter (KF). The Kalman filter was invented in 1950s by Rudolph Emil Kalman as a technique for filtering and prediction in linear systems. The kalman filter implements belief computation for continuous states.It is not applicable to discrete or hybrid state spaces.

The Kalman filter represents beliefs by the moments representation: At time t, the belief is represented by the mean μ_t and the covariance Σ_t. Posteriors are Gaussian if the following three properties hold, in addition to the Markov assumptions of the Bayes filter.

1. The next state probability $p(x_t|u_t, x_{t-1})$ must be a linear function in its arguments with added Gaussian noise. This is expressed by the following equation:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \tag{5.2}$$

wherex_t and x_{t-1} are state vectors, and u_t is the control vector at time t. In our notation, both of these vectors are vertical vectors. A_t and B_t are matrices. A_t is a square matrix of size $n \times n$, where n is the dimension of the state vector x_t. B_t is of size $n \times m$, with m being the dimension of the control vector u_t. By multiplying the state and control vector with the matrices A_t and B_t, respectively, the state transition function becomes *linear* in its arguments. Thus, Kalman filters assume linear system dynamics.

The random variable ε_t is a Gaussian random vector(GRV) that models the randomness in the state transition. It is of the same dimension as the state vector. Its mean is zero and its covariance will be denoted R_t.

Equation 5.3 defines the state transition probability $p(x_t|u_t, x_{t-1})$. This probability is obtained by plugging Equation (5.3) into the definition of the multivariate normal distribution (5.1):

$$p(x_t|u_t, x_{t-1}) = det(2\pi R_t)^{-\frac{1}{2}} exp\left\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t)\right\} \tag{5.3}$$

2. The measurement probability $p(z_t|x_t)$ must also be linear in its arguments, with added Gaussian noise:

$$z_t = C_t x_t + \delta_t \tag{5.4}$$

Here C_t is a matrix of size $k \times n$, where k is the dimension of the measurement vector z_t. The vector δ_t describes the measurement noise. The distribution of δ_t is a multivariate Gaussian with zero mean and covariance Q_t. The measurement probability is thus given by the following multivariate normal distribution:

$$p(z_t|x_t) = det(2\pi Q_t)^{-\frac{1}{2}} exp\left\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t)\right\} \tag{5.5}$$

3. Finally, the initial belief $bel(x_0)$ must be normal distributed. We will denote the mean of this belief by μ_0 and the covariance by Σ_0:

$$bel(x_0) = p(x_0) = det(2\pi\Sigma_0)^{-\frac{1}{2}} exp\left\{-\frac{1}{2}(x_0 - \mu_0)^T \Sigma_t^{-1}(x_0 - \mu_0)\right\} \tag{5.6}$$

These three assumptions are sufficient to ensure that the posterior $bel(x_t)$ is always a Gaussian, for any point in time t.

5.2.1 The Kalman filter algorithm

The Kalman filter algorithm is depicted in Algorithm 1 . In the first two lines the predicted belief $\bar{\mu}$ and $\bar{\Sigma}$ is calculated representing the belief $\bar{bel}(x_t)$ one time step later, but before incorporating the measurement z_t.

In the remaining lines the update step is conducted, by incorporating the measurement z_t. The variable K_t, computed in line 4, is called *Kalman Gain.* It specifies the degree to which the measurement is incorporated into the new state estimate. The Kalman gain is a function of the relative certainty of the measurements and current state estimate, and can be "tuned" to achieve particular performance. With a high gain, the filter places more weight on the measurements, and thus follows them more closely. With a low gain, the filter follows the model predictions more closely, smoothing out noise

Algorithm 1 Algorithm Kalman filter

1: **procedure** ALGORITHM KF($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)
Prediction
2: $\bar{\mu}_t = A_t\mu_{t-1} + B_t u_t$
3: $\bar{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R_t$
Update:
4: $K_t = \bar{\Sigma}_t C_t^T (C_t\bar{\Sigma}_t C_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t)$
6: $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$
7: **return** μ_t, Σ_t
8: **end procedure**

but decreasing the responsiveness. At the extremes, a gain of one causes the filter to ignore the state estimate entirely, while a gain of zero causes the measurements to be ignored.

5.3 Extended Kalman Filter

5.3.1 Formulation

The assumptions of linear state transitions and linear measurements with added Gaussian noise are rarely fulfilled in practice. For example, a robot that moves with constant translational and rotational velocity typically moves on a circular trajectory, which cannot be described by linear next state transitions. This observation, along with the assumption of unimodal beliefs, renders plain Kalman filters, as discussed so far, inapplicable to all but the most trivial robotics problems.

The extended Kalman filter (EKF) overcomes one of these assumptions: the linearity assumption. Here the assumption is that the next state probability and the measurement probabilities are governed by nonlinear functions g and h, respectively:

$$x_t = g(u_t, x_{t-1}) + w_t \tag{5.7}$$

$$z_t = h(x_t) + v_t \tag{5.8}$$

This model strictly generalizes the linear Gaussian model underlying Kalman filters, postulated in Equations (5.2) and (5.4). This function g

replaces the matrices A_t and B_t in (5.2), and h replace C_t in (5.4). Unfortunately with arbitrary functions g and h, the belief is no longer Gaussian. In fact, performing the belief update exactly is usually impossible for nonlinear functions g and h. Thus, the EKF inherits from the KF the basic belief representation, but it differs in that this belief is only approximate, not exact as was the case in Kalman filters.

5.3.2 Linearization in EKF

The key idea underlying the EKF is called *linearization*. Suppose we are given a non linear next state function g. A gaussian projected through this function is typically non-Gaussian. This is because non-linearities in g distort the belief in ways that destroy its Gaussian shape. Linearization approximates g by a linear function that is tangent to g at the mean of the Gaussian. By projecting the Gaussian through this linear approximation, the posterior is Gaussian. In fact, once g is linearized, the mechanics of belief propagation are equivalent to those of the KF. The same argument applies to the multiplication of Gaussians when a measurement function h is involved.

There exist many techniques for linearizing nonlinear functions. EKF utilizes a method called (first order) Taylor expansion. Taylor expansion constructs a linear approximation to a function g from its g's value and slope. The slope is given by the partial derivative:

$$g'(u_t, x_{t-1}) := \frac{\delta g(u_t, x_{t-1})}{\delta x_{t-1}} \tag{5.9}$$

Clearly, both the value of g and its slope depend on the argument of g. A logical choice for selecting the argument is to chose the state deemed most likely at the time of linearization. For Gaussians, the most likely state is the mean of the posterior μ_{t-1}. In other words, g is approximated by its value at μ_{t-1} (and at u_t), and the linear extrapolation is achieved by a term propotional to the gradient of g at μ_{t-1} and u_t :

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + g'(u_t, \mu_{t-1})(x_{t-1} - \mu_{t-1}) = g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}) \tag{5.10}$$

Notice that G_t is a matrix of size $n \times n$, with n denoting the dimension of

	Kalman Filter	EKF
State Prediction	$A_t\mu_{t-1} + B_t u_t$	$g(u_t, \mu_{t-1})$
measurement prediction	$C_t\bar{\mu}_t$	$h(\bar{\mu}_t)$

Table 5.1: Main difference between KF and EKF algorithm

the state. This matrix is often called the *Jacobian*. The value of the Jacobian depends on u_t and μ_{t-1}, hence it differs for different points in time.

EKFs implement the exact same linearization for the measurement function h. Here the Taylor expansion is developed around μ_t, the state deemed most likely by the robot at the time it linearizes h:

$$h(x_t) \approx h(\bar{\mu}_t) + h'(\bar{\mu}_t)(x_t - \bar{\mu}_t) = h(\bar{\mu}_t) + H_t(x_t - \bar{\mu_{t-1}}) \tag{5.11}$$

5.3.3 Extended Kalman Filter Algorithm

In Algorithm 2 states the EKF algorithm. In many ways, this algorithm is similar to the Kalman filter algorithm stated in 1.

Algorithm 2 Extended Kalman Filter Algorithm

1: **procedure** ALGORITHM EKF($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)
Prediction
2: $\bar{\mu}_t = g(u_t, \mu_{t-1})$ ▷ Predicted state estimate
3: $\bar{\Sigma}_t = G_t\Sigma_{t-1}G_t^T + R_t$ ▷ Predicted covariance estimate
Update:
4: $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T$ ▷ Near-optimal Kalman gain
5: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$ ▷ Updated state estimate
6: $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$ ▷ Updated estimate covariance
7: **return** x_t, Σ_t
8: **end procedure**

The most important differences are summarized in table 5.1. That is, the linear predictions in Kalman filters are replaced by their nonlinear generalizations in EKFs. Moreover, EKFs use JacobianG_t and H_t instead of the corresponding system matrices A_t, B_t, and C_t in Kalman filters. The Jacobian G_t corresponds to the matrices A_t and B_t and the JacobianH_t corresponds to C_t.

5.3.4 EKF implementation

Define x_t and z_t

The first thing we have to do is to define the state vector and the state equation of the model we want to describe. We want to estimate the distance between two nodes, so the state vector will be:

$$x = \begin{bmatrix} d \\ \dot{d} \end{bmatrix} \tag{5.12}$$

where d is the estimated distance, and $\dot{d}$ is the discrete-time approximation of the derivative of distance. We include the speed in the state vector in order to have a smoother trajectory, considering the uniform linear motion. The state equations will be:

$$x_t = \begin{bmatrix} d_t \\ \dot{d}_t \end{bmatrix} = \begin{bmatrix} d_{t-1} + \Delta t * \dot{x}_{t-1} + w_d \\ \dot{d}_{t-1} + w_d \end{bmatrix} \tag{5.13}$$

where Δt is the time between consecutive state predictions and w_d is a Gaussian random vector that models the uncertainty introduced by the state transition. Its mean is zero and its covariance will be denoted R_t:

$$R_t = \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ 0 & \Delta t \end{bmatrix} \sigma_w^2 \tag{5.14}$$

where σ_w is the variance of the noise w_d. Then, we have to write the measurement vector and the measurement equations associated to it.When we measure both ToF and RSSI, we use the measurement vector in 5.15 and using the measurement equations in 5.16:

$$z = \begin{bmatrix} \hat{d} \\ \hat{\rho} \end{bmatrix} \tag{5.15}$$

$$z_t = \begin{bmatrix} \hat{d}_t \\ \hat{\rho}_t \end{bmatrix} = \begin{bmatrix} d_t - bias_d + v_d \\ \rho_0 - 10\alpha \log d_t + v_\rho \end{bmatrix} \tag{5.16}$$

where $bias_d$ is the bias of the ToF measurement, v_d and v_ρ describe respectively the measurement Gaussian noise of the distance and of the channel. The mean and the standard deviation will be denoted respectively μ_d, σ_d and μ_ρ, σ_ρ previously calculated in 4.18 and 4.19. The Covariance will be

denoted as Q_t:

$$R_t = \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\rho^2 \end{bmatrix} \tag{5.17}$$

When we have only the RSSI measurement, z will be a scalar and the measurement equation will be:

$$z_t = \rho_0 - 10\alpha \log d_t + v_\rho \tag{5.18}$$

and Q_t will be simply: $Q_t = \sigma_\rho$

Define G_t and H_t

Now that we have defined x_t and z_t in (5.13) and (5.16), we can use them to calculate G_t and H_t introduced in (5.10) and (5.11):

$$G_t = \left.\frac{\partial g}{dx}\right|_{x_{t-1},u_t} = \begin{bmatrix} \frac{\partial d_t}{d_{t-1}} & \frac{\partial d_t}{\dot{d}_{t-1}} \\ \frac{\partial \dot{d}_t}{d_{t-1}} & \frac{\partial \dot{d}_t}{\dot{d}_{t-1}} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{5.19}$$

$$H_t = \left.\frac{\partial h}{dz}\right|_{x_t} = \begin{bmatrix} \frac{\partial \hat{d}_t}{d_t} & \frac{\partial \hat{d}_t}{\dot{d}_t} \\ \frac{\partial \hat{\rho}_t}{d_t} & \frac{\partial \hat{\rho}_t}{\dot{d}_t} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{-10\alpha}{d_t \ln 10} & 0 \end{bmatrix} \tag{5.20}$$

5.4 UKF

The EKF has become a standard technique used in a number of nonlinear estimation and machine learning applications. A central and vital operation performed in the Kalman Filter is the propagation of a Gaussian random variable (GRV) through the system dynamics. In the EKF, the state distribution is approximated by a GRV, which is then propagated analytically through the first-order linearization of the nonlinear system. This can introduce large errors in the true posterior mean and covariance of the transformed GRV, which may lead to sub-optimal performance and sometimes divergence of the filter. The UKF addresses this problem by using a deterministic sampling approach. The state distribution is again approximated by a GRV, but is now represented using a minimal set of carefully chosen sample points. These sample points completely capture the true mean and covariance of the GRV, and when propagated through the true non-linear system, captures the posterior mean and covariance accurately to the 3rd

order (Taylor series expansion) for any nonlinearity. The GRV, in contrast, only achieves first-order accuracy. Remarkably, the computational complexity of the UKF is the same order as that of the EKF. To elaborate on this, we start by first explaining the Unscented Transformation (UT).

