POLITECNICO DI MILANO

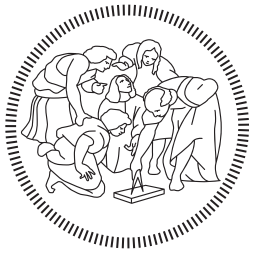
Facoltà di Ingegneria

Scuola di Ingegneria Industriale e dell'Informazione

Dipartimento di Elettronica, Informazione e Bioingegneria

Master of Science in

Computer Science and Engineering



Realization and performance evaluation of a hybrid UWB/WiFi indoor localization system

Supervisor:

PROF. MATTEO CESANA

Assistant Supervisor:

ING. RAZVAN PITIC

Master Graduation Thesis by:

MARCO IENI

Student Id n. 863752

Academic Year 2017-2018

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AOA	Angle of Arrival
AP	Access point
DOA	Direction of Arrival
DW	DecaWave
IPS	Indoor positioning system
LOS	Line-of-sight
NLOS	Non-line-of-sight
RSP	Received Signal Phase
RSSI	Received signal strength indication
RSS	Received signal strength
RTLS	Real Time Location Systems
RTOF	Round-trip time-of-flight
SNR	Signal-to-noise ratio
TDOA	Time difference of arrival
TOA	Time of arrival
TOF	Time of flight
TS	Taylor-series

TWR Two way ranging

UWB Ultra-wideband

WSN Wireless sensor network

ABSTRACT

Indoor localization is a very important theme in the industry 4.0 era. This thesis presents some of the main problems and solutions that regards this topic, by also analyzing hybrid approaches combining different techniques. Besides the generic overview of the topic and the presentation of the most commons techniques and the current literature review, in this thesis there also results and methodologies of a practical work with the goal of analyzing the utilization of Ultra-wideband (UWB) and WiFi wireless protocols, together with localization techniques Received signal strength (RSS) and two-way ranging (TWR), which is based on Time of flight (TOF).

The practical work is divided in two main parts: the first one consists in setting up a system that is able to collect data by exploiting both technologies and using it in some experiments that reproduce both Line-of-sight (LOS) and Non-line-of-sight (NLOS) conditions; the second one is about the analysis of the data that had been collected, in which Received signal strength indication (RSSI) WiFi is predicted through the data relative to UWB in order to study the relation between the two protocols, RSSI values obtained with both UWB and WiFi are compared in a visual way and localization is performed by using information of both protocols. The devices that were chosen in order to analyze the performances of UWB and WiFi protocols are respectively the DW1000 Decawave chip and some simple Raspberry Pi 3 model B.

The results obtained at the end of this work show that by combining the data that were collected during the measurements process with the coordinates estimated by the DecaWave system with Machine Learning techniques it is possible to improve the precision of the Decawave system itself, up to double it in some scenarios.

SOMMARIO

La localizzazione in ambienti interni è una tematica molto importante nell'era dell'industria 4.0. Questa tesi presenta alcuni dei principali problemi e soluzioni che caratterizzano questo argomento, analizzandone anche approcci ibridi che combinano più tecniche tra di loro. La panoramica generale di questa tematica, la presentazione delle tecniche di uso più comune e l'analisi dello stato attuale della letteratura sono accompagnate da un lavoro pratico che ha l'obiettivo di analizzare l'utilizzo dei protocolli wireless Ultra-wideband (UWB) e WiFi, affiancati alle tecniche di localizzazione Received signal strength (RSS) e two-way ranging (TWR), la quale è basata su Time of flight (TOF).

Il lavoro pratico è diviso in due parti principali: la prima consiste nella messa in piedi di un sistema che possa raccogliere dati sfruttando entrambe le tecnologie e nell'impiego di quest'ultimo in esperimenti che riproducono sia condizioni di linea di vista che non; la seconda consiste nell'analisi dei dati raccolti, in cui viene predetto il Received signal strength indication (RSSI) del WiFi mediante i dati relativi ad UWB in modo da studiare la relazione tra i due protocolli, vengono confrontati i valori di RSSI ottenuti sia con UWB che con WiFi in maniera visiva e viene effettuata la localizzazione usando le informazioni di entrambi i protocolli. I dispositivi scelti per analizzare le prestazioni dei protocolli UWB e WiFi sono rispettivamente il chip DW1000 della DecaWave e dei semplici Raspberry Pi 3 model B.

I risultati ottenuti alla fine di questo lavoro mostrano che combinando i dati raccolti durante il processo di misurazioni con le coordinate stimate dal sistema DecaWave mediante techiche di Machine Learning è possibile migliorare la precisione del sistema DecaWave stesso, fino anche a raddoppiarla in alcuni scenari.

1 INTRODUCTION

Indoor positioning nowadays is a very important service. Products stored in a warehouse, police dogs trained to find explosives in a building or an elderly patient inside a retirement home are just few examples of entities that is extremely important to localize with high precision.

Positioning systems may be applied in different contexts, like for example indoor navigation, position tracking, route tracking, entertainment, autonomous navigation, recommender systems, smart buildings, etc. For instance, for what regards indoor navigation and position tracking, if you take a cruise ship as an example, you may want to let the passengers know the position of their kids, or guide them in order to show the way to their point of interest, like their cabin or the gym. Regarding route tracking, a use case may be tracking the position of visitors inside a museum, in order to know which are the most crowded areas or which display cabinets are less interesting and reorganize the exposure logic. Autonomous navigation, instead, consists in knowing your position inside the environment in order to find the right route to reach a point of interest. For example, a robot could use this in order to arrive at a specific position.

Global navigation satellite systems (GPS or GNSS) are generally not suitable for indoor localization use cases, since microwaves are easily attenuated and scattered by roofs, walls and other objects. To this extent, other localization techniques are nowadays available for the specific case of indoor positioning.

The main purpose of this thesis is to analyze the performances of different hybrid localization approaches, which combine in a smart way some of the most relevant indoor localization techniques.

For this purpose, we developed a testbed for hybrid indoor localization, which is made of a DecaWave TREK1000 evaluation kit, which utilizes Ultra-wideband (UWB) as wireless protocol and two-way ranging (TWR) as localization technique, and four Raspberry Pi 3 model B, which enable to perform localization with WiFi and RSS. The DecaWave kit is composed of four EVB1000 boards and we configured three of them as anchors and one of them as tag. Similarly, we configured three Raspberry Pi's as WiFi access points and the remaining one in a way that constantly scans wireless networks in order to measures the RSSI from the three WiFi access points. We carried out the measurements at a laboratory with different wide areas, in which we were able to reproduce both Line-of-sight (LOS) and Non-line-of-sight (NLOS) conditions.

After the measurements process, we evaluated different localization algorithms by exploiting the data collected with the previously discussed testbed.

In Section 2 we show the main characteristics of some of these techniques, also providing a general background on localization fundamentals. After that, we discuss about some wireless protocols that are commonly used in indoor positioning and we present some localization algorithms, some of which are hybrid, i.e. they combine different techniques with each other in order to achieve better performances.

In Section 3 we explain the details of the measurement process, therefore we give an overview of Raspberry Pi 3 model B and DecaWave TREK1000 and we show how we configured them in order to achieve our goals and the steps of the data acquisition phase.

In Section 4 we analyze the results we obtained in order to compare the different technologies that we used and to see their statistical relations. Finally, we perform localization with the data that we obtained, increasing the performances of the starting localization system.

2 LITERATURE REVIEW

In this section we summarise the main literature for what regards wireless indoor positioning solutions, we give an overview of the wireless protocols that we used in the measurement process and we present some localization algorithms, some of which are hybrid. In fact, since we are going to perform localization by exploiting both Round-trip time-of-flight (RTOF) and Received signal strength (RSS), we will do an overview of some localization techniques that combines the two.

The following sections are strongly based on "Survey of wireless indoor positioning techniques and systems" [1], which is a fundamental review for the indoor localization field.

2.1 LOCALIZATION OVERVIEW

Localization is the process to determine the location of an entity, like for instance an object or a human. However, location information may be of different types:

PHYSICAL LOCATION It is expressed in the form of coordinates, which identify a point on a 2-D or 3-D map.

SYMBOLIC LOCATION It is expressed as a logic location that makes sense for the user, such as "office", "third-floor", etc.

ABSOLUTE LOCATION It is expressed in the form of coordinates, but uses a shared reference grid for all located objects.

RELATIVE LOCATION It is based on the proximity to known reference points or base stations.

Furthermore localization may be either active or passive:

PASSIVE LOCALIZATION The entity that you want to localize has no devices on it, therefore it does not have the ability to send or receive signals. It is less intrusive than active localization, but due to its nature it is less precise. Passive Localization systems usually leverages static devices called anchor nodes which are placed at a priori known position and periodically send signals to each other. Basing on the received signals they estimate how the environment is composed. However, since you have nothing that differentiates the subjects between them, you cannot say which subject you are localizing.