5.4.1 The Unscented Transformation (UT)

The Unscented Transformation (UT) is a method for calculating the statistics of a random variable which undergoes a nonlinear transformation. Consider propagating a random variable $\mathbf{x}$ (dimension L) through a nonlinear function, $\mathbf{y} = g(\mathbf{x})$. Assume $\mathbf{x}$ has mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}_{\mathbf{x}}$. To calculate the statistics of $\mathbf{y}$, we form a matrix $\boldsymbol{\mathcal{X}}$ of $2L+1$ sigma vectors $\mathcal{X}_i$ (with corresponding weights W_i), according to the following:

$$\mathcal{X}_0 = \bar{\mathbf{x}} \tag{5.21}$$

$$\mathcal{X}_i = \bar{\mathbf{x}} + \left(\sqrt{(L+\lambda)\mathbf{P}_{\mathbf{x}}}\right)_i \qquad i = 1, \ldots, L \tag{5.22}$$

$$\mathcal{X}_i = \bar{\mathbf{x}} - \left(\sqrt{(L+\lambda)\mathbf{P}_{\mathbf{x}}}\right)_{i-L} \qquad i = L+1, \ldots, 2L \tag{5.23}$$

$$W_0^{(m)} = \lambda/(L+\lambda) \tag{5.24}$$

$$W_0^{(c)} = \lambda/(L+\lambda) + (1-\alpha^2+\beta) \tag{5.25}$$

$$W_i^{(m)} = W_i^{(c)} = 1/\{2(L+\lambda)\} \qquad i = 1, \ldots, 2L \tag{5.26}$$

where $\lambda = \alpha^2(L+\kappa) - L$ is a scaling parameter. α determines the spread of the sigma points around $\bar{\mathbf{x}}$ and is usually set to a small positive value (e.g., 1e-3). κ is a secondary scaling parameter which is usually set to 0, and β is used to incorporate prior knowledge of the distribution of $\mathbf{x}$ (for Gaussian distributions, $\beta = 2$ is optimal). $(\sqrt{(L+\lambda)\mathbf{P}_{\mathbf{x}}})_i$ is the ith row of the matrix square root.

These sigma vectors are propagated through the nonlinear function,

$$\mathcal{Y}_i = g(\mathcal{X}_i) \quad i = 0, \ldots, 2L \quad , \tag{5.27}$$

and the mean and covariance for $\mathbf{y}$ are approximated using a weighted sample

mean and covariance of the posterior sigma points,

$$\bar{\mathbf{y}} \approx \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_i \tag{5.28}$$

$$\mathbf{P}_{\mathbf{y}} \approx \sum_{i=0}^{2L} W_i^{(c)} \{\mathcal{Y}_i - \bar{\mathbf{y}}\} \{\mathcal{Y}_i - \bar{\mathbf{y}}\}^T \tag{5.29}$$

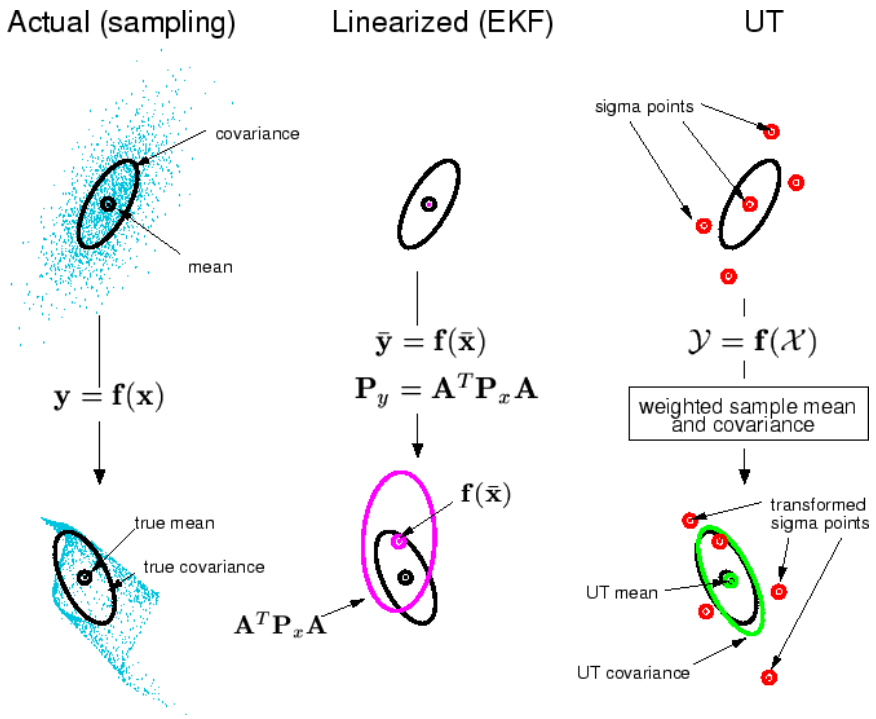


Figure 5.1: Example of the UT for mean and covariance propagation. a) actual, b) first-order linearization (EKF), c) UT.

A simple example is shown in Figure 5.1 for a 2-dimensional system: the left plot shows the true mean and covariance propagation using Monte-Carlo sampling; the center plots show the results using a linearisation approach as would be done in the EKF; the right plots show the performance of the UT (note only 5 sigma points are required). The superior performance of the UT is clear.

5.4.2 UKF algorithm

The UKF equations are given in Algorithm 3 . Note that no explicit calculation of Jacobians or Hessians are necessary to implement this algorithm. Furthermore, the overall number of computations are the same order as the EKF.

Algorithm 3 Extended Kalman Filter Algorithm

1: Initialize with:
$\hat{x}_0 = E[x_0]$
$P_0 = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]$
$\hat{x}_0^\alpha = E[x^\alpha] = [\hat{x}_0^T 00]^T$
$P_0^\alpha = E[(x_0^\alpha \hat{x}_0^\alpha)(x_0^\alpha \hat{x}_0^\alpha)^T] = \begin{bmatrix} P_0 & 0 & 0 \\ 0 & P_V & 0 \\ 0 & 0 & P_n \end{bmatrix}$ ▷ For $k \in 1, \ldots, \infty$

2: Calculate Sigma points:
$\mathcal{X}_{k-1}^\alpha = \left[\hat{x}_{k-1}^\alpha \quad \hat{x}_{k-1}^\alpha \pm \sqrt{(L+\lambda)P_{k-1}^\alpha}\right]$

3: Time Update:
$\mathcal{X}_{k|k-1}^\alpha = F[\mathcal{X}_{k-1}^x, \mathcal{X}_{k-1}^v]$
$\hat{x}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}_{k|k-1}^x$
$P_k^- = \sum_{i=0}^{2L} W_i^{(c)} \left[\mathcal{X}_{i,k|k-1}^x - \hat{x}_k^-\right] \left[\mathcal{X}_{i,k|k-1}^x - \hat{x}_k^-\right]^T$
$\mathcal{Y}_{k|k-1} = \mathcal{H}\left[\mathcal{X}_{k|k-1}^x, \mathcal{X}_{k-1}^n\right]$
$\hat{y}_k^- = \sum_{i=0}^{2L} W_i^{(m)} mathcalY_{i,k|k-1}$

4: Measurement update equations:
$\mathcal{P}_{\bar{y}_k \bar{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} \left[\mathcal{Y}_{i,k|k-1} - \hat{y}_k^-\right] \left[\mathcal{Y}_{i,k|k-1} - \hat{y}_k^-\right]^T$
$\mathcal{P}_{x_k y_k} = \sum_{i=0}^{2L} W_i^{(c)} \left[\mathcal{X}_{i,k|k-1} - \hat{x}_k^-\right] \left[\mathcal{Y}_{i,k|k-1} - \hat{y}_k^-\right]^T$
$\mathcal{K} = \mathcal{P}_{x_k y_k} \mathcal{P}_{\bar{y}_k \bar{y}_k}^{-1}$
$\hat{x}_k = \hat{x}_k^- + \mathcal{K}(y_k - \hat{y}_k^-)$
$\mathcal{P}_k = \mathcal{P}_k^- - \mathcal{K} - \mathcal{P}_{\bar{y}_k \bar{y}_k} \mathcal{K}^T$

where $x^\alpha = [x^T v^T n^T]^T$, $\mathcal{X}^\alpha = [(\mathcal{X}^{\S^T})(\mathcal{X}^{\sqsubseteq^T})(\mathcal{X}^{\setminus^T})]^T$, $\lambda =$ composite scaling parameter, $L =$ dimension of augmented state, $\mathcal{P}_v =$process noise covariance, $W_i =$ weights as calculated before.

5.5 Implementation on Matlab

5.5.1 Simulation Setup

In the previous section we described three important Filters: KF, EKF, and UKF. It is known from the theory that the Kalman filter is optimal in case that a) the model perfectly matches the real system, b) the entering noise is white and c) the covariances of the noise are exactly known. but it requires that the model is linear.

We implemented the EKF and UKF algorithm in MATLAB using Equations and matrix described in 5.3.4. Then, We created a simulation using our previously collected data. The simulation consists of two nodes, one is static at 0, and the other is moved to different positions (in the range of $1m-10m$) according to a circular buffer. For each of the positions, a random measurement is selected from our dataset, and used to calculate an estimate of the model parameters. A ToF measurement is selected after every 9 RSSI measurements. When a ToF is received, the model trusts the new measurement and estimates the distance. Then it updates the channel model with our model estimator. When only RSSI is received it simply estimates the distance.

5.5.2 Results

The nanoLOC device moves to different positions. In Figure 5.2 you can see the real Distance (blue line). You can also see the two Implementations: EKF (black line) and UKF (magenta line). From that figure we can see that the model changes following the real distance very fast. This is because we trust in the ToF Measurements.

In Figure 5.3 you can see the mean error between real distance and estimated distance with the EKF implementation (black) and UKF implementation (magenta). From this and from the previous figure you can see that EKF and UKF are really similar. UKF is better only in few moments: For example at second 38, after a fast movement from 9 meters to 1 meter, UKF better fits the real distance.

Figure 5.3 gives us another important information: The mean error has zero mean, thus we obtained a good accuracy of the estimated distance.

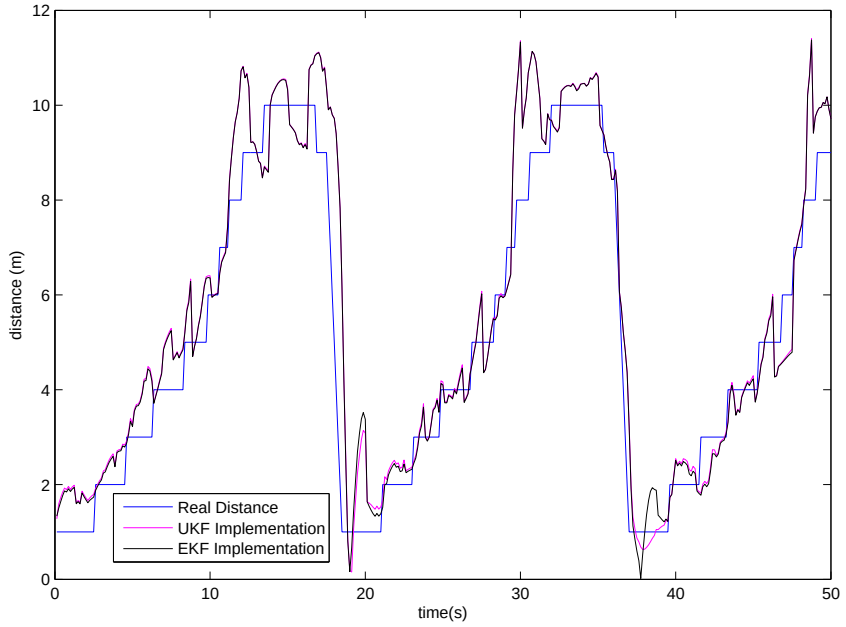


Figure 5.2: Real Distance(blue), Estimated distance with EKF(black),and UKF(magenta)

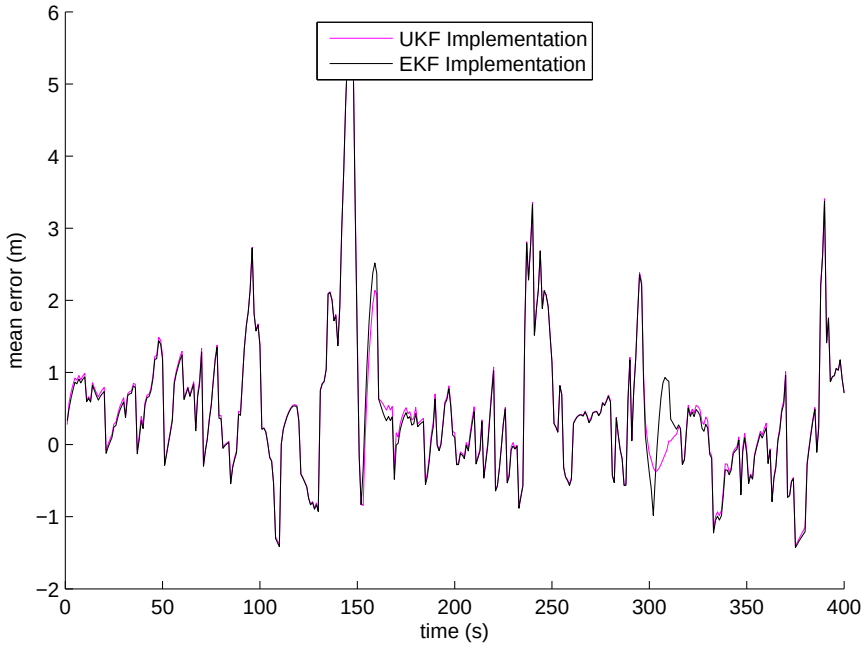


Figure 5.3: Difference between Estimated distance and Real Distance

5.6 To UKF or not to UKF?

In the previous section we showed a comparison between an EKF and an UKF implementation. We noticed that the two implementation are almost the same. This is why unless the system is "very non-linear" the benefits of an UKF implementation are very small. So the question is: to use UKF or not?