ACTIVE LOCALIZATION The entity that you want to localize has a device on it, called tag, that has the ability to send/receive signals to/from the anchors. Localization is performed by analyzing these signals. Since every tag may have a unique ID, an important advantage of active localization is that you are able to identify the entities that you are localizing. Furthermore, because of the presence of an active object, this type of localization offers better localization performances with respect to passive localization, but it has some limitations. In particular, it cannot always be used, because in some cases the entity that you want to localize is not in the condition of keep a piece of hardware with it. In the following, when not specified differently, we will always refer to active localization.

An Indoor positioning system (IPS) is a system that locates objects or people inside a building by using radio waves, magnetic fields, acoustic signals, or other sensory information collected by mobile devices [2]. It consists in at least two separate hardware components: the tag, which is mobile, and the anchor, which is fixed.

Usually the tag is a simple and inexpensive device, attached to the entity that you want to localize. The anchor, instead, is what in most of the use cases carries the major part of the system "intelligence", since it determines the distance between itself and the tag.

In positioning systems different system topologies are possible:

REMOTE POSITIONING SYSTEM The tag, acting as signal transmitter, sends the signal, while the anchor, acting as measuring unit, receives the tag signal. The results from all the measuring units are sent to a remote side (like for instance a master station), which finally estimates the tag location.

SELF-POSITIONING The anchor, acting as signal transmitter, sends the signal, while the tag, acting as measuring unit, receives the anchor signal. Since in this configuration the tag knows the position of the anchor, it is able to compute its location according to the measured signals.

INDIRECT SELF-POSITIONING It is like remote positioning system topology, but the measurement result is sent from the remote side to the tag.

INDIRECT REMOTE POSITIONING It is like self-positioning topology, but the measurement result is sent from the tag to the remote side.

2.2 TRIANGULATION

Triangulation is one of the most used location positioning algorithms for IPSs and it is based on the geometric properties of triangles. It has two derivations: lateration and angulation.

2.2.1 *Lateration Techniques*

Lateration estimates the position of an object by measuring its distances from multiple reference points. The intuition behind lateration is that in order to compute the position of the tag, you draw a circle around each anchor with diameter equal to the estimated distance between that anchor and the tag. The position of the tag is ideally given by the intersection of all the circles.

In order to enable 2-D positioning, the distances should be estimated using signals that arrive from at least three anchors. Furthermore, if you assume that the tag is always above or under the plane that intersects the three anchors, you can also tell the Z-axis value.

In the following we present several lateration techniques.

2.2.1.1 *TOA*

Time of arrival (TOA) or Time of flight (TOF) is the travel time of a radio signal from a single transmitter to a remote single receiver.

The TOA technique (as well as TDOA and RTOF) is based on the fact that the distance between the tag and the anchor is directly proportional to the propagation time. Ultra-wideband (UWB) is an example of signaling technique that can be used to measure TOA.

TOA technique calculates the distance between tag and anchor by measuring the one-way propagation time and multiplying it by the speed of light, which is the velocity at which the signal is assumed to travel. This approach presents two problems related to the implementation phase:

- all the clocks of transmitters and receivers in the system have to be precisely synchronized;
- the transmitting signal must be labeled with a timestamp in order for the measuring unit to discern the distance the signal has traveled.

Regarding performances, instead, one of the main drawbacks of the TOA method is its accuracy degradation in NLOS conditions. The multipath signal that comes from different paths, in fact, introduces an extra time delay that adds errors to the range estimation [3].

2.2.1.2 *TDOA*

With the Time difference of arrival (TDOA) technique you are able to determine the position of the tag by examining the difference in time at which the signal arrives at multiple anchors, rather than the absolute arrival time of TOA at a certain base station.

With this approach, only the anchors have to be synchronized with each other, while no requirement on the tag is imposed, because the timing error is cancelled (or at least strongly reduced) in the time difference operation.

This advantage makes TDOA more preferable with respect to TOA in remote positioning system topologies, because with TDOA you do not need to implement complex synchronization mechanisms in the tags and therefore you are able to get a more scalable system. Furthermore, with TDOA the timing error related to NLOS conditions that is present in the TOA technique is cancelled or at least reduced in the time difference operation [3].

2.2.1.3 *RSS*

Received signal strength (RSS) technique consists in estimating the distance between the tag and the anchor, by using the attenuation of the emitted signal strength. In particular, the difference between the strengths of the transmitted and the received signals are translated into a range estimate.

When there is a strong enough multipath effect, time and angle of an arrival signal suffer from it and consequently also the accuracy of estimated location. In scenarios like this one, RSS may give better performances. In this regard, in Section 2.4.2.1 it is presented a hybrid TDOA/RSS Algorithm that uses the estimated ranges obtained with TDOA when the received signal strength exceeds a preset certain threshold, while it uses the estimated ranges obtained with RSS, when the received signal strength is below the threshold.

Another way to try to reduce multipath effect is using fingerprinting. With this method you train the system beforehand, by storing all the RSSI measurements at different positions in the environment where you want to localize. Then, at the moment of localization, you use the RSSI vector measured by the anchors to access the database that you have created before to obtain the estimated position. , that may be due to However, this method suffers environment changes, because it is based on the data collected beforehand and so it does not consider some changes in the furniture or just people walking around during the localization phase.

The main advantages of RSS are its easiness in adoption and the low technological cost, while its main drawback is that it is not robust to noise when the distance between transmitter and receiver is high. In fact, the power drops with distance, while noise usually

not. Therefore, localization precision decreases when you have a low Signal-to-noise ratio (SNR).

2.2.1.4 *RTOF*

Round-trip time-of-flight (RTOF) technique (sometimes called simply TOF) measures the RTOF, i.e. the time-of-flight of the signal traveling from the tag to the anchor and back. The range measurement mechanism is the same as that of the TOA, but a more moderate relative clock synchronization requirement replaces the more challenging one of TOA. Therefore this technique partially solves the synchronization problem that was mentioned in Section 2.2.1.1.

The way it works is similar to the way bats sense the environment around them: the measuring unit sends a message to the other device and the latter replies with another message. In this way the measuring unit is able to compute the complete round-trip propagation time.

RTOF grants good performances, but not as good as if the anchors where synchronized, since it only partially compensates the phase differences. Furthermore, this method has some drawbacks, since it is difficult for the measuring unit to know the delay/processing time of the responder. In long-range or medium-range systems you have long transmission time and so this delay could be ignored if it is small enough, but for short range systems it must be taken into account.

A known solution to this drawback, that for instance has been implemented in DW1000 chip, is the so-called delayed transmission procedure. This mechanism consists in aligning the start of radio packet's transmission to a pre-defined timestamp of the internal clock. In this way, the responder does not answer to the message as soon as it processes it, but only at a moment which is known a priori by both the measuring unit and the responder itself. Therefore, the variable processing time of the responder is substituted with a fixed delay known in advance [4].

In Section 3.1.1.1 we will deepen a variant of RTOF technique, which is the one used in DecaWave TREK1000, the kit that was used during the measurement process.

2.2.1.5 *RSP*

The Received Signal Phase (RSP) technique uses the carrier phase (or phase difference) to estimate the range. In fact, if all the transmitting stations are synchronized and emit pure sinusoidal signals of the same frequency with no phase offset, you can tell how far you are from each transmitting station based on the phase of the received signal.

Let's call f the frequency of the sinusoidal signals and ϕ the phase of the received signal. In Equation 2.1 is reported the expression of the signal.

$$S(t) = \sin(2\pi f t + \phi) \tag{2.1}$$

As we said, the delay is represented by the phase of signal $S(t)$, therefore it may be expressed as in Equation 2.2, in which D is the range estimation and c is the speed of light.

$$\phi = \frac{2\pi f D}{c} \tag{2.2}$$

Finally, the range estimation D from each transmitting station may be calculated with Formula 2.3.

$$D = \frac{c\phi}{2\pi f} \tag{2.3}$$

Of course, since the the signal is sinusoidal, in practice you are able to distinguish values of ϕ that goes from 0 to 2π, therefore the transmitted signal's wavelength should not be longer than the distance between the transmitting station and the receiver. You may also use RSP in combination with TOA, TDOA or RSS in order to fine-tune the location positioning.

2.2.2 *Angulation Techniques*

The angulation derivation of triangulation consists in estimating the position by computing angles relative to multiple reference points.

Angle of Arrival (AOA) or Direction of Arrival (DOA) technique consists in finding the angles between an anchor reference direction and the tag. The location of the tag is ideally given by the intersections of all the direction lines of the angles that you found. AOA may be implemented with directional antennae or with an array of antennae.