Usually it is better to use UKF when f and/or h are non linear and when there are difficulties in EKF implementation or poor EKF performances.

Instead, go for a KF implementation if the system is linear (KF is optimal if the model is linear); use EKF if the model is non-linear, it exhibits an acceptable performance and the computations are lighter than using UKF.Finally, consider using others filters if the model is *"strange"* (f and h too non-linear, distribution are e.g. bimodal).

5.7 Filtering the RSSI readings

In indoors, the RSSI readings experience large fluctuations, even when the robots are static, due to complex propagation phenomena. For a group of mobile nodes, this instability becomes even harder to handle. Therefore, in order to filter those fluctuations, we use a sliding window median filter.

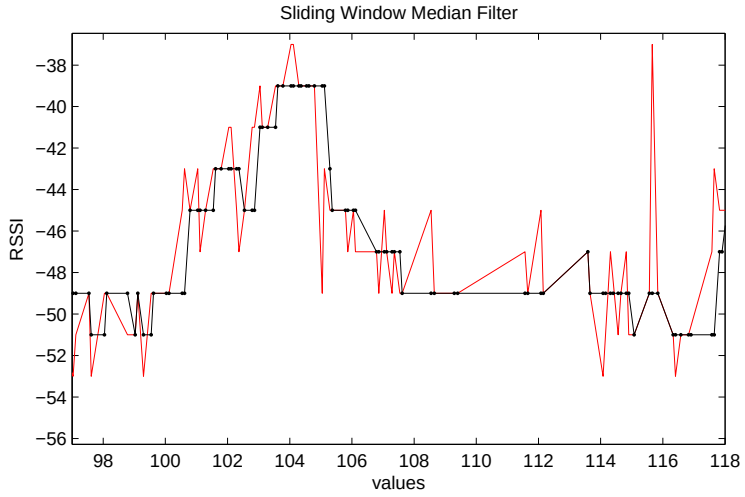


Figure 5.4: Sliding window median filter (k = 5): The median filter(black) filters out the outliers in the raw measurements(red)

Whenever an RSSI reading is received, the measured value goes through the filter that returns the median of the last k measurements. This may affect response time to true variations on the RSSI of moving robots, therefore a small value of k should be used. The main difference between using just a sliding window and a more complex filter as the KF is that the latter not only cleans up the data measurements but also projects these measurements onto the state estimate.

In Fig. 5.4 we can see an example of the sliding window median filter with $k = 5$. The median filter filters out the outliers in the raw measurements. As you can see this filters removes all the spikes and gives a smoother curve.

We repeated the comparison between the EKF and UKF implementation including the Sliding Window Median Filter. In Figure 5.5 we can see that the error in both EKF and UKF implementations is slightly reduced. This kind of filter is extremely simple to implement (a simple vector of k elements).

5.8 Final Algorithm

In this chapter we studied and implemented several filters. Based on these, we can now propose a method for the estimation of distances between robots using RF transmissions, i.e., our RF-adaptive algorithm. In Figure 5.7 you can see the final version of the algorithm. It is composed of three main blocks:

- The Sliding Window Median filter applied to the raw RSSI data
- The log-distance path loss model estimator
- The Extended Kalman Filter to estimate the distance between robots

The First block helps us to improve the quality of the raw RSSI measurements. When only an RSSI measurement is received, the algorithm uses the past channel model and the EKF to estimate the distance. When a ToF measurement is received we can update the model estimator and at the same time estimate the distance with the EKF filter.

Updating the channel model is the most powerful action in this algorithm because in this way the algorithm adapts its parameters to the environment.

For each robot we will have several parameters that give us the best approximation of the channel model for every other robots it communicates

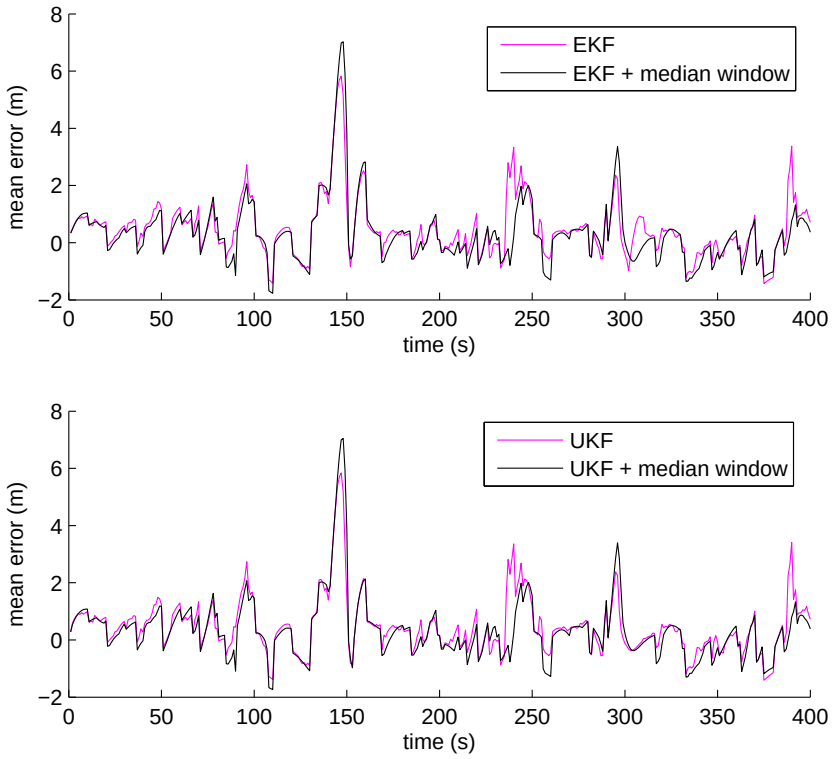


Figure 5.5: Difference between EKF and UKF without and with median window.

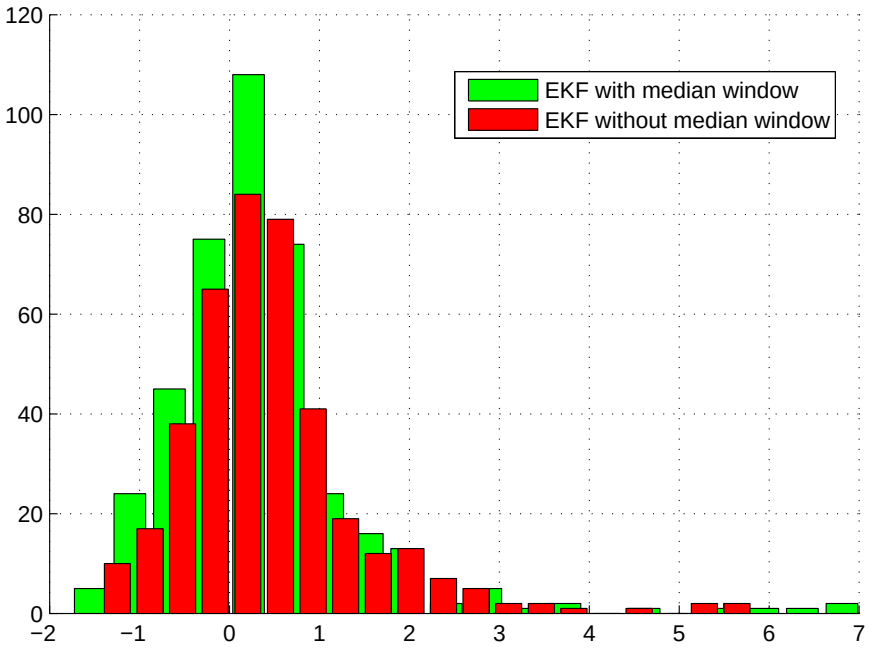


Figure 5.6: Error Distribution in EKF implementations with and without median window.

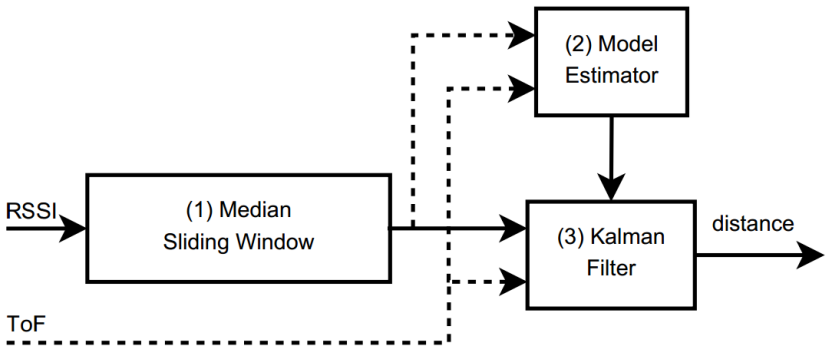


Figure 5.7: Final version of the Adaptive RF-Ranging Algorithm

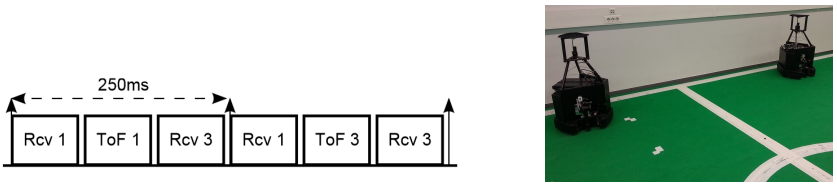
with. For example if we have two robots in a courtyard and a third in a building, the robots outside will store two very different channel models, one for the robot outside and one for the robot inside.

Chapter 6

Algorithm-Implementation on a real environment

6.1 Experiment setup

We programmed the nanoLOC devices with the software developed for, which synchronized the communications with an adaptive TDMA scheme. In our setup, we used three such units with a communication period of $250ms$ (Fig. 6.1a). Consequently, in the absence of communication failures, each node ranges one different node every $250ms$, and receives one communication from each node between ranges. Those three nodes were then placed on top of three robots (Fig. 6.1b) in an indoor laboratory (approx. $20m \times 6m$), with a small ($9.90m \times 5.75m$) soccer field. There, the robots are able to localize themselves using an omnidirectional camera, which we consider as our ground-truth distance.



(a) (b)

Figure 6.1: (a) Communication period as seen by node 2: Receives broadcast from node 1,ranges node 1, receives broadcast from node 3, receives broadcast from node 1, ranges node 3, receives broadcast from node 3, and repeats. (b) Robots in the soccer fields.

Robot 1 and Robot 3 where stopped in each side of the mid-field and robot 2 was moved manually (remote control) to perform the trajectory, see Fig.6.2.

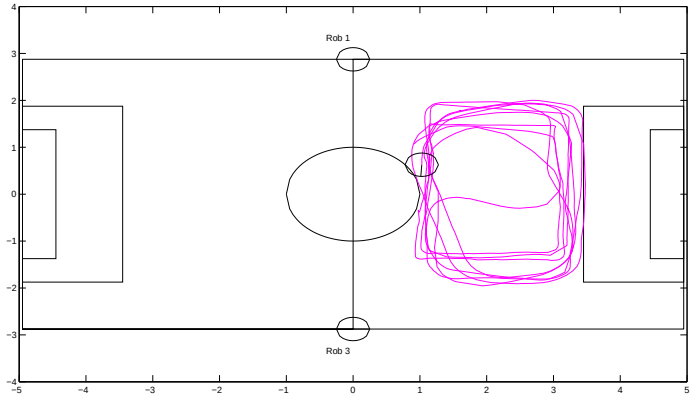


Figure 6.2: Soccer field where experiments were made: Robot 1 on the top middle field, Robot 3 on the bottom mid-field, Robot 2 moving along the magenta trajectory

6.2 Five different approaches

We logged the data from three experiments containing ground truth, ToF distances, and RSSI measurements. Then, we post-processed them using five different approaches, since we want to show that our system correctly adapts to a new communication environment:

1. Using a corridor indoor propagation model with only RSSI
2. Using the pre-calculated lab propagation model with only RSSI
3. Using the online estimator whenever data is available
4. Using the online estimator every second
5. Using the online estimator every ten seconds

In the first approach we set both models to the parameters corresponding to the corridor environment evaluated in the indoor experiment carried out in Chapter 4. The corridor model is redefined in Eq. (6.1). In this way we want to show how RSSI performs in a standard indoor environment.

$$\begin{bmatrix} \rho_0 \\ \alpha \end{bmatrix} = \begin{bmatrix} -37.6455 \\ 2.1849 \end{bmatrix} \tag{6.1}$$

In the second approach, the models were set to the parameters corresponding to the lab environment (Eq. (6.2)). In this way, we want to show the quality of the estimation with a pre-calculated propagation model. Note that the corridor model is very different from the model estimated in the field for either robot.

$$\begin{bmatrix} \rho_0 \\ \alpha \end{bmatrix}_{robot1} = \begin{bmatrix} -38.1485 \\ 1.6505 \end{bmatrix}, \begin{bmatrix} \rho_0 \\ \alpha \end{bmatrix}_{robot2} = \begin{bmatrix} -39.6955 \\ 1.1558 \end{bmatrix} \tag{6.2}$$

Finally, the last three approaches aim at testing the adaptability of the model estimation algorithm to a different environment. Therefore, in spite of the robots being located in the lab environment, the initial channel parameter values were set on purpose to the values in Eq. (6.1) corresponding to the corridor environment.

Note that the behaviour in all three experiments was similar, favouring their confidence level. Therefore, only plots from the first experiment are presented. Larger differences would be noticeable with sudden changes in the environment, as when a robot crosses a door and enters a corridor. In this case, the higher the rate of ToF rangings, the higher the reactivity of the model adaptation.

6.3 Results

We use an online channel model estimator to improve the accuracy of RSSI-based distance measurements. However, in order to estimate the true channel parameters, we would need to take measurements at several distances. In our case, the robots have access to a small observation window,only, in a certain frame Δt. Therefore, the estimated channel will not be the true channel, but rather a local approximation about a given distance. Despite that, if we can obtain parameters that approach the true channel locally, then we can estimate correct distances from the RSSI measurements. In order to prove the capabilities of our channel estimation algorithm to adapt to the time-varying channel conditions we have plotted in the 15-sample average of $10^{(\rho_0 - medianrssi)/(10\alpha)} - d_{gT.}$ (Fig.6.3).

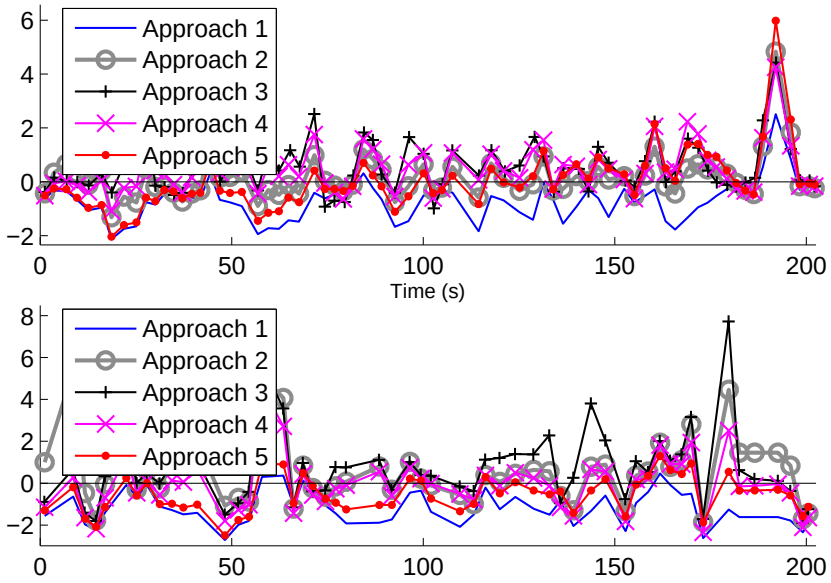


Figure 6.3: Error imposed by the communication channel model on the accuracy of RSSI-based measurements ($10^{(\rho_0 - medianrssi)/(10\alpha)} - d_{gT.}$) : (top) Robot 1; (bottom) Robot 3

This data represents the error imposed by the communication channel model on the accuracy of RSSI-based distance measurements. When the corridor model is used (blue line with no markers), the distance is always underestimated, i.e. is biased, and since this bias will vary with the environment it cannot be filtered. Consequently if we change the environment, the wrong model will degrade our estimate.

When the lab model is used (grey line with 'o' markers), the results are substantially improved, the estimation bias tends to oscillate around the zero error instead of being negative.

The third and fourth approaches (black line with '+' markers, and magenta line with 'x' markers respectively) produce a result very similar to the lab model,which implies that the model is locally correct.

The fifth approach (red line with '.' markers) initially is very similar to the corridor model. This was expected, since it only estimates the model every ten seconds. Despite that, in the end it behaves very similar to the lab model, which means that it converged to a locally correct model.

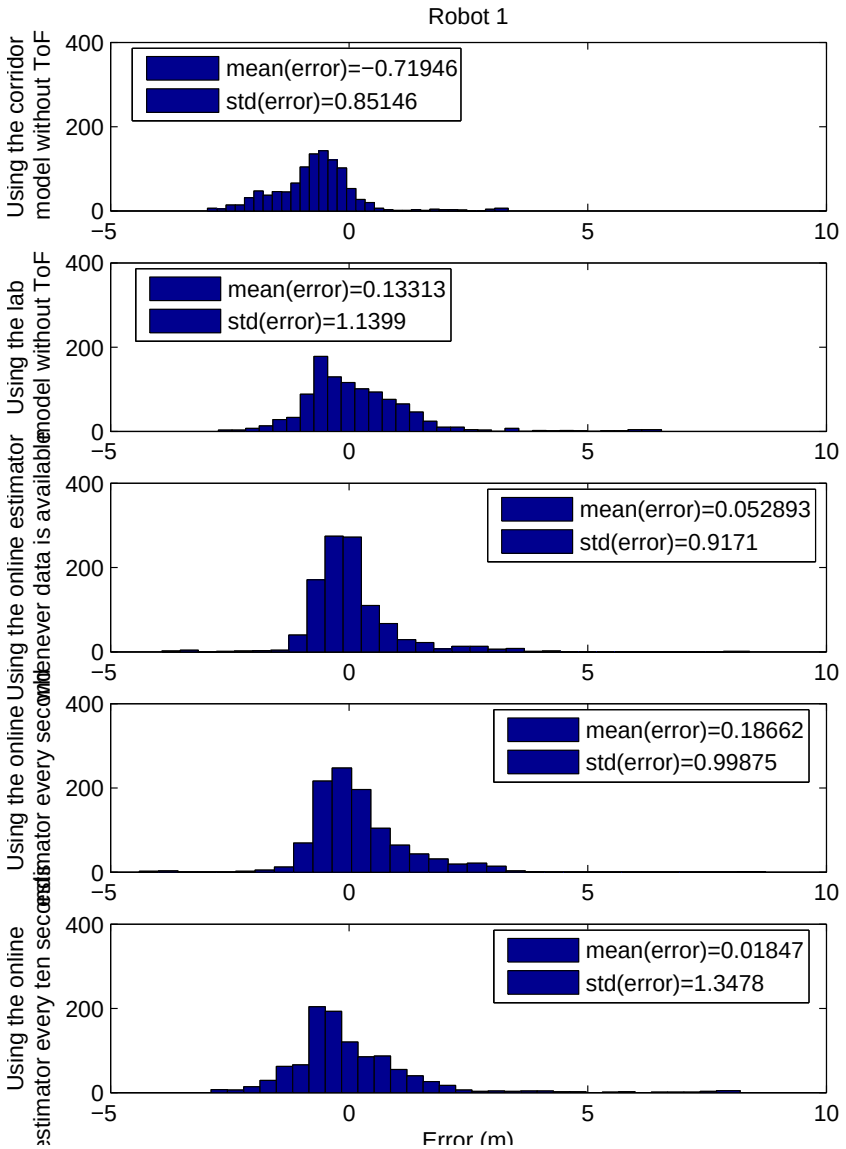


Figure 6.4: Error distribution of the estimated distance between robot 1 and 2

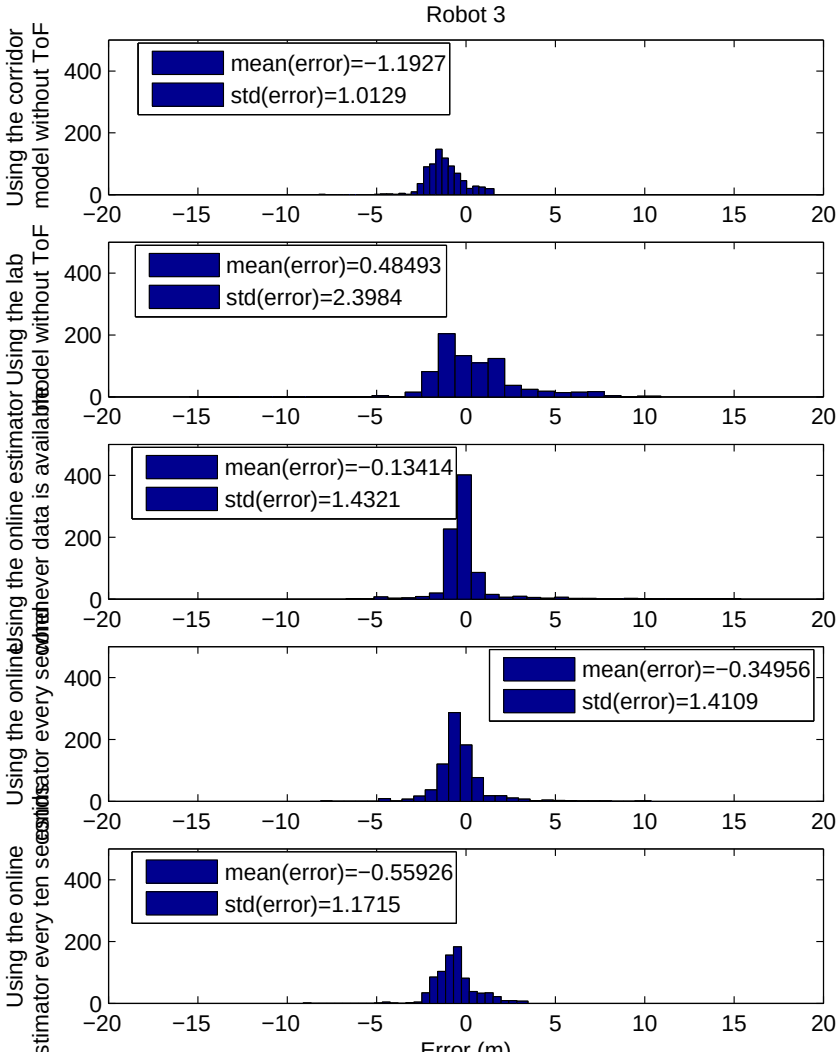


Figure 6.5: Error distribution of the estimated between robot 2 and 3

The effect of these different approaches on the estimated distance can be seen in Table 6.1 and Table 6.2 , that summarises the results of the three experiments for each robot. Figures 6.4,and 6.5 present the distribution of the errors on experiment 1 using the five different approaches. As expected from the previous results, when the robots are using the corridor model, the kalman filter produces an error with a large bias. Moreover, when we compare our online estimator with the lab model, we can still improve on those results. That can be justified by the usage of the highly accurate ToF ranging on the data fusion. Finally, by comparing the three approaches

		Appr.1	Appr. 2	Appr. 3	Appr. 4	Appr. 5
Exp 1	mean	-0.7195	0.1331	0.0529	0.1866	0.0185
	std	0.8515	1.1399	0.9171	0.9988	1.3478
Exp 2	mean	-0.8199	0.1033	0.0457	0.1024	0.0431
	std	0.7640	0.9319	0.8025	0.8613	0.9024
Exp 3	mean	-0.8086	0.0778	0.0394	0.1417	0.0452
	std	0.6918	0.8676	0.8075	0.8637	0.8233

Table 6.1: Results from the three experiments - Mean and Std of the estimated distance between Robot 1 and Robot 2 for each approach

		Appr.1	Appr. 2	Appr. 3	Appr. 4	Appr. 5
Exp 1	mean	-1.1927	0.4849	-0.1341	-0.3496	-0.5593
	std	1.0129	2.3984	1.4321	1.4109	1.1715
Exp 2	mean	-1.2319	0.3017	-0.1866	-0.3893	-0.5199
	std	0.9976	2.7022	1.3593	1.9220	1.2927
Exp 3	mean	-1.2279	0.1954	0.0151	-0.1349	-0.6710
	std	0.9923	2.329	1.9418	1.8334	1.3242

Table 6.2: Results from the three experiments - Mean and Std of the estimated distance between Robot 3 and Robot 2 for each approach

of the online estimation, we can see that by increasing the number of ToF ranges we can improve the results estimation. This was expected because of the high accuracy of ToF when compared with RSSI ranging. However, we also show that if the medium is constant enough that allows for a small number of channel estimates, we can still have a good accuracy with RSSI only. consequently, depending on the conditions the robots are expected to operate in, we can trade-off accuracy for bandwidth. If we have a high number of ToF rangings, we have more accuracy, if we have less ToF rangings we have less accuracy. Note that each ranging uses 20ms, in which the robots cannot communicate.

6.3.1 Considerations about Online-Channel-model

In Fig. 6.6 and 6.7 we can see how the channel model changes in all the presented approaches. Robot 2 estimates online the channel model with both robot 1 (6.6) and Robot 3 (6.7). For each figure we can see plotted the measured parameters ρ_0 and α compared to those of the lab model. As expected when the 3th approach is used ρ_0 and α change very fast and adapt to better estimate the true channel locally. The 4th and the 5th approaches

change their parameters slower than the 3th approach but, as can we see, they converge to the lab model parameters(blues lines).

These figures represent one of the key factors of this Adaptive RF-ranging algorithm. Doing an online estimation of the channel model allows us having the best approximation of the model without having to worry about the environment in which we find ourselves. If the robot moves from an indoor environment to outdoors, the algorithm continuously adapts to the environment. Using longer RF-ranging periods, like the 5th approach($t = 10s$) only degrades the accuracy in rapid changes in the environment, but will give to us more bandwidth gain.