One of the main advantages is that in order to do 2-D or 3-D positioning you need one anchor less with respect to lateration techniques. In fact, for 2-D case you only need two anchors, while for the 3-D case you need three. In general, this technique produces good results even by using the minimum number of receivers only. Furthermore, the clocks of the anchors do not need to be synchronized.

The main disadvantage, instead, is that the hardware needed to be placed at each receiver in order to implement this technique is relatively expansive. Another problem is that in presence of multipath effect the quality of the position degrades, because the estimated angle is not very accurate.

2.3 WIRELESS PROTOCOLS

In this section we are going to give an overview of the two wireless protocols that we used in the measurement process, i.e. WiFi and UWB. This section is strongly based on "A Comparative Study of

Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi" [5], a paper that provides an overview of these communication standards.

2.3.1 *UWB*

Ultra-wideband (UWB) corresponds to the IEEE 802.15.3 standard and it is an indoor short-range high-speed wireless communication method. The systems based on the UWB radio signals are among the most promising ones and are nowadays vastly becoming more and more popular. The frequency band of UWB goes from 3.1 up to 10.6 GHz, the channel bandwidth range is from 500 MHz to 7.5 GHz, while its bandwidth is over 110 Mbps (up to 480 Mbps), which can easily satisfy most of the multimedia applications such as audio and video delivery in home networking.

Unlike the classic spread spectrum technique, UWB has the advantage to transmit in a manner that does not interfere with conventional narrowband and carrier wave transmission in the same frequency band. One of the most important differences between conventional radio transmissions and UWB, in fact, is that the first ones transmit information by varying power level, frequency, and/or phase of a sinusoidal wave, while UWB transmissions transmit information by generating radio energy at different time intervals and occupying a large bandwidth. In particular, in UWB systems the data are carried by the bursts of pulses in the range of nanoseconds or subnanoseconds, which means spreading the spectrum of the signal over the band of hundreds of MHz. Such short pulses lead to two important advantages of UWB:

- TOA of the received radio signal may be estimated with subnanosecond accuracy, which means a ranging error in the order of centimeters [6].
- The radio signal is more robust against the negative effects of multipath propagation, which are an important characteristic for the indoor environment [4]. In fact, since each pulse occupies the entire UWB bandwidth, some of the frequencies will have a line-of-sight trajectory [7].

Therefore, in the context of UWB networks, ranging achieves good precision with techniques based on timestamps (like for instance TOA or TDOA) because the used bandwidth is beyond 500 MHz. In general, the observed TOA or TDOA error can be drastically reduced as the bandwidth increases beyond a certain value [8].

However, it is not true that UWB does not come with some challenging drawbacks. The first one is that due to its high bandwidth and the restrictions for the maximum transmit power spectral densities prescribed by the authorities, the communication range is relatively short, which means that several anchor nodes have to be installed

to cover a normal building. The second drawback is related to the previous one, because since you may need an high number of devices, the minimization of costs and energy consumption is crucial in order to make UWB systems economically feasible. Finally, a problem which is essential for localization, is detecting and handling non line-of-sight radio signal propagation. In this regard, in paper "Real-time identification of NLOS range measurements for enhanced UWB localization" [9] is proposed a simple but very efficient method to distinguish between NLOS and LOS conditions that does not need the training data or a prior knowledge about the environment and thus it enables realtime NLOS identification.

2.3.2 *WiFi*

WiFi is a technology for wireless local area networking (WLAN) with devices based on the IEEE 802.11 standards, which supports station mobility transparently to upper layers. It allows users to surf the Internet at broadband speeds when connected to an access point (AP) or in ad hoc mode. The frequency bands are 2.4, 3.6, 5 and 60 GHz with a stream data rate that arrives up to 20 Gbit/s with an indoor range of 10 meters for the 802.11.ay, that will be released in 2019, and an indoor range that arrives up to 70 meters with a stream data rate up to 288.8 Mbps for the 802.11.n, that was released in 2009.

This technology suffers from the so called WiFi pollution, which consists in an excessive number of access points in the area, that can heavily decrease SNR. Additionally, other devices use the common 2.4 GHz band, like microwave ovens, security cameras, ZigBee and Bluetooth devices, baby monitors and so on. In order to try to solve this problem, WiFi uses dynamic frequency selection and transmission power control.

2.4 LOCALIZATION ALGORITHMS

In this section we present some localization algorithms, that exploit the protocols and the techniques showed previously in order to perform localization in an efficient way.

Some of these localization algorithms are hybrid, i.e. they combine different techniques with each other in order to achieve better performances.

2.4.1 *Single-Sided Asymmetric TWR*

In Section 2.2.1.4 we explain the RTOF technique, which some authors indicate as the basic version of Two-Way-Ranging (TWR), while in Section 3.1.1.1 we give an overview of the TWR algorithm used in TREK1000 kit, which is sometimes called Symmetric Double-Sided

TWR, that aims to reduce the effect of the clock mismatch between the tag and the anchor.

In this section, instead, we show another variation of TWR that is presented in paper "On the selection of protocol and parameters for UWB-based wireless indoors localization" [4]. This variation is called Single-Sided Asymmetric TWR and in Figure 2.1 we report a sequence diagram that shows its basic functioning. Single-Sided Asymmetric TWR is particularly useful in a scenario in which the following conditions hold:

- the tag is the device that needs to know the distances between itself and the anchors;
- the tag is the device that initializes the localization procedure;
- you do not have the necessity of having simple tags.

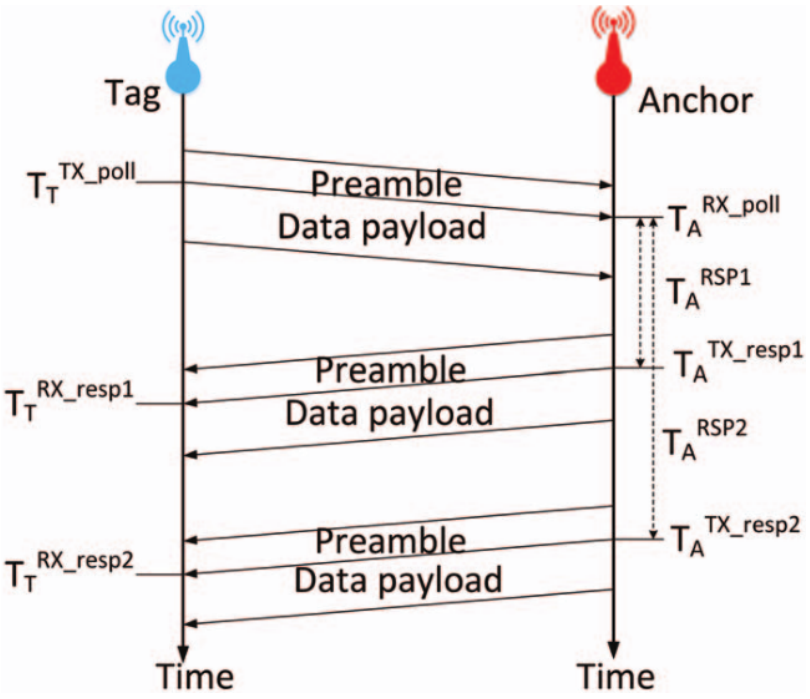


Figure 2.1: Single-sided asymmetric TWR procedure.

In scenarios like this, in fact, if you want to use Symmetric Double-Sided TWR, you are in the case of indirect self-positioning system topology, because the remote side is the one that estimates the location of the tag and then it has to send the result to the latter in order for the tag to know its position. If you want to keep all the complexity on the remote side, this approach may be good, but if the tag is already a complex object, like a robot for example, it has some computational power that can be exploited in order to compute the location on the tag itself and keep the remote side simpler. In this way, you switch from the indirect self-positioning system topology

to the self-positioning system one. Therefore, everytime you perform localization, you do not have to send the forth message, i.e. the report one, which is used by the remote side to communicate the position to the tag in the indirect self-positioning system topology.

Similar to the hypothesis done in paper "Ranging with ultrawide bandwidth signals in multipath environments" [10], we assume that the time period T is seen by anchor (A) and tag (T) nodes as

$$T = T_T(1 + k_T) + \mu_T = T_A(1 + k_A) + \mu_A \tag{2.4}$$

where k_T and k_A represents the difference between the "ideal" clock and the real one, respectively for the tag and the anchor, while μ_T and μ_A are the initial clock offsets of tag and anchor, respectively. However, when measuring a period of time, the clock offsets μ_T and μ_A may not be considered, provided that you do not take into account the quantization error.