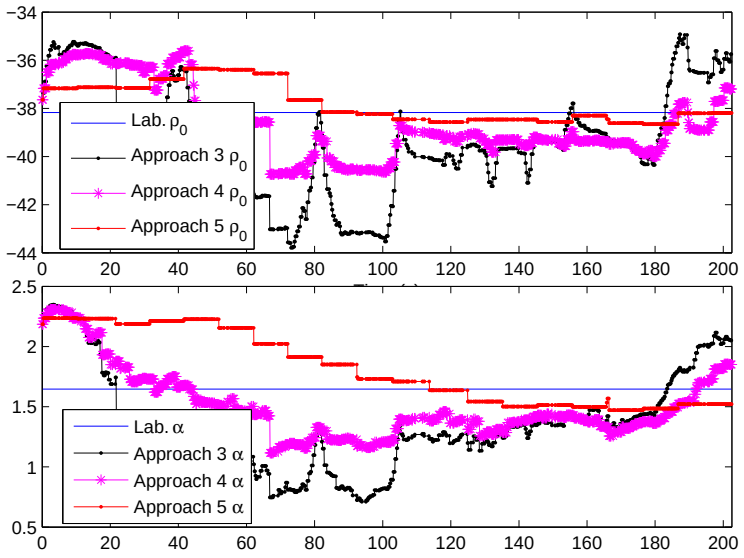


Figure 6.6: Comparison between pre-estimated ρ_0 (2nd approach) and estimated ρ_0 in approaches 3,4,5 between Robot 1 and Robot 2

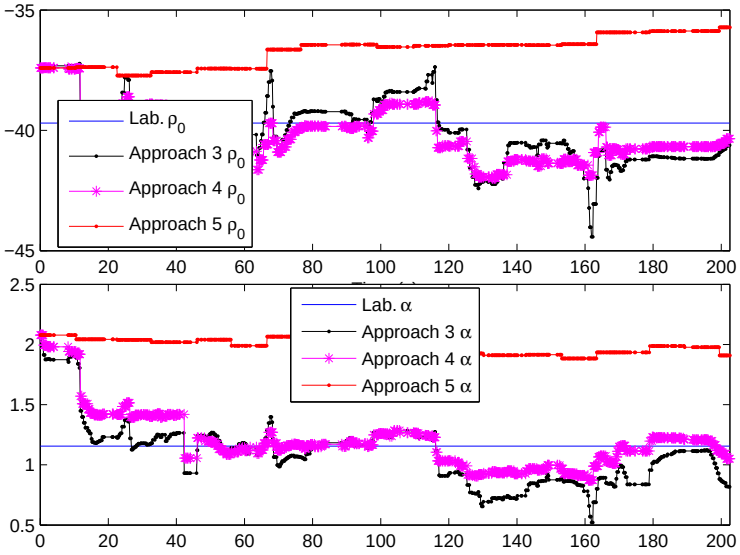


Figure 6.7: Comparison between pre-estimated ρ_0 (2nd approach) and estimated ρ_0 in approaches 3,4,5 between Robot 2 and Robot 3

Chapter 7

From distance to location

In the previous chapters we have designed and implement an algorithm to estimate the distance between two nodes. However, it is interesting to note how accurate the algorithm is in relation to relative location between nodes. As we said in 2 there are two types of location,relative and absolute. However, indifferently by both, there are several localization algorithms based on Ranging like:

- Triangulation
- Trilateration
- Multi-Lateration

7.1 Triangulation

In trigonometry and geometry, triangulation is the process of determining the location of a point by measuring angles to it from known points at either end of a fixed baseline, rather than measuring distances to the point directly (trilateration). The point can then be fixed as the third point of a triangle with one known side and two known angles.

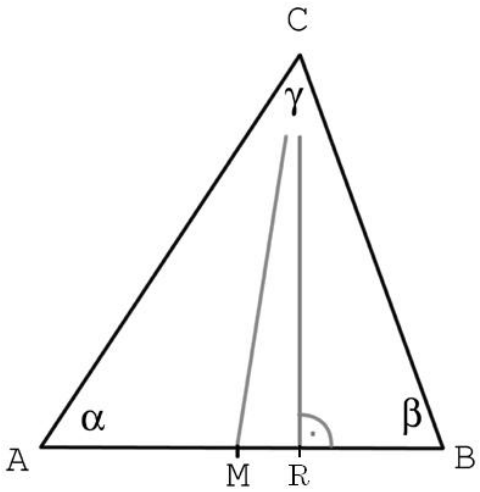


Figure 7.1: Triangulation may be used to calculate the coordinates of C. With α, β, BR or the coordinates of A and B known, then the law of sines can be applied to find the coordinates at C

Let's consider the triangle in Fig. 7.1. By knowing the two angles α and β and the distance AB we can obtain the distance between our node C and the line joining the vertices A and B, that is the height of the triangle:

$$AB = \frac{CR}{\tan\alpha} + \frac{CR}{\tan\beta} \tag{7.1}$$

Therefore

$$CR = \frac{AB}{\frac{1}{\tan\alpha} + \frac{1}{\tan\beta}} \tag{7.2}$$

Using the trigonometric identities $\tan\alpha = \sin\alpha/\cos\alpha$ and $\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$, this is equivalent to:

$$CR = \frac{AB\sin\alpha\sin\beta}{\sin(\alpha+\beta)} \tag{7.3}$$

From this, it is easy to determine the distance of the unknown point C from either observation point,AR and RB distances, and finally its full coordinates.

7.2 Trilateration

In geometry, trilateration is the process of determining absolute or relative locations of points by measurement of distances, using the geometry of circles, spheres or triangles. In a localization system, the centers of the spheres are given by coordinates of the reference points, while the radii are the distances detected by the node compared to the reference points nearby. The intersection of the spheres thus identifies the position of the node. In a reference system installed on a flat surface, such as a floor of a building, eliminating the z axis, the problem is simplified to finding the intersection of three circles.

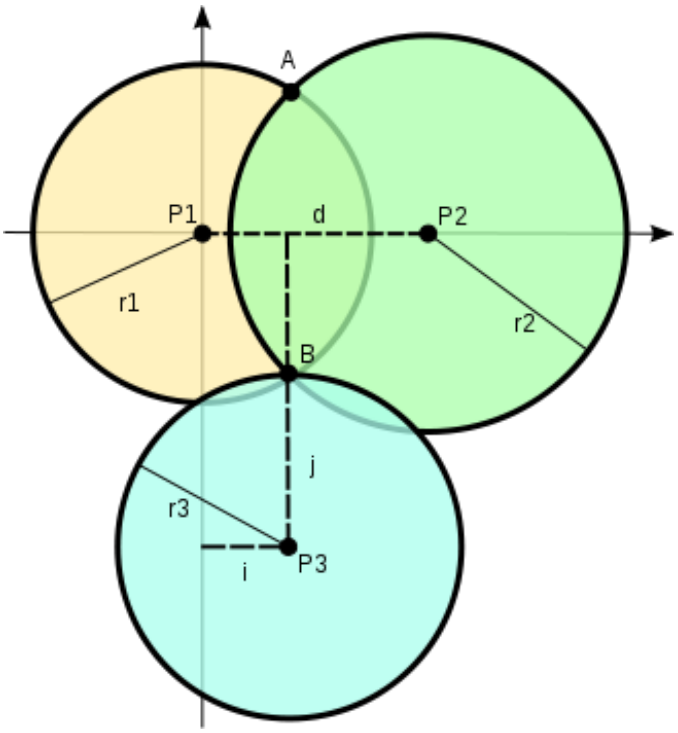


Figure 7.2: with $z = 0$, three circle with center in $P1, P2, P3$ and radius $r1, r2, r3$(distances d, i, j) you can find B as intersection of the three circles.

We start with the equations for the three spheres:

$$r_1^2 = x^2 + y^2 + z^2, \tag{7.4}$$
$$r_2^2 = (x-d)^2 + y^2 + z^2, \tag{7.5}$$
$$r_3^2 = (x-i)^2 + (y-j)^2 + z^2 \tag{7.6}$$
$$\tag{7.7}$$

We need to find a point located at (x, y, z) that satisfies all three equations. First we subtract the second equation from the first and solve for x:

$$x = \frac{r_1^2 - r_2^2 + d^2}{2d}. \tag{7.8}$$

We assume that the first two spheres intersect in more than one point, that is:

$$d - r_1 < r_2 < d + r_1. \tag{7.9}$$

In this case substituting the equation for x back into the equation for the first sphere produces the equation for a circle, the solution to the intersection of the first two spheres:

$$y^2 + z^2 = r_1^2 - \frac{(r_1^2 - r_2^2 + d^2)^2}{4d^2}. \tag{7.10}$$

Substituting $z^2 = r_1^2 - x^2 - y^2$ into the formula for the third sphere and solving for y there results:

$$y = \frac{r_1^2 - r_3^2 - x^2 + (x-i)^2 + j^2}{2j} = \frac{r_1^2 - r_3^2 + i^2 + j^2}{2j} - \frac{i}{j}x. \tag{7.11}$$

Now that we have the x- and y-coordinates of the solution point, we can simply rearrange the formula for the first sphere to find the z-coordinate:

$$z = \pm\sqrt{r_1^2 - x^2 - y^2}. \tag{7.12}$$

Now we have the solution to all three points x, y and z. Because z is expressed as the positive or negative square root, it is possible to get zero, one or two solutions to the problem.

This last part can be visualized as taking the circle found from intersecting the first and second sphere and intersecting that with the third sphere.

If that circle falls entirely outside or inside of the sphere, z is equal to the square root of a negative number: no real solution exists. If that circle touches the sphere on exactly one point, z is equal to zero. If that circle touches the surface of the sphere at two points, then z is equal to plus or minus the square root of a positive number.

7.3 Multi-lateration

As can be seen from its name the multilateration is the generic case of trilateration in which can have more than just 3 points of reference. In a localization system in which the measures of distance with respect to the reference points are subject to errors, it can be very useful to use this calculation technique, which allows taking into account the measures compared to all the reference points detected by the mobile node.

Lets consider a mobile node that collects n distances to n nodes. For simplicity let us consider that the robots move in the same plane with z=0. This results in a system of n equations.Each equation is a circle with center (x_i, y_i) (i-th node) and radius r_i (distance between the mobile node and the i-th node).

$$\begin{cases} (x_1 - x)^2 + (y_1 - y)^2 = r_1^2 \\ (x_2 - x)^2 + (y_2 - y)^2 = r_2^2 \\ \vdots \\ (x_n - x)^2 + (y_n - y)^2 = r_n^2 \end{cases} \tag{7.13}$$

therefore, if we subtract the last equation to the others $n-1$ equation you obtain:

$$\begin{cases} x_1^2 - x_n^2 - 2(x_1 - x_n)x + y_1^2 - y_n^2 - 2(y_1 - y_n)y = r_1^2 - r_n^2 \\ x_2^2 - x_n^2 - 2(x_2 - x_n)x + y_2^2 - y_n^2 - 2(y_2 - y_n)y = r_2^2 - r_n^2 \\ \vdots \\ x_{n-1}^2 - x_n^2 - 2(x_{n-1} - x_n)x + y_{n-1}^2 - y_n^2 - 2(y_{n-1} - y_n)y = r_{n-1}^2 - r_n^2 \end{cases} \tag{7.14}$$

Now if we re-order the therms inside the equation we can solve the system in the form $Ax = b$ where:

$$A = \begin{bmatrix} 2(x_1 - x_n) & 2(y_1 - y_n) \\ 2(x_2 - x_n) & 2(y_2 - y_n) \\ \vdots & \vdots \\ 2(x_{n-1} - x_n) & 2(y_{n-1} - y_n) \end{bmatrix}, b = \begin{bmatrix} x_1^2 - x_n^2 + y_1^2 - y_n^2 + r_n^2 - r_1^2 \\ x_2^2 - x_n^2 + y_2^2 - y_n^2 + r_n^2 - r_2^2 \\ \vdots x_{n-1}^2 - x_n^2 + y_{n-1}^2 - y_n^2 + r_n^2 - r_{n-1}^2 \end{bmatrix} \tag{7.15}$$

Finally we can solve the system with the formula $x = (A^T A)^{-1} A^T b$ where x is a vector with the abscissa and ordinate (x_m, y_m). If measurements are without errors, all the circles touch in one point. In a more realistic case all the measurements have an error and most of the circles intersect in more than one point creating an area of possible solutions. This algorithm gives the most likely solution.

7.4 Multidimensional Scaling

The techniques we have seen so far estimate the position of each mobile node M_n independently, for $n = 1, \ldots N$. In these techniques, the relationship between the ensemble of the nodes was neglected. In some cases, it is desiderable to estimate the position of multiple nodes jointly. We want to estimate the spatial topology of the entire network of nodes simultaneously. This problem may be efficiently solved by using the so called Multi-Dimensional Scaling (MDS) Algorithm.

MDS is a modern technique for visualizing data in a multi-dimensional space. It is a means of visualizing the level of similarity of individual cases of a dataset. Applications include scientific visualisation and data mining in fields such as cognitive science, information science, psychophysics, psychometrics, marketing and ecology. New applications arise in the scope of autonomous wireless nodes that populate a space or an area. MDS uses pairwise "similarity" or "dissimilarity" measures. It orders multidimensional objects by mutual similarity (special case of ordination). The algorithm takes as input data pairwise (dis)similarities (e.g. distances) and returns a set of coordinates(a local map) (Fig. 7.4).