In Single-Sided Asymmetric TWR, the TOF is estimated as

$$\begin{aligned} TOF_{est} = (&T_T^{RX\_resp1} - T_T^{TX\_poll} - (T_T^{RX\_resp2} - T_T^{RX\_resp1}) \\ &(T_A^{RSP1}/(T_A^{RSP2} - T_A^{RSP1})))/2 \end{aligned} \tag{2.5}$$

which is equivalent to

$$TOF_{est} = \frac{TOF}{1 + k_T} + 0.5\frac{\mu'_A - \mu'_T}{1 + k_T} \tag{2.6}$$

and the respective error is

$$\Delta TOF_{est} = \frac{k_T TOF}{1 + k_T} - 0.5\frac{\mu'_A - \mu'_T}{1 + k_T}. \tag{2.7}$$

2.4.2 *Hybrid localization*

As seen in a lot of practical experiments, the positioning precision and accuracy is improved by using the hybrid TDOA/RSS algorithm as compared to the TDOA only algorithm [3]. However, other types of hybrid localization techniques were proposed in literature too. For example, in paper "Hybrid RSS/AOA emitter location estimation based on least squares and maximum likelihood criteria" [11] was proposed a hybrid algorithm based on the combination of RSS and AOA. However, if you want to keep your system scalable enough, you should consider simple schemes with low complexity and cost, which is the case of RSS and TDOA.

2.4.2.1 *Hybrid positioning technique: TDOA/RSS*

In this section we give an overview of the hybrid TDOA/RSS solution that improves the position accuracy in indoor environment presented in paper "Accurate wireless indoor position estimation by using hybrid TDOA/RSS algorithm" [3].

The actors are the mobile station (MS), that acts as a receiver, and the base station (BS), that acts as a transmitter. The MS constantly monitors the received signal strength from the BSs, while the receiver always estimate the ranges from all the base stations by using both TDOA and RSS. So, every range estimation will be characterized by two values: one derived with TDOA and one with RSS. In order to choose which range to use you use the following selection rule:

- if the received signal strength exceeds a preset certain threshold, you select the estimated ranges using TDOA;
- if the received signal strength is below a preset certain threshold, you select the estimated ranges using RSS.

This selection rule leverages the fact that if the RSSI is high, then there are good LOS conditions between the BS and the MS and therefore a valid SNR as well as a good ranging accuracy is guaranteed. If the RSSI is low, instead, you are in NLOS conditions and so it is probable that RSS will give better precision and accuracy, since a time based measurements would suffer more from the presence of obstacles and high noise with respect to this technique.

2.4.2.2 *Hybrid RSS/TDOA localization utilizing WiFi networks*

In this section we present a hybrid RSS/TDOA wireless sensors localization algorithm that improves position accuracy in indoor environments by leveraging WiFi networks, which is described in paper "An Improved Hybrid RSS/TDOA Wireless Sensors Localization Technique Utilizing Wi-Fi Networks" [12]. The algorithm simultaneously exploits RSS from WiFi access points and both TDOA and RSS from the anchors of the system.

The main target of the algorithm are the scenerios in which a Wireless sensor network (WSN) is involved and it is shown that RSS information obtained from a WiFi network can be incorporated to produce better estimations through a cooperative operation inside the network. The tags, in fact, are assumed to be able to communicate with not only the anchors but also with other tags.

The presented hybrid algorithm is based on the networking structure depicted in Figure 2.2, which is composed of three types of nodes:

ANCHORS They are capable of processing RSS and TDOA measurements of the signals exchanged with the tags.

WIFI ACCESS POINTS They exchange RSS measurements with the tags. They are very common in today's wireless communications infrastructures, so it is probable that they are already present in the environment where you want to perform localization.

TAGS They exchange RSS and TDOA measurements with the anchors and RSS measurements with WiFi Access points.

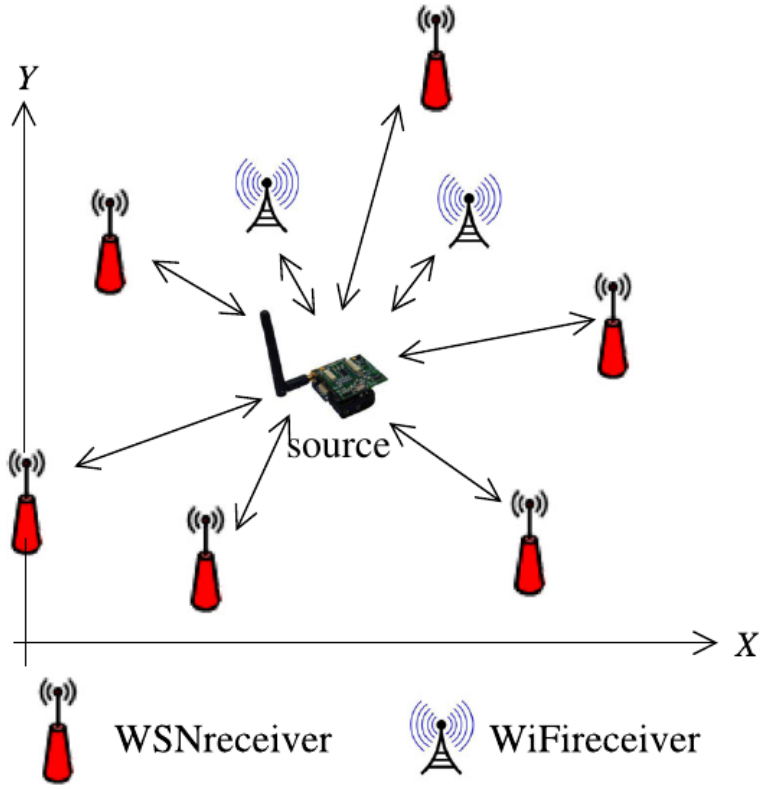


Figure 2.2: Networking structure of hybrid RSS/TDOA localization utilizing WiFi networks.

The collected RSS and TDOA data must be translated into equations to be solved in order to estimate the position. However, for the considered hybrid scheme, the equations are non linear, with a non-trivial solution to find. In order to overcome this drawback, the set of RSS and TDOA non linear equations are tranformed into a set of linear equations by Taylor-series (TS) expansion. The result with this approach can achieve high accuracy, but you should be careful in selecting the initial guess, because otherwise the algorithm may converge to a local minimum or a saddle point and thus causing a not negligible estimation error. In order to solve the initialization problem, an initial guess can be obtained from the maximum-likelihood-solved RSS [13]. Therefore, by using TS, the algorithm will quickly converge to an estimation and therefore reduce the overall computational power, which is very important, especially in WSNs.

Furthermore, together with the TS solution, it is presented another estimator, that leverages maximum-likelihood (ML), which requires rearranging the RSS/TDOA equations as a linear function of source location, distance between source and reference receiver, and sum of squared of the source location.

Simulation results show that the proposed hybrid positioning approach outperforms the previously considered localization solutions in WSNs, thanks to the joint process of the received signals' power and time difference of arrival. In most of the cases these improvements comes almost for free if you consider that in the environment where you want to perform localization there are already several WiFi access points, since they are becoming increasingly prevalent.

3 TESTBED DESCRIPTION

For this work we used four Raspberry Pi 3 model B and a TREK1000 evaluation kit, which is composed of four EVB1000 boards. In the following, we will see in what each device consists and how it was configured to work properly for this series of experiments. After that, we will explain all the steps that goes from positioning the devices in the environment to filling the database.

3.1 DEVICES OVERVIEW

In this section we give an overview of the devices we used in the testbed, by showing their main characteristics and features. For what regards TREK1000, we will also show the basic functioning of the ranging algorithm that is present on the board.

3.1.1 *TREK1000*

TREK1000 is an IC Evaluation Kit for TWR Real Time Location Systems (RTLS) and it enables users to evaluate the performance of DecaWave's DW1000 Ultra-Wideband (UWB) IC in RTLS application use cases based on various topologies (combinations of anchors and tags) [14]. As we already mentioned, it is composed of four EVB1000, which are some evaluation boards incorporating DecaWave's DW1000 IEEE 802.15.4-2011 UWB compliant wireless transceiver IC, STM32F105 ARM Cortex M3 processor, USB interface, LCD display and off-board antenna [15]. They are shipped together with a two-way ranging application already installed that runs a ranging algorithm that we will explain later.

One of the main advantages of this product is that it is sold together with the source code of the software application that runs on it and therefore you are able to deeply personalize its behavior. Furthermore, they also give you a PC application software with its source code that is able to evaluate the tags position by connecting the computer with the anchor with ID 0.

In our opinion, this is one of the best ways to evaluate the performances of the DW1000, which is the single chip wireless transceiver based on ultra-wideband that you may want to integrate in your own real product if you want to develop a device that exploits DecaWave UWB localization technique. Some examples of commercial products that integrate these chip are the ones from Sewio and Pozyx.

The characteristics of this chip declared by DecaWave are the following:

- Allows the location of objects in real time location systems (RTLS) to a precision of 10 cm indoors;
- Supports 110 kbit/s, 850 kbit/s & 6.8 Mbit/s data rates;
- Communications range of up to 300 m;
- Short packet durations support high tag densities (up to 11,000 in a 20 m radius);
- Highly immune to multipath fading;
- 6 frequency bands supported with center frequencies from 3.5 GHz to 6.5 GHz;
- It is not possible to be in the TX and RX states simultaneously (the DW1000 is a half-duplex transceiver device);
- Low power consumption (Transmit mode from 31 mA, receive mode from 64 mA and up to 100 nA in deep sleep mode) [16].

The DW1000 utilizes TWR as ranging algorithm, which is based on the estimation of TOF, in fact one of the most important task of the chip is to measure the time period between transmitting a request and receiving the reply frame. Because of this, it requires really accurate mechanisms for accurate time stamping both the transmission and reception of the radio packets. In this respect, since the DW1000 UWB transceiver first generates a 125 MHz internal clock which is used for time stamping the received or transmitted frames, there is a special engine, which applies the adjustment procedures to improve the resolution to around 15.65 picoseconds. In theory, this enables to achieve the distance error of less than three centimeters [4].

3.1.1.1 *Ranging algorithm*

In this section we will show how the details of the ranging algorithm of the default TREK1000 application works. We haven't modified the basic behavior of this algorithm, so what is written in the following is also valid for our experiments. The only thing that we modified is that together with the distance between the tag and the anchors we also log RSSI and timestamp values.

The two-way ranging algorithm that DecaWave decided to adopt for this kit is described in Figure 3.1, where Device A represents the tag and Device B represents the anchor.

In the following we describe the algorithm, by showing its main steps:

1. The tag initializes the range measurement, by sending an initial poll message;

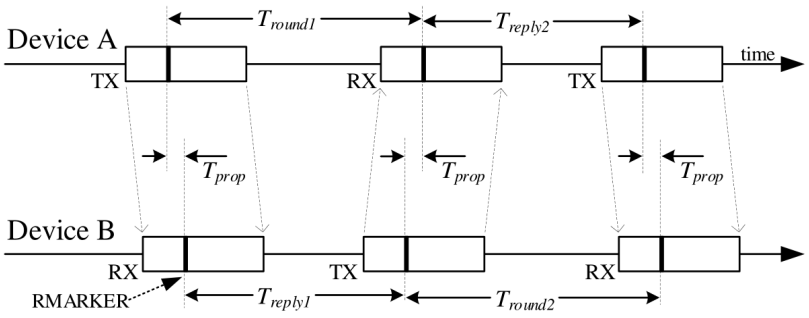


Figure 3.1: DecaWave Ranging algorithm between two devices.

2. The anchor always listen to its assigned channel, so if the tag and the anchor are connected to the same channel, the range between the two is low enough and the channel is not busy, then the anchor will receive the message;

3. The anchor replies back to the tag;

4. The tag receives the message and replies to the anchor with a final message;

5. The anchor receives the final message and evaluates the TOF between itself and the tag.

In Figure 3.2 it is shown the functioning of the algorithm for our scenario, i.e. with one tag and three anchors.

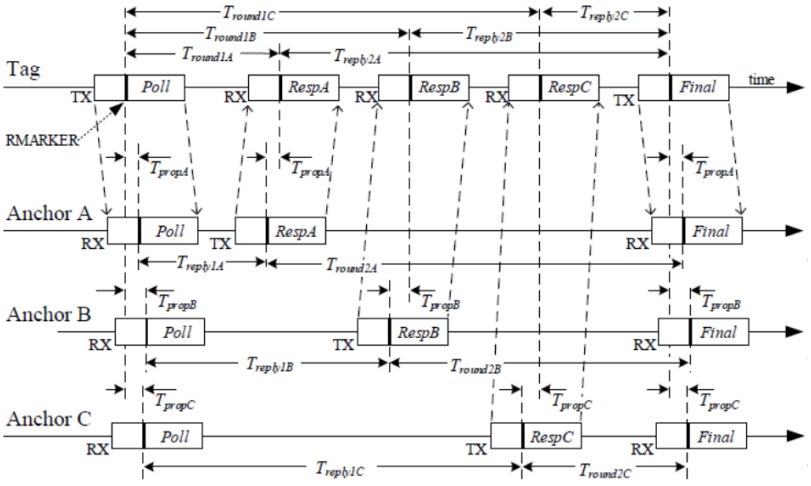


Figure 3.2: Ranging algorithm used by DecaWave system.

In order to evaluate the TOF, anchors and tags keep track of time intervals. The tag saves two time intervals for each anchor that answered with a Resp packet:

T_{ROUND1} Difference between the time at which the Poll message was sent and the time at which the Resp message was received;

T_{REPLY2} Difference between the time at which the Resp message was received and the time at which the Final message was sent.

The anchor saves two time intervals for each tag that transmitted a Poll message:

T_{REPLY1} Difference between the time at which the Poll message was received and the time at which the Resp message was sent;

T_{ROUND2} Difference between the time at which the Resp message was sent and the time at which the Final message was received.

The tag communicates T_{round1} and T_{reply2} to the anchors via the final message and finally, the anchors have all the data needed to evaluate the TOF [17]. Each anchor calculates the TOF between itself and the tag by using the following formula:

$$TOF = \frac{T_{round1} * T_{round2} - T_{reply1} * T_{reply2}}{T_{round1} + T_{round2} + T_{reply1} + T_{reply2}} \tag{3.1}$$

After that, the anchor sends the TOF to the tag, that at this point knows the ranges between itself and each anchor. However, as we already said before, every anchor always listens to the channel, therefore it can read the message that the other anchors send to the tag. This means that every anchor knowns the TOF between the tag and every other anchor. This is the reason why if you connect with the anchor with ID 0 to your laptop, it is able to log all the distances.

3.1.2 *Raspberry Pi*

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries [18]. For this thesis we used the Raspberry Pi 3 model B, which is the board that replaced the Raspberry Pi 2 Model B in February 2016. It is more powerful than what is needed for this experiments since it features a Quad Core 1.2GHz Broadcom BCM2837 64bit CPU and 1GB of RAM. Furthermore, it has BCM43438 wireless LAN, that provides on-board 2.4 GHz WiFi 802.11n connectivity [19].

As operating system for our devices we choose the Foundation's official supported operating system: Raspbian, a free operating system based on Debian optimized for the Raspberry Pi hardware [20]. The version that we choose was the last one available, i.e. Debian Stretch with an handy desktop environment already installed.

3.2 DEVICES CONFIGURATION

In this section we will show how we configured the devices to work properly for this work.

We configured three EVB1000 as anchors and one as tag, while three Raspberry Pi's were configured as WiFi access points. Every anchor was connected to a Raspberry Pi access point, while the tag was connected to the Raspberry Pi that was not configured as access point. We connected every EVB1000 with a Raspberry Pi via UART, in order to access the data that they output and log them into a file. In this way we are able to log the RSSI received from the WiFi access points together with the info of the DecaWave tag.

The data that were available by default that we used are the distances between the tag and every anchor. We modified the source code of the EVB1000 application in order to obtain also the TOF between the tag and every anchor and the RSSI between the DecaWave tag and the current anchor. Distances and timestamps are information that all the anchors know about each other, but RSSI between the anchor and the tag is a value that is present only on the anchor itself, because this info is calculated starting from values that are not shared with the network, but that are saved only locally in some specific registers.

So, we understand that every device contains a log with some unique info, because the Raspberry Pi's connected to the anchors know the RSSI between the anchor and the tag, while the Raspberry Pi connected with the tag contains the RSSI value of the access points. Therefore, in order to obtain all the data, there is the need to parse all the four logs together.

3.2.1 *TREK1000*

We configured all the four EVB1000 with Channel 2 (3.993 GHz) and 110kbits/s data rate. You have to choose the channel and the data rate according to a tread-off in terms of range and power efficiency, but also to the law of your country (for what regards the channel). In particular, the range and the utilized power are directly proportional to the channel frequency and inversely proportional to the data rate.

As we said before, we had to change the source code of the predefined device application in order to log also TOF and RSSI together with the distance. By following the flow of how the distances are evaluated was not that hard to log the TOF, while getting the value of the RSSI required more effort. First of all, the RSSI is not measured, but estimated only. The approximation of the estimation follows the trend of the graph that you find in Figure 22 of the DW1000 User Manual [17]. However, for the values that we obtained in our experiments, this approximation was very close to the expected value and so this fact did not affected importantly our results.

In order to implement RSSI logging we applied the following formula, which is reported in Section 4.7.2 of the latter document.

$$RSSI = 10\log_{10}\left(\frac{C * 2^{17})}{N^2}\right) - 113.77\,dBm \tag{3.2}$$

Where:

C Channel Impulse Response Power value reported in the CIR_PWR field of Register file 0x12 (Rx Frame Quality Information);

N Preamble Accumulation Count value reported in the RXPACC field of Register file 0x10 (RX Frame Information Register).