Given N nodes in two (or three dimensions), and estimated pairwise distances $\hat{\delta_{l,q}}$ between nodes l and q, MDS recovers the nodes coordinates $x_n = [x_n, y_n]^T$ minimizing the mismatch between the estimated distances

Figure 7.3: Multidimensional Scaling

$\hat{\delta_{l,q}}$ and the distances $d_{l,q}(x_l, x_q)$ corresponding to the unknown coordinates $x_n, n = 1, \ldots, N$. The mismatch is called *Stress Function*.

Given exact range measurements, the entire spatial topology of the network is perfectly recovered. The original distances are preserved exactly. It is important to note that the coordinates are not "absolute", but "relative". They are recovered up to a distance preserving (isometric) transformation. Such transformations are called "rigid motions": rotations, reflections (orthogonal matrices) and translations. They can be computed in closed-form based on the known locations of at least three reference nodes (or 4, in 3D). In practice, not all pairwise distances are known. To obtain the complete map, smaller maps are computed separately and stitched together by performing rigid motions. We are only allowed to perform translations, rotations and flips.

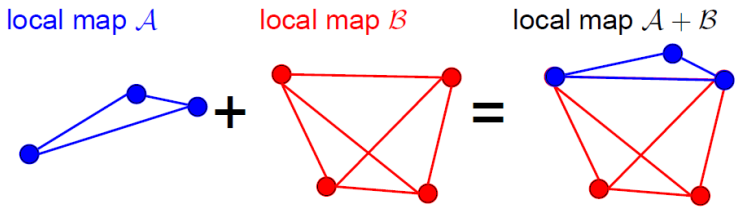


Figure 7.4: Stitching local maps together

If the estimated distances are free of errors, the result will be the exact map. However, we now that range estimation is subject to very large errors. Preserving the exact distances is not possible (e.g. incompatible set of distances). Therefore, the metric assumption becomes unnecessary. and different transformations of the distance may be used. Estimated distances $\hat{\delta_{l,q}}$ are replaced by pseudo-distances $f(\hat{\delta_{l,q}})$.

One approach is Ordinal Multi-Dimensional Scaling. Instead of attempting to preserve the exact distances between nodes, it tries to preserve their

ordering, only, using a monotonically increasing distance mapping function f (called pseudo-distance or disparity). f can be determined experimentally (e.g. by using isotonic regression, monotone splines, etc.). It achieves better performance than the metric MDS (and lower stress values) Another approach is *Weighted MDS* where accuracies r_n are assigned to each node n's position. Nodes are then classified as follows:

- A anchor nodes, perfect knowledge of their location $r_n = 1$
- B unknown nodes, imperfect or no a priori position information $0 <= r_n < 1$
- the total number of nodes is $N = A + B$

We can generalize if we consider that J estimates $\hat{\delta}_{l,q}^{j}, j = 1, \ldots, J$ are available for each distance $d_{l,q}$.

7.4.1 Adjusting the relative coordinates

So far, we discussed the relative position of a team of mobile nodes with no physical anchor. However, for the MDS algorithm, a small perturbation in the distance matrix would bring totally different results for the coordinates X. One of the causes for such behaviour is the way MDS sorts out certain ambiguities that are inherent to the relative localization process, e.g. eigenvector switching causes map flips. Since the nodes position is only recovered up to rigid motion, orientation of the team cannot be determined just with pair-wise distances, neither can symmetry relationships. To obtain relative positions estimates that vary smoothly, we carry out the following adjustments of the coordinates provided by the MDS (considering only the result presented in 2D space, i.e. $m = 2$) , as suggested in [2]. Let $R = [r_{ij}] = (r_0, r_1, \ldots, r_{n-1})$ denote the coordinates determined with MDS and $S = [s_{i,j}]_{n \times 2} = (s_0, s_1, \ldots, s_{n-1})$ denote the final coordinates. We consider the three nodes with the smallest IDs as being local references (here in referred as 0,1, and 2)(Fig. 7.5a). The coordinates adjustment includes shift (Fig. 7.5b), rotation (Fig. 7.5c) and reflection (Fig. 7.5d) so that node 0 is at the origin point (0,0), node 1 on the positive y-axis and node 2 on the right half-plane. Thus, we first let $s_0 = (0, 0)$, and compute the clockwise angle α from vector $(\bar{r}_1 - \bar{r}_0)$ to y-axis, as in Equation 7.16 deducing an

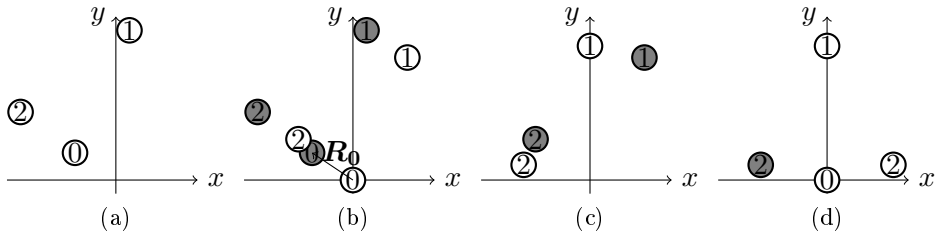


Figure 7.5: Adjusting Coordinates: a) Positions given by MDS; b) Positions after shifting node 0 to origin ; c) Positions after rotation of node 1 ; d) Position after flipping node 2 to the right plane

intermediate position T. Finally, we check if t_2 is on the right half-plane,i.e. if node 2 has a positive x-coordinate. If so, $S = T$, else we reflect T over the vertical axis as in Eq. 7.17.

$$(t_0, t_1, t_2, \ldots, t_{n-1}) = (s_0, r_1 - r_0, r_2 - r_0, \ldots, r_{n-1} - r_0) \times \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & cos(\alpha) \end{pmatrix} \tag{7.16}$$

$$S = \begin{cases} T, & \text{if } t_2 \text{ is in the right plane} \\ T \times \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, & \text{otherwise} \end{cases} \tag{7.17}$$

7.5 Implementation and Results

In Chapter 6 we showed the implementation and the results of our ranging algorithm. In that experiment we programmed 3 devices: Robot 1 and Robot 3 were stopped in each side of a mid-field and robot 2 was moved manually (remote control) to perform the trajectory colored in magenta in Fig. 6.2 .

We used Classic MDS algorithm on MATLAB with our collected data. Since we know Robot 1 and 3 positions, we can shift,rotate and flip the resulting map from MDS. In Figures 7.6,7.7,7.8,and 7.9 we plot only the first meters of robot 2 path, to better understand how MDS work, and how big is the error in the relative localization. For each figure we can see:

- Blue line: the real path previously covered by Robot 2

- Red and Green Circle: Respectively new estimated and ground truth position of Robot 1
- Red and Green dot: Respectively new estimated and ground truth position of Robot 3
- Red and Green x: Respectively new estimated and ground truth position of Robot 2

In Figure 7.6 we can see the the first part of the path covered by Robot 2. As we can see there are several estimated positions with x coordinate $= 0$. This happens because the two estimated distances $\delta_{2,1}$ and $\delta_{2,3}$ sometimes do not intersect, so no solution is given in 2D (only 1D),so x coordinate $= 0$.

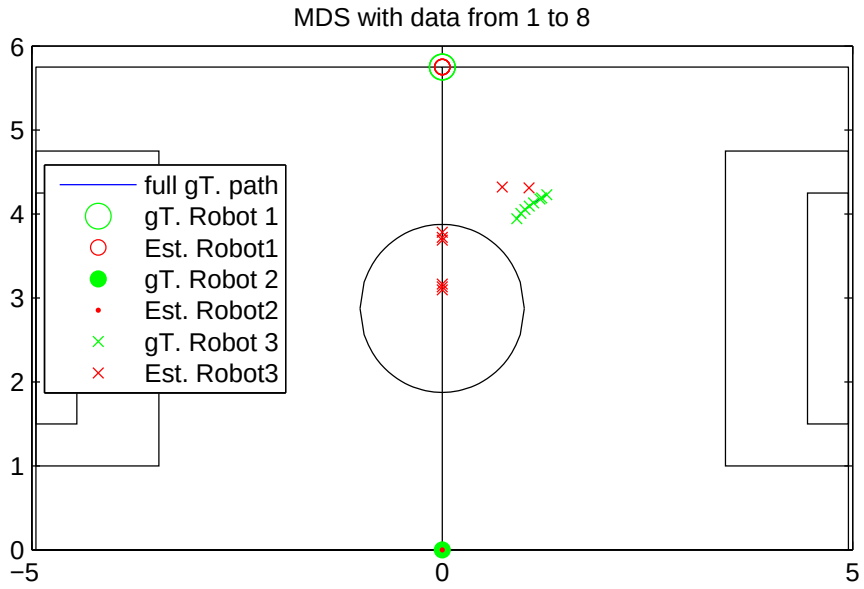


Figure 7.6: Estimated and real position of Robot 2 (red and green) at steps 1-8

In fig. 7.7 we can see the second part of the path: the estimated positions are near the real position but there is a visible error (about 1,2 meters). The same things happens In fig. 7.8, and fig. 7.9 where the error increases.

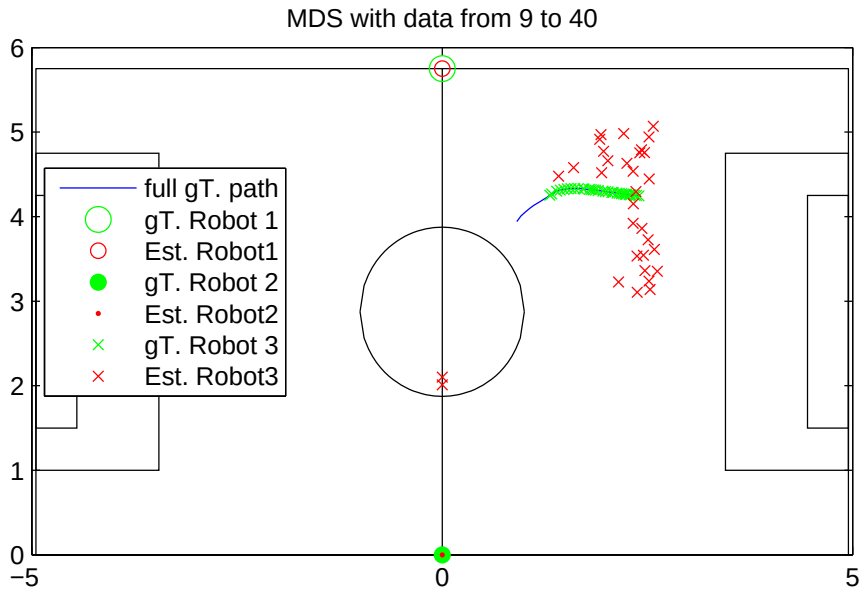


Figure 7.7: Estimated and real position of Robot 2 (red and green) at steps 9-40

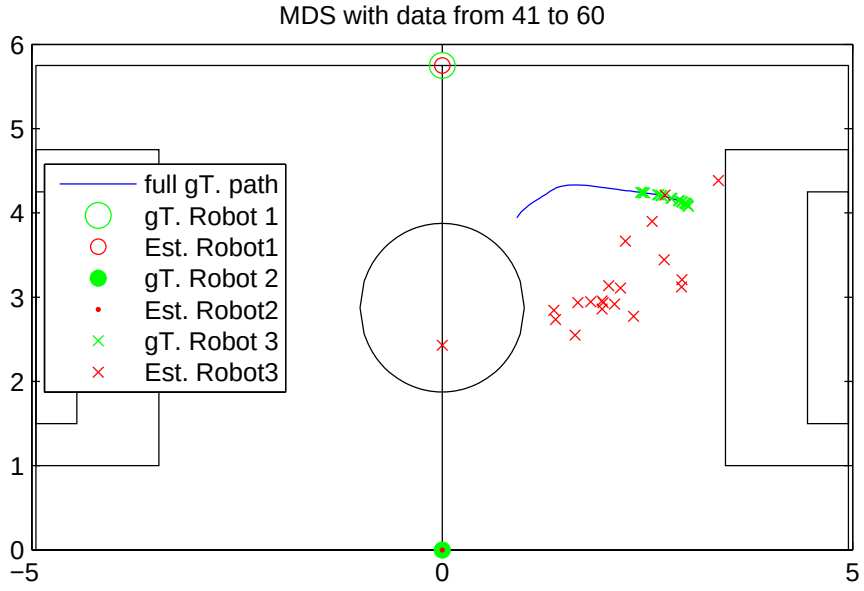


Figure 7.8: Estimated and real position of Robot 2 (red and green) at steps 41-60

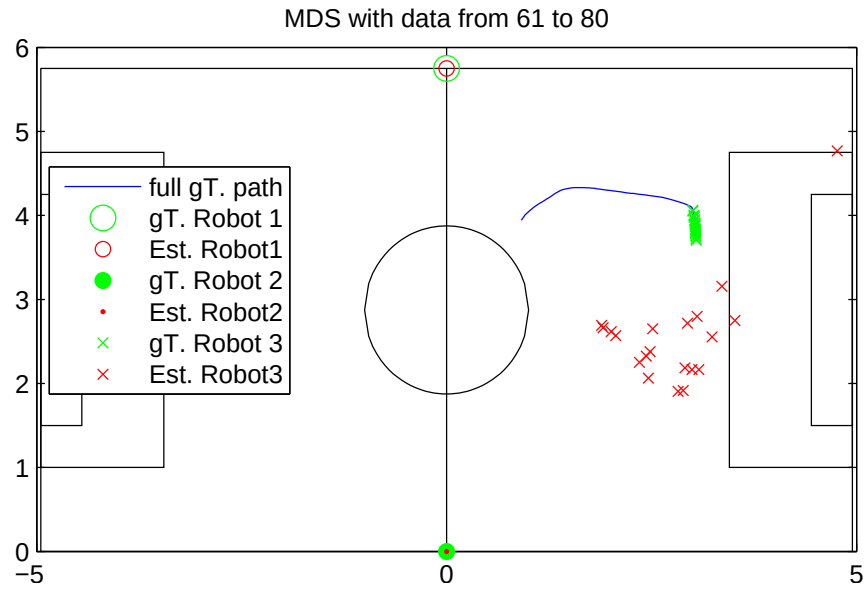


Figure 7.9: Estimated and real position of Robot 2 (red and green) at steps 61-80

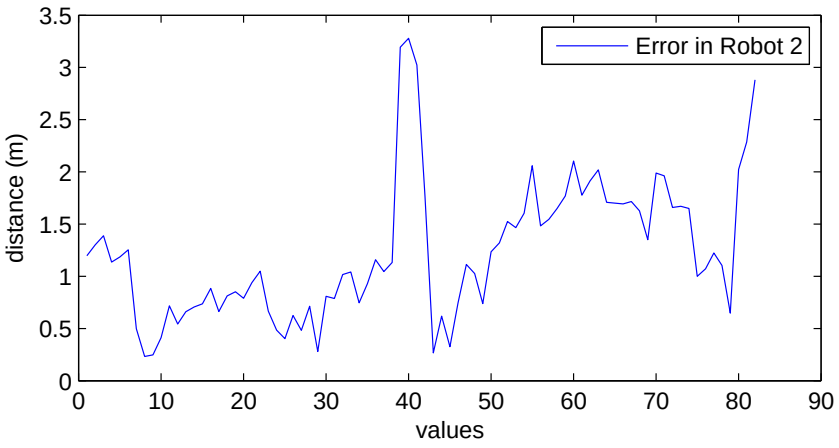


Figure 7.10: Error between Real and Estimated Distance in MDS

Finally fig. 7.10 shows the error between real and estimated distance of robot 2. The mean error is $\mu = 1.2321$ with a $std = 0.6662$, but there are some considerations to do:

1. MDS works better with a larger number of nodes. Here in our implementation we considered 3 nodes, only. With more nodes, MDS improves significantly and the error decreases.

2. EKF gives to us two fundamental values: The estimated distance and the covariance. With the covariance we have an important tool that helps understanding how much we can trust in our estimation. This leads us to be able to use Weighted MDS. However, In this particular case we cannot implement it because our configurations have two anchors nodes in a known position, and only one moving. The Distance Matrix and the Weight matrix should be:

$$D = \begin{bmatrix} 0 & \hat{\delta}_{1,2} & 5.75 \\ \hat{\delta}_{2,1} & 0 & \hat{\delta}_{2,3} \\ 5.75 & \hat{\delta}_{3,2} & 0 \end{bmatrix}, W = \begin{bmatrix} 0 & \sigma_{1,2} & 1 \\ \sigma_{2,1} & 0 & \sigma_{2,3} \\ 1 & \sigma_{3,2} & 0 \end{bmatrix} \tag{7.18}$$

where $\hat{\delta}_{1,2}, \sigma_{1,2}(\hat{\delta}_{2,3}, \sigma_{2,3})$ are the estimated distance and covariance from EKF filter between Robot 1 and Robot2 (Robot 3). In this case, the weighted MDS will return exactly the same result as classic MDS because two of three nodes have weight equal to 1.

Chapter 8

Conclusion and future work

In this work, we have successfully combined the ToF and RSSI ranging to perform an online estimation of the indoor log-distance path loss model, which together with an EKF was used to track distance between robots. Results show that by using our online estimator, we can approach the performance of a pre-calibrated channel model, with the advantage of supporting dynamic changes on the communication environment. Moreover, we show that it is possible to dramatically reduce the number of ToF ranges, with negligible accuracy loss. This reduction is only possible if the communication channel is stable for large periods of time, however, it translates in bandwidth gain. Some issues still remain open, specifically, the optimization of the time interval between ranges.

8.1 Future work

Since some issues still remain open, we can briefly describe some future works:

- Implementation with a team of robots (e.g. 5) in a dynamic environment: corridor, indoor, outdoor. This implementation will show the real capabilities of the developed algorithm because we will be able to show how it dynamically adapts to the rapid changes of the environment and because, as said before, EKF returns the covariance matrix that should be used in the Weighted MDS to improve the localization measurements.

- Implementation of a Kalman filter(or others) to reduce position error in MDS estimation. This type of filtering will reduce jumps in the estimation coordinates, creating a smoother trajectory of the robots.
- Optimize the time period of ToF-ranges. We will study how the accuracy decreases using longer time periods. In this way we will able to set the minimum time interval between ToF Ranges to have a certain QoS.

Appendix A

Middle-ware for teams of mobile robots

Different middle-ware layers have been developed to help the task of programming teams of autonomous agents, providing logical abstractions to support cooperation. Unfortunately, the actual use of communication and synchronization by the specific middle-ware layer may impose different delays and, in the end, may cause the middle-ware to fail supporting the requirements referred above.

Therefore, to support such requirements efficiently, a specific software infrastructure was developed for CAMBADA (Cooperative Autonomous Mobile Robots with Advanced Distribuited Architecture) middle-size robotic soccer team of the University of Aveiro, Portugal, which is composed by several components Two main components are: a middleware based on a Real-Time Database (RTDB) and a wireless communication protocol based on WiFi and implementing a Reconfigurable and Adaptive TDMA(RA-TDMA).

The following sections may be useful for understanding the middleware at the base of this dissertation and refer to the work done by Frederico Santos, Luis Almeida, Luis Seabra Lopes, and Pauolo Pedreias from University of Coimbra,and Aveiro, Portugal ([18],[17]).

A.1 The Real-Time Database

A replicated blackboard called Real-Time Database (RTDB) was developed, which holds the state data of each agent together with local images of the

state data shared by other team members. A specialized communication systems triggers the required transactions in the background at an adequate rate to guarantee the refresh of those local images. In the robotic soccer case, the information within the RTDB holds the absolute positions and postures of all team members, as well as the position of the ball, among other less relevant data. This approach allows a robot to easily use the other robots sensing capabilities to complement its own. For example, if a robot temporarily loses track of the ball it might use the position of the ball as detected by another robot. this is done without explicit use of communication, abstracting away the data distribution itself.

A.1.1 RTDB Implementation

The RTDB is fully implemented in *ANSIC* over several blocks of shared memory. One of the blocks is private area for local information, only, i.e., which is not to be disseminated to the others robots; and the other blocks (one corresponding to each team member) are the shared area with global information. One of the shared blocks is written by the agent itself (read-write), whose data is sent to the others and could also be used for interprocess communication, while the remaining blocks(read-only) are used to store the information received from the others agents. The allocation of shared memory is carried out by means of a specific function call, $DB_init()$, called once by every process that needs access to the RTDB. The memory allocation is executed by the first process to use such call, only. Subsequent calls just return the shared memory block handler and increment a process count. Conversely, the memoery space used by the RTDB is freed using the function call $DB_free()$ that decreases the process count and, when zero, release the shared memory. The RTDB is accessed concurrently by processes that capture and process images and implement complex behaviours, and by the periodic task that manages the communication with the other robots through the wireless interface. All processes access the RTDB, with local non-blocking function calls, $DB_put()$ and $DB_get()$ that allow writing and reading records, respectively. $DB_get()$ further requires the specification of the agent from which the item to be read belongs to, in order to identify the respective area in the database.

A.1.2 Internal Structure

The RTDB is organized in a set of records plus a set of associated data blocks. The records contain the fields referred in fig. namely an identifier, a pointer to the respective data block, the size of that block, a timestamp for computing the age of the data, the update period reflecting the dynamism of the respective item, and a control field for data consistency. To enforce data consistency during concurrent accesses a double data block is used for each record. With this scheme any write operation on that item is made on the block that is free at that instant. This method ensures consistent data retrieval, as long as there is only one process updating the same item.

A.2 Adaptive TDMA protocol

The basis of the communication protocol is a Time-Division Multiple-Access (TDMA). However, since the load in the network cannot be totally controlled by the team, the only alternative left is to adapt to the current channel conditions and reduce access collisions among team members. This is achieved using an adaptive TDMA transmission control as proposed in [22]. The TDMA round period is set off-line and called *team update period(T_{tup})*, setting the responsiveness and the temporal resolution of the global communication. It is, thus, an application requirement. T_{tup} is divided equally by the number of the team members generating the TDMA slot structure. With equal slots, if the agents transmit at the beginning of their slots, their transmission are separated as much as possible. The target inter-slot period can be computed as

$$T_{xwin} = \frac{T_{tup}}{N} \tag{A.1}$$

where N is the number of team agents. Normally each robot will only use a fraction of its slot and the unused part constitutes leeway to accommodate the uncontrolled load. When the respective TDMA slot comes, all currently scheduled transmissions are piggybacked on one single 802.11 frame and sent to the channel. The presence of uncontrolled load may lead to a delay (δ) of the packet reception. Each agent uses this delay to compute its next transmission instant, thus adapting the effective TDMA round period (Figure A.1).

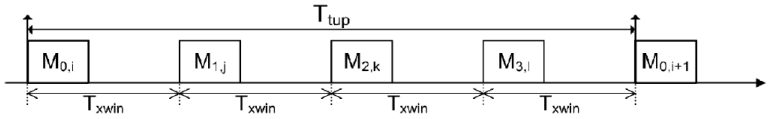


Figure A.1: Adaptive TDMA round

when a robot j transmits at time $t_{j,now}$ it sets its own transmission instant $t_{j,next} = t_{j,now} + T_{tup}$,i.e. one round after. However, it continues monitoring the arrival of the frames from the others robots. When the frame from robot i arrives, the delay δ_i of the effective reception instant with respect to the expected instant is calculated. If this delay is within a validity window $[0, \Delta]$, with Δ being a global configuration parameter, the next transmission instant is delayed according to the longest such delay among the frames received in one round (Fig. A.1),i.e.,

$$t_{j,next} = t_{j,now} + T_{tup} + max(\delta_i)_{i=0...N-1, i \neq j \wedge \delta_i \leq \Delta} \tag{A.2}$$

On the other hand, if the reception instant is outiside that validity window, then δ_i is set to 0 and does not contribute to update $t_{i,next}$.

The practical effect of the adaptation in the protocol is that the transmission instant of a frame in each round may be delayed up to Δ with respect to the predefined round period T_{tup}. Therefore, the effective round period will vary within $[T_{tup}, T_{tup} + \Delta]$.

A.3 Hardware: nanoLOC Development Kit 3.0

One of the proposal is to do not use any extra sensors except that the transceiver for the communications. For this reason we use the nanoLOC Development Kit 3.0. It is a complete, easy to use set of tools for evaluating, prototyping and developing applications based on the nanoLOC TRX Transceiver.

The nanoLOC TRX Transceiver is a highly integrated mixed signal chip offering robust wireless communication and ranging capabilities. It utilizes Chirp Spread Spectrum (CSS), a unique wireless communication technology developed by nanotron for the 2.4 GHz ISM band.

The nanoLOC Development Kit 3.0 is composed of 5 nanoLOC DK

boards. Each board, wich uses the nanoLOC TRX Transceiver and the ATmega128L microcontroller[1], is designed for location and ranging applications.

Figure A.2: NanoLOC Development Kit 3.0

[1] A low power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture with 128 Kb Flash and 4 Kb SRAM. This microcontroller drives the nanoLOC TRX Transceiver via the SPI interface. See also Atmel ATmega128L datasheet available from Atmel.