So, in order to evaluate the RSSI, we logged C and N in every anchor together with timestamps and distances and after the measurement phase (at the moment of inserting the data in the database with a PC) we evaluate RSSI starting from these two values.

After that we modified the source code of the application we flashed it to each EVB1000 board by using a ST-LINK/V2, which is an in-circuit debugger/programmer for STM8 and STM32 [21].

3.2.2 *Raspberry Pi*

As we already said, three Raspberry Pi's were configured as Wifi access point, while the network configuration of the other one was left as default. In order to turn our Raspberry Pi into a wireless access point different steps were required, like for instance, installing hostapd and edit its config file located at `/etc/hostapd/hostapd.conf` in order to configure the WiFi SSID, indicate the location of this config file by editing the `DAEMON_CONF` field of the file `/etc/default/hostapd` and so on. In order to do this we followed the article "How to use your Raspberry Pi as a wireless access point" on the website "www.thepi.io" [22].

In order to take the logs from the EVB1000 and to measure the WiFi RSSI, we created a relatively small python project, which was divided into the following modules:

PARSE Processes the Wifi scanning command output and returns a vector of three elements which are the WiFi RSSI values of the three Raspberry Pi's configured as wireless access point.

NETWORK_HANDLER Implements the class NetworkHandler, which is in charge of scanning the network in order to read the RSSI values of the Raspberry Pi's configured as access points. It calls the linux command `iwlist wlan0 scan` and calls the parse module in order to parse its output. Furthermore, it manages the case in which one of the expected access point is not present and it also offers the possibility to log WiFi RSSI for testing purposes.

```
1 def log_serial_and_wifi(self):
2     with serial.Serial(self.device, self.baud_rate, timeout=1) as
          ser:
3         while True:
4             line = ser.readline().rstrip()
5             rssi = self.nh.get_rssi_string()
6             self.data_logger.info(line + " " + rssi)
```

Listing 3.1: Logging of UWB and WiFi data.

UART Implements the class UartHandler, which let's you read values from the uart interface by specifying the device and the baud_rate. In the Raspberry Pi connected with the tag, everytime you receive something from the serial port, you call the network_handler module to read the WiFi RSSI values, in order to log the latter data and the DecaWave ones together in the same line.

LOG_HELPER Is just a utility that is used by the other modules to simplify the logging logic. It creates the log file and returns the logger object associated to it taking as input the logger name, its logging level and the format string of the log.

In Listing 3.1 we only report the very simple `log_serial_and_wifi(self)` function form the UartHandler, which is the function called by the tag in order to log both the information from the EVB1000 that are read from the UART interface and the WiFi RSSI values, that are read from the `wlan0` interface.

The only function parameter is "self", which in python indicates that the function is a method of the class and therefore the object itself is passed to the function, in order to access to its attributes.

Then, in line 2 we try to open a connection with the serial port using device and baud_rate that were defined in the constructor and with a default timeout of 1 second.

In line 3 an infinite loop starts. However, the CPU will not be in a busy-waiting state, because the function at line 4 is a blocking one, because it waits for the data to be ready on the serial port.

When the data is ready, line 5 is executed, where `get_rssi_string()` function of the NetworkHandler class (which the instance is saved as the attribute `self.nh`) returns a string that represents the current WiFi RSSI vector.

Finally, in line 6 we log the two strings that we build before with the data logger, that takes care of adding the current timestamp as prefix.

We set all the Raspberry Pi's to run the python script at startup, in order to avoid to launch it by hand and to automate as much as possible the measurement process.

```
18-05-03 14:47:43,102 6,12,0.35
18-05-03 14:48:30,323 5,12,0.35
18-05-03 14:49:09,129 5,11,0.35
```

Figure 3.3: Extract of the log that was took on the PC during the measurement process.

3.3 MEASUREMENT PROCESS

We carried out the measurements at the IoT Lab (Laboratorio di ricerca applicata del Politecnico di Milano) [23], where there are different wide areas both with LOS and NLOS conditions.

In order to get the real position of the tag, we divided the areas that we exploited for the experiments into a grid with squares of 1 meter x 1 meter. We fixed an origin in the grid, and so a correspondence between the floor of the lab and a Cartesian plane was established.

The anchors were placed on some plastic poles, at a distance greater then 15 cm from any wall, as specified in the TREK1000 user manual [24]. Also, we tried to keep the wires as far away as possible from the antennas.

We placed the anchors at fixed a priori known positions and heights and so their coordinates are known. Since the LOS and NLOS conditions where obtained in two different areas, the anchors were placed in two different positions, one for each case. However, the relative position between the anchors in the two cases were preserved, i.e. we ideally translated the anchors all together. We report the position of the anchors in the 3-D space in Table 3.1.

dataset	anchor 1	anchor 2	anchor 3
LOS	(5,0,1.5)	(1,6,1.2)	(6,12,1.5)
NLOS	(9,-1,1.5)	(5,5,1.2)	(10,11,1.5)

Table 3.1: Position of the anchors in the 3-D space.

The tag, on the contrary of the anchors that are kept fixed, was moved from a point to the other of the grid, in order to log all the data at different positions. We put it on top of a box 35 centimeters high.

Since we need to know in which position the tag was in every moment, before moving the tag to the next point, its real position was always logged in a PC. In particular, the tag was kept at a fixed position for 10 seconds before and after its real position was recorded in the laptop.

In order to explain better this concept, in Figure 3.3 we report an extract of the log that we took on the PC.

```
1 18-05-03 14:53:03,453 5353,7152,14374 17496,23631,48029
      -40,-46,-59 -81.694,-82.303,-89.604
2 18-05-03 14:53:04,145 5353,7199,14421 17496,23788,48186
      -43,-47,-65 -84.027,-82.333,-89.555
3 18-05-03 14:53:05,528 5357,7175,14445 17543,23709,48264
      -40,-47,-60 -81.518,-81.084,-89.488
```

Figure 3.4: Exctract of the final aggregated log.

For example, from the first row we know that at 14:47:43 and 102 milliseconds of day 2018-05-03, the tag was positioned at the point with coordinates (6,12,0.35). On the tag side, we have a log with day, time and all the other data that we track.

Therefore, by exploiting timestamps, we are able to associate the data that the tag logs to its current real position.

3.4 DATA STORING

During the measuring process we collected four logs: three from the anchors and one from the tag.

We developed a small python project with the goal of parsing the logs and aggregating their data all together into a single log. Furthermore, there is a module which is in charge of getting the RSSI value from C and N parameters, by applying formula 3.2. In Figure 3.4 we present an extract of the final aggregated log:

If we split each row with a space we obtain the following elements in order:

1. date when the data had been taken;
2. time when the data had been taken;
3. distances in millimeters between the tag and respectively anchor 1, 2 and 3;
4. TOF in picoseconds between the tag and respectively anchor 1, 2 and 3;
5. WiFi RSSI between the Raspberry Pi connected to the tag and the ones connected respectively to anchor 1, 2 and 3;
6. RSSI between the tag and respectively anchor 1, 2 and 3.

At this point, we give this file as input to a modified version of the C++ DecaWave localization software. The changes that we applied to the source code regards mainly bypassing the GUI, and taking as input the log file and the anchors positions.

This program scans the log line by line. For each line it reads the distance vector, it estimates the location of the tag according to it and appends the coordinates of the location at the end of the line.

```
1 18-05-03 13:55:36,725 9397,5498,3886 31221,18013,12535
      -57,-51,-57 -90.537,-88.932,-81.605 9.040,8.088,-0.888
2 18-05-03 13:55:37,414 9397,5484,4126 31221,17966,13333
      -55,-51,-58 -90.633,-88.940,-80.997 8.934,8.017,-1.143
3 18-05-03 13:55:45,974 8211,5924,4280 27199,19468,13881
      -59,-50,-49 -89.211,-85.313,-80.736 10.492,6.963,0.165
```

Figure 3.5: Exctract of the log with estimated position.

In Figure 3.5 we report an extract of the log that the program outputs:

The format of this log is the same as before, except that if we split each row with a space, the seventh element is the estimated position of the tag expressed as x,y,z coordinates.

However, not all the entries of this log contain useful information. Some of them, in fact, are not valid, because they may be relative to a moment in which we were moving the tag or the anchors for instance. Furthermore, as we said before, we have the necessity to know where each tag was at every time, in order to be able to put in relation the data that we collected with the real position of the tag.

For all of these reasons, we developed another small python project, which is in charge of scanning the log in which we specified the real position of the tag together with the current timestamp, and selecting only the valid rows from the log that the DecaWave localization software outputs. Furthermore, it inserts the data of every valid row that it finds in an sqlite3 database, together with the real position found. As we said before, the tag is kept at a fixed position for 10 seconds before and after its real position was recorded in the laptop, therefore an entry is considered valid only if the difference between the timestamp of the entry and one of the timestamps of the log with the real positions is less than 10 seconds. However, the clocks of the Raspberry Pi and the laptop are not synchronized, therefore the difference of time between the two was taken into account.