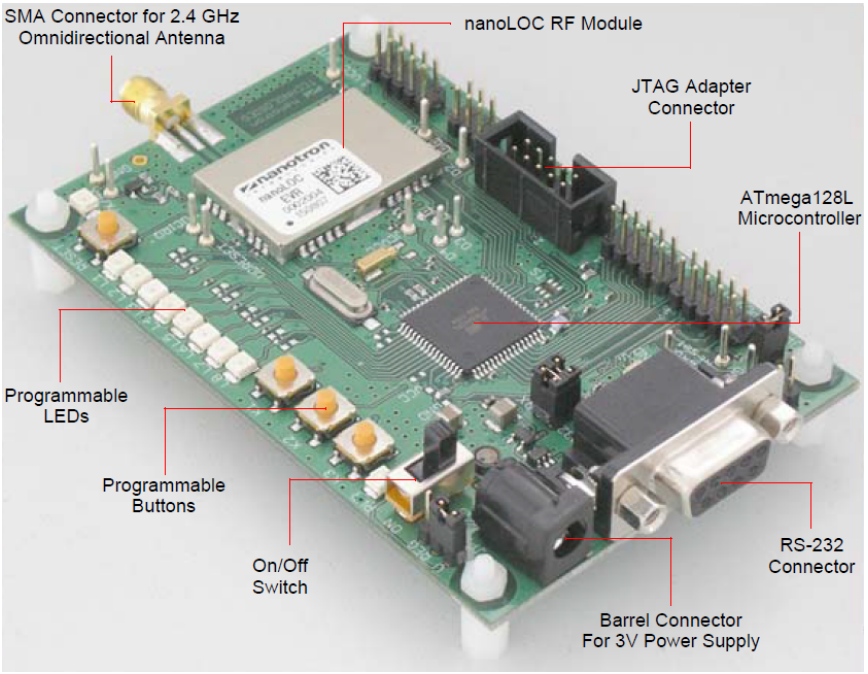


Figure A.3: NanoLOC Device

Appendix B

Matlab Code

B.1 Chatpter 4

B.1.1 Maximum likelihood estimator (MLE)

```
function [ X ] = modelMLE( distance, rssi )

notBadDistances=distance>0;
notBadRSSI=rssi~=-35;
notBadDistances=notBadDistances&notBadRSSI;

goodDistances=distance(notBadDistances);
goodRSSI=rssi(notBadDistances);

A = [ ones(length(goodDistances(:)),1), -10*log10(goodDistances(:))];
b = goodRSSI(:);

X = pinv(A'*A)*A'*b;

end
```

B.1.2 Our Model estimator: 1point-MLE

```
function [ X ] = ourModelEstimator(X,measuredD,measuredRSSI)
%%
n = 10;
realdist = logspace(0,1,n)';
```

```
A = [ ones(n,1), -10*log10(realdist)];
b = X(1) - 10 *X(2)* log10(realdist);

A = [ A ; [ 1 -10*log10(measuredD)]];
b = [ b ; measuredRSSI];

X = pinv(A'*A)*A'*b;
end
```

B.2 Chapter 5

B.2.1 EKF implementation

```
function [ X, S ] = KF_predict( X, S, A, R )

    X = A*X;
    S = A*S*A' + R;

end
```

```
function [ X,S ] = KF_measure_distance( X,S,Q,measured_distance, measured_rssi,PLM, bias

    H = [1 0 ; -10*PLM(2)/(X(1)*log(10)) 0];
    z = [measured_distance ; measured_rssi];

    K = S*H'*pinv(H*S*H' + Q);
    X = X + K*(z - hd_r(X,bias,PLM(1),PLM(2)));
    S = (eye(2) - K*H)*S;

end
```

```
function [ X,S ] = KF_measure_RSSI( X,S,Q, measured_rssi,PLM, bias )

    H = [-10*PLM(2)/(X(1)*log(10)) 0];
    z = [measured_rssi];

    K = S*H'*pinv(H*S*H' + Q);
    X = X + K*(z - hr(X,bias,PLM(1),PLM(2)));
    S = (eye(2) - K*H)*S;
end
```

```
function [hs]  = hd_r(s,b,ro0,alpha)

    hs = [s(1) - b(1) ; ro0 - b(2) - 10*alpha*log10(s(1))];
end


function [hs]  = hr(s,b,ro0,alpha)

    hs = [ ro0 - b(2) - 10*alpha*log10(s(1))];
end
```

B.2.2 UKF implementation

```
function [x,P]=ukf(fstate,x,P,hmeas,z,Q,R)
% UKF   Unscented Kalman Filter for nonlinear dynamic systems
% [x, P] = ukf(f,x,P,h,z,Q,R) returns state estimate, x and state covariance, P
% for nonlinear dynamic system (for simplicity, noises are assumed as additive):
%           x_k+1 = f(x_k) + w_k
%           z_k   = h(x_k) + v_k
% where w ~ N(0,Q) meaning w is gaussian noise with covariance Q
%       v ~ N(0,R) meaning v is gaussian noise with covariance R
% Inputs:   f: function handle for f(x)
%           x: "a priori" state estimate
%           P: "a priori" estimated state covariance
%           h: fanction handle for h(x)
%           z: current measurement
%           Q: process noise covariance
%           R: measurement noise covariance
% Output:   x: "a posteriori" state estimate
%           P: "a posteriori" state covariance
%
L=numel(x);                                 %numer of states
m=numel(z);                                 %numer of measurements
alpha=1e-3;                                 %default, tunable
ki=0;                                       %default, tunable
beta=2;                                     %default, tunable
lambda=alpha^2*(L+ki)-L;                    %scaling factor
c=L+lambda;                                 %scaling factor
Wm=[lambda/c 0.5/c+zeros(1,2*L)];           %weights for means
Wc=Wm;
Wc(1)=Wc(1)+(1-alpha^2+beta);               %weights for covariance
c=sqrt(c);
X=sigmas(x,P,c);                            %sigma points around x
```

```
[x1,X1,P1,X2]=ut(fstate,X,Wm,Wc,L,Q);          %unscented transformation of process

if(x1(1)<0)
    x1(2)=-x1(2);
    x1(1)=-x1(1);
end

for ii=1:length(X1)
    if(X1(1,ii)<0)
        X1(2,ii)=-X1(2,ii);
        X1(1,ii)=-X1(1,ii);
    end
end

[z1,Z1,P2,Z2]=ut(hmeas,X1,Wm,Wc,m,R);      %unscented transformation of measurments
P12=X2*diag(Wc)*Z2';                        %transformed cross-covariance
K=P12*pinv(P2);
x=x1+K*(z-z1);                              %state update
P=P1-K*P12';                                %covariance update

function [y,Y,P,Y1]=ut(f,X,Wm,Wc,n,R)
%Unscented Transformation
%Input:
%        f: nonlinear map
%        X: sigma points
%       Wm: weights for mean
%       Wc: weights for covraiance
%        n: numer of outputs of f
%        R: additive covariance
%Output:
%        y: transformed mean
%        Y: transformed smapling points
%        P: transformed covariance
%       Y1: transformed deviations
L=size(X,2);
y=zeros(n,1);
Y=zeros(n,L);
for k=1:L
    Y(:,k)=f(X(:,k));
    y=y+Wm(k)*Y(:,k);
end
Y1=Y-y(:,ones(1,L));
P=Y1*diag(Wc)*Y1'+R;
```

```
function X=sigmas(x,P,c)
%Sigma points around reference point
%Inputs:
%       x: reference point
%       P: covariance
%       c: coefficient
%Output:
%       X: Sigma points
A = c*chol(P)';
Y = x(:,ones(1,numel(x)));

X = [x Y+A Y-A];
```

B.2.3 Window Median Filter implementation

```
function [window]=window_push(window,sample,size)
window = [sample;window];
window = window(1: size);


function [ val ] = window_pop( window,ignore,type )
f=find(window~=ignore);
if(isempty(f))
    val=0;
    return
end
switch(type)
    case 'mean'
        val=mean(window(f));

    case 'medi'
        val=median(window(f));
end
end
```

List of acronyms

RF Radio-Frequency :

Although radio frequency is a rate of oscillation, the term "radio frequency" are also used as a synonym for radio,i.e., to describe the use of wireless communication

ToF Time-of-Flight :

describes a variety of methods that measure the time that it takes for an object, particle or acoustic, electromagnetic or other wave to travel a distance through a medium.

RSSI Received Signal Strength Indicator :

In telecommunications, Received Signal Strength Indicator is a measurement of the power present in a received radio signal.

UWB Ultra Wide Band :

is a radio technology which may be used at a very low energy level for short-range, high-bandwidth communications using a large portion of the radio spectrum.

ToA Time-of-arrival :

is the travel time of a radio signal from a single transmitter to a remote single receiver. ToA uses the absolute time of arrival at a certain base station

TDoA Time-Difference-of-Arrival :

uses measured time difference between departing from one and arriving at the other station.

U.S. ultrasound :

Ultrasound is an oscillating sound pressure wave with a frequency greater than the upper limit of the human hearing range.

RIPS Radio Interferometric Positioning System :

The Radio Interferometric Positioning System (RIPS) utilizes standard MICA2 motes for self localization. The technique relies on a pair of nodes emitting radio waves simultaneously at slightly different frequencies.

AoA Angle of arrival :

Angle of arrival measurement is a method for determining the direction of propagation of a radio-frequency wave incident on an antenna array.

MLE Maximum-Likelihood Estimator :

is a method of estimating the parameters of a statistical model. When applied to a data set and given a statistical model, maximum-likelihood estimation provides estimates for the model's parameters.

LSE Least Squares Estimator :

The method of least squares is a standard approach to the approximate solution of overdetermined systems, i.e., sets of equations in which there are more equations than unknowns. "Least squares" means that the overall solution minimizes the sum of the squares of the errors made in the results of every single equation.

Std. Standard Deviation :

In statistics and probability theory, the standard deviation (represented by the Greek letter sigma, σ) shows how much variation or dispersion from the average (mean, also called expected value) exists

KF Kalman Filter :

is an algorithm that uses a series of measurements observed over time, containing noise (random variations) and other inaccuracies, and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone.

EKF Extended Kalman Filter :

is the nonlinear version of the Kalman filter which linearizes about an estimate of the current mean and covariance.

UKF Unscented Kalman Filter :

A nonlinear Kalman filter which shows promise as an improvement over the EKF.

GRV Gaussian random variable :

A random variable with a Gaussian distribution

UT Unscented Transformation :

The Unscented Transform (or UT) is a mathematical function used to estimate the result of applying a given nonlinear transformation to a probability distribution that is characterized only in terms of a finite set of statistics.

MDS Multi-Dimensional Scaling :

is a modern technique for visualizing data in a multi-dimensional space. It is a means of visualizing the level of similarity of individual cases of a dataset. Used also in localization

Bibliography

[1] Traian E Abrudan, Azadeh Haghparast, and Visa Koivunen. Time synchronization and ranging in ofdm systems using time-reversal.

[2] Hongbin Li, Luis Almeida, Zhi Wang, and Youxian Sun. Relative positions within small teams of mobile units. In *Mobile Ad-Hoc and Sensor Networks*, pages 657–671. Springer, 2007.

[3] Luis Oliveira, Hongbin Li, Luis Almeida, and Traian E Abrudan. Rssi-based relative localisation for mobile robots. *Ad Hoc Networks*, 2013.

[4] Traian E Abrudan, Luis M Paula, Joao Barros, Joao Paulo Silva Cunha, and NB Carvalho. Indoor location estimation and tracking in wireless sensor networks using a dual frequency approach. In *IEEE International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2011.

[5] Paolo Barsocchi, Stefano Lenzi, Stefano Chessa, and Francesco Furfari. Automatic virtual calibration of range-based indoor localization systems. *Wireless Communications and Mobile Computing*, 12(17):1546–1557, 2012.

[6] Mohamed Laaraiedh, Lei Yu, Stephane Avrillon, and Bernard Uguen. Comparison of hybrid localization schemes using rssi, toa, and tdoa. In *Wireless Conference 2011-Sustainable Wireless Technologies (European Wireless), 11th European*, pages 1–5. VDE, 2011.

[7] David Macii, Alessio Colombo, Paolo Pivato, and Daniele Fontanelli. A data fusion technique for wireless ranging performance improvement. 2013.

[8] Marco Avvenuti. Localization in wsn. Pervasive and Sensor Network Systems, 2012.

[9] In Jae Myung. Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, 47(1):90–100, 2003.

[10] Sebastian Thrun, Wolfram Burgard, Dieter Fox, et al. *Probabilistic robotics*, volume 1. MIT press Cambridge, 2005.

[11] James V Candy. *Bayesian signal processing: Classical, modern and particle filtering methods*, volume 54. John Wiley & Sons, 2011.

[12] Irene Markelic and William Paley. Kalman filter tutorial.

[13] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. IEEE, 2000.

[14] Ingwer Borg. *Modern multidimensional scaling: Theory and applications*. Springer, 2005.

[15] Duke Lee. *Localization using multidimensional scaling (LMDS)*. PhD thesis, UNIVERSITY OF CALIFORNIA, 2005.

[16] Luis Oliveira, Luis Almeida, and Frederico Santos. A loose synchronisation protocol for managing rf ranging in mobile ad-hoc networks. In *RoboCup 2011: Robot Soccer World Cup XV*, pages 574–585. Springer, 2012.

[17] Frederico Santos, Luís Almeida, and Luís Seabra Lopes. Self-configuration of an adaptive tdma wireless communication protocol for teams of mobile robots. In *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, pages 1197–1204. IEEE, 2008.

[18] Frederico Santos, Luis Almeida, Paulo Pedreiras, and Luis Seabra Lopes. A real-time distributed software infrastructure for cooperating mobile autonomous robots. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1–6. IEEE, 2009.

[19] Luis Oliveira, Carmelo Di Franco, Traian E. Abrudan, and Luis Almeida. Rf-based ranging for mobile robots: Using time-of-flight and rssi for channel estimation. 2nd PhD. Studends conference in electrical and computer engineering (StudECE 2013), 2013.

[20] Luis Oliveira, Carmelo Di Franco, Traian E. Abrudan, and Luis Almeida. Fusing time-of-flight and received signal strength for adaptive radio-frequency rangin. In *ICAR 2013. to appear*. IEEE, 2003.

[21] Nanotron. nanoloc development kit. `http://www.nanotron.com/EN/PR_nl_dev_kit.php`, 2010.

[22] Frederico Santos, Luís Almeida, Paulo Pedreiras, L Seabra Lopes, and Tullio Facchinetti. An adaptive tdma protocol for soft real-time wireless communication among mobile autonomous agents. In *Proc. of the Int. Workshop on Architecture for Cooperative Embedded Real-Time Systems, WACERTS*, volume 2004, pages 657–665, 2004.