So, thanks to this programs we inserted the valid data of the log file in two different database: one for the LOS case and another one for the NLOS case.

In the following table we are going to see how we organized the data that we collected, i.e. the database structure.

Field	Type	Comment
id	INTEGER	Primary key, autoincrement.
insertion_timestamp	TEXT	When the data had been inserted in the db. It is set automatically to current timestamp.
collect_timestamp	TEXT	When the data had been collected.

Continued on next page

Continued from previous page

Field	Type	Comment
estimated_x	REAL	X coordinate estimated by the DecaWave system.
estimated_y	REAL	Y coordinate estimated by the DecaWave system.
estimated_z	REAL	Z coordinate estimated by the DecaWave system.
real_x	REAL	Actual x coordinate of the tag.
real_y	REAL	Actual y coordinate of the tag.
real_z	REAL	Actual z coordinate of the tag.
distance_a1	REAL	Distance between tag and anchor 1, measured by the DecaWave system.
distance_a2	REAL	Distance between tag and anchor 2, measured by the DecaWave system.
distance_a3	REAL	Distance between tag and anchor 3, measured by the DecaWave system.
time_a1	REAL	TOF between tag and anchor 1, measured by the DecaWave system.
time_a2	REAL	TOF between tag and anchor 2, measured by the DecaWave system.
time_a3	REAL	TOF between tag and anchor 3, measured by the DecaWave system.
dw_rss_a1	REAL	RSSI between tag and anchor 1, measured by the DecaWave system.
dw_rss_a2	REAL	RSSI between tag and anchor 2, measured by the DecaWave system.
dw_rss_a3	REAL	RSSI between tag and anchor 3, measured by the DecaWave system.
raspi_rss_a1	REAL	WiFi RSSI between tag and anchor 1, measured by the Raspberry Pi's.
raspi_rss_a2	REAL	WiFi RSSI between tag and anchor 2, measured by the Raspberry Pi's.

Continued on next page

Continued from previous page

Field	Type	Comment
raspi_rss_a3	REAL	WiFi RSSI between tag and anchor 3, measured by the Raspberry Pi's.

Table 3.2: Database structure.

4

DATA ANALYSIS

In this chapter we are going to analyze the data that we collected in order to see the performance difference between the DecaWave Real Time Location System and a simple one based on WiFi RSS performed with some Raspberry Pi's. We will also try to predict the values of Wifi RSSI starting from different fields of the database in order to study the relation between them. Furthermore, we will also see whether or not it is worth to combine these two systems and which performances we can achieve by exploiting WiFi RSSI only at the moment of real localization, with a previous data collection done with the DecaWave system too.

In the following, for the sake of simplicity, when we talk about "tag" and "anchor" we indicate both the Raspberry Pi's and the DecaWave device connected together. All the statistical computing and graphics presented in this section were obtained with the R programming language.

4.1 COLLECTED DATA

As we already said, we populated two databases: the first one in free space condition (LOS) and the second one in was in presence of obstacles (NLOS), specifically industrial machines, like an assembly line for instance.

In Table 4.1 we show the number of entries of the database and the mean and the standard deviation of the localization error (distance between the real position and the one estimated by the DecaWave system).

dataset	entries	error mean	error standard deviation
LOS	1267	0.176	0.112
NLOS	1190	0.379	0.307

Table 4.1: Localization error mean and standard deviation of test database.

In Figure 4.1 we present the normal distribution of localization error for both datasets.

The error that we refer to is the distance in meters between the real position given by real (x,y,z) coordinates vector and the one estimated by the DecaWave system, that is given by estimated (x,y,z) coordinates vector. These data had been taken from the original dataset where all the entries are present.

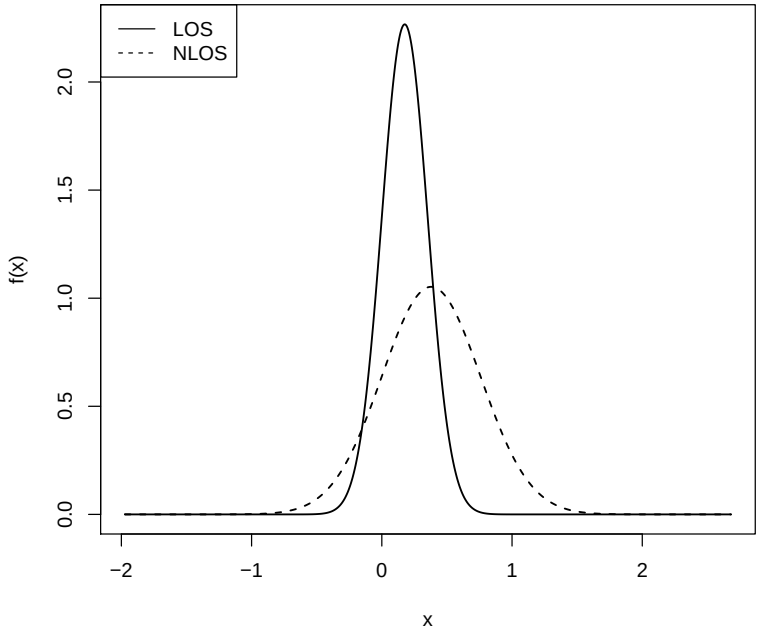


Figure 4.1: Localization error: Normal distribution of localization error for LOS and NLOS databases.

However, starting from this dataset, we build another one, where there is only one entry per each real (x,y,z) coordinates vector. In order to obtain this database, all the entries that had the same (`real_x, real_y, real_z`) were condensed in only one row, where all the other fields were calculated as the average of the values of all the entries with the same real coordinates.

The error mean and standard deviation were calculated for this database too, obtaining the results showed in Table 4.2:

dataset	entries	error mean	error standard deviation
LOS	78	0.162	0.076
NLOS	89	0.356	0.270

Table 4.2: Number of entries and location error mean and standard deviation of condensed database.

In Figure 4.2 we show the normal distribution of localization error for the condensed datasets.

4.2 DATA VISUALIZATION

In the following we are going to plot the RSSI measured from the DecaWave devices vs the RSSI measured from the Raspberry Pi's.

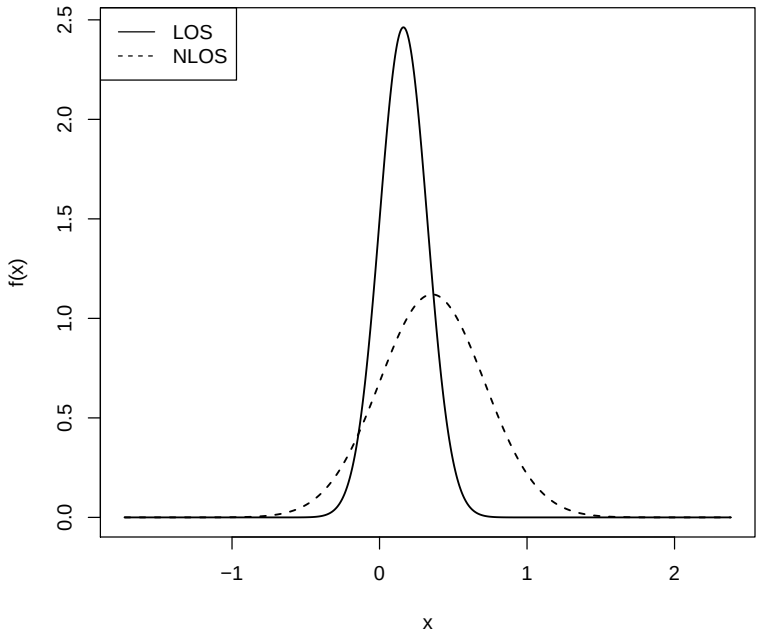


Figure 4.2: Localization error: Normal distribution of localization error for LOS and NLOS condensed databases.

Together with the points, we plot also the regression line and the LOWESS (LOcally WEighted Scatterplot Smoothing) line in order to compare the linear approximation with a trend that is closer to reality.

In Figures 4.3, 4.4 and 4.5 we show the plot derived from the data of the LOS original database for every anchor, while in Figure 4.6 we aggregate the points of all the anchors together. For anchors 1 and 2, the angular coefficient of the regression line is very close to zero, i.e. the correlation is low. Furthermore, if you compare the regression line with the LOWESS, you see that the linear trend is a good approximation. For anchor 3, instead, there is a worse linear approximation with respect to the other anchors, but correlation is much higher.

In Figures 4.7, 4.8 and 4.9 we show the plot derived from the data of the NLOS original database for every anchor, while in Figure 4.10 we aggregate the points of all the anchors together. In this case, the correlation is almost the same for all the anchors and the linear approximation is good in the range between -90 dBm and -83 dBm of the x-axis. Considering the data related to all the anchors, the correlation of the LOS original database is definitely higher with respect to the one of the NLOS original database.

In Figures 4.11, 4.12, 4.13 we show the plot derived from the data of the LOS condensed database for every anchor, while in Figure 4.14 we aggregate the points of all the anchors together. If you compare Figure 4.14 with Figure 4.6, you notice that in the LOS condensed database

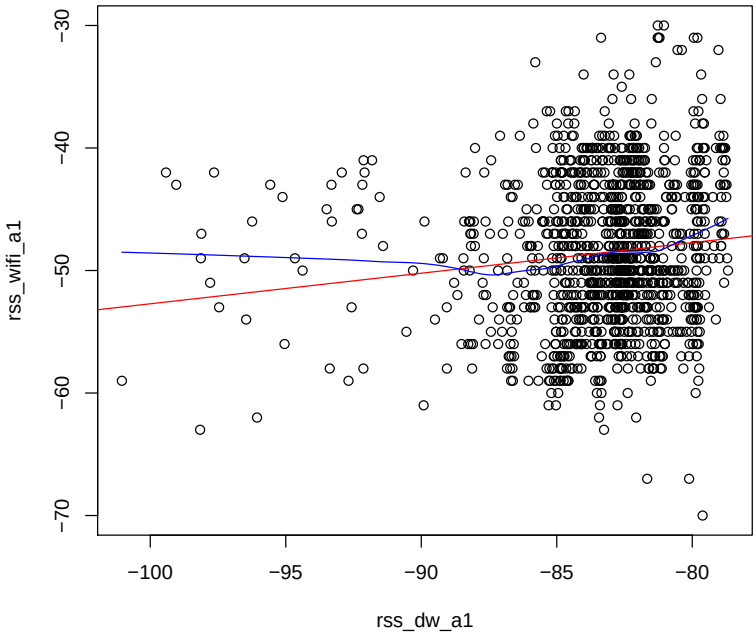


Figure 4.3: RSSI WiFi vs RSSI UWB of LOS original database for anchor 1.

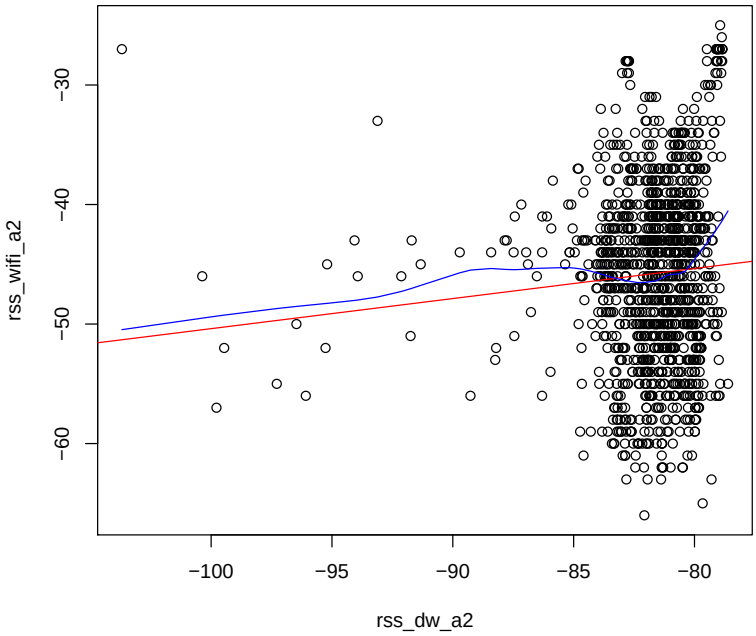


Figure 4.4: RSSI WiFi vs RSSI UWB of LOS original database for anchor 2.

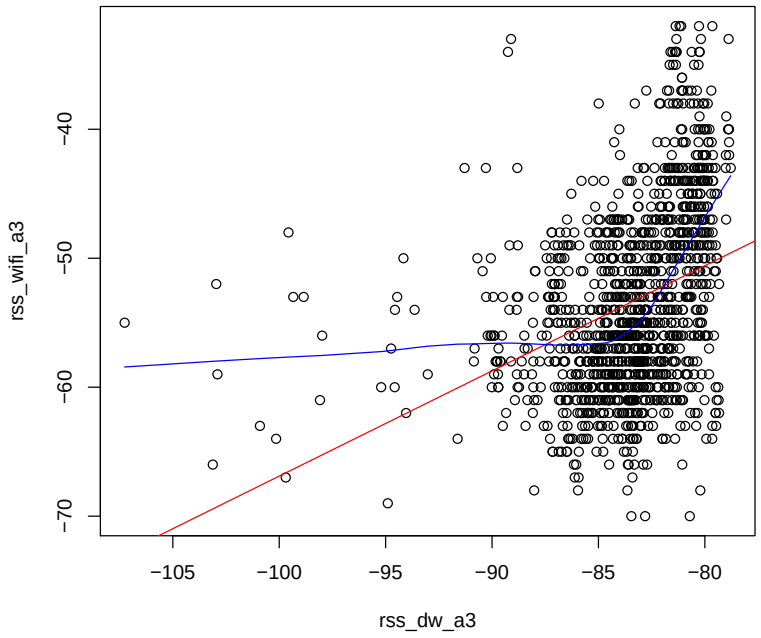


Figure 4.5: RSSI WiFi vs RSSI UWB of LOS original database for anchor 3.

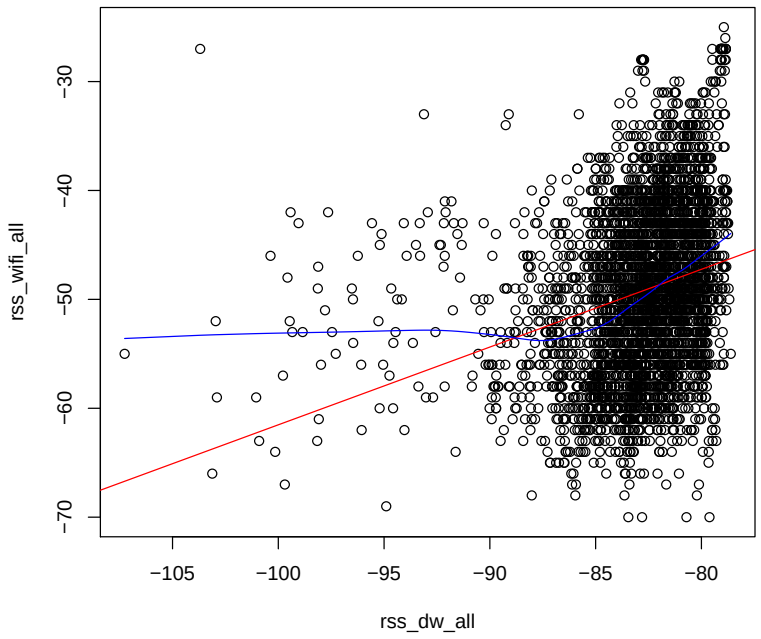


Figure 4.6: RSSI WiFi vs RSSI UWB of LOS original database for all anchors.

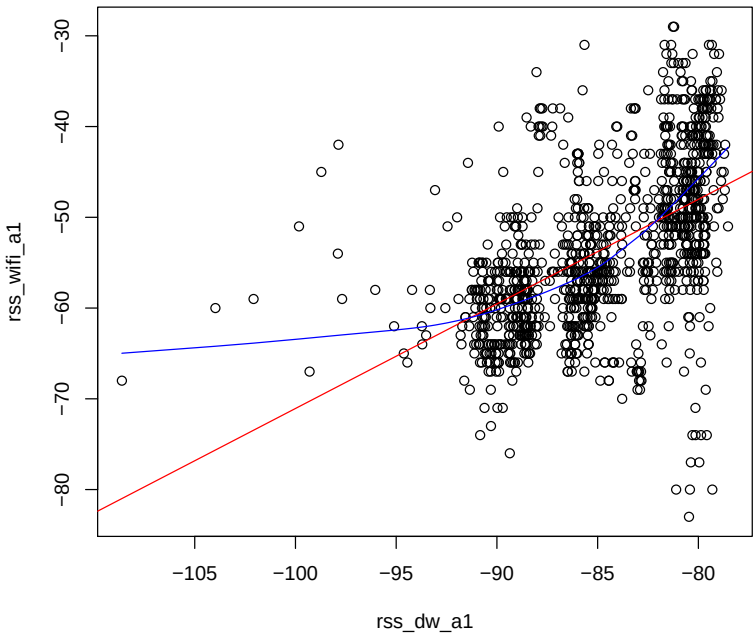


Figure 4.7: RSSI WiFi vs RSSI UWB of NLOS original database for anchor 1.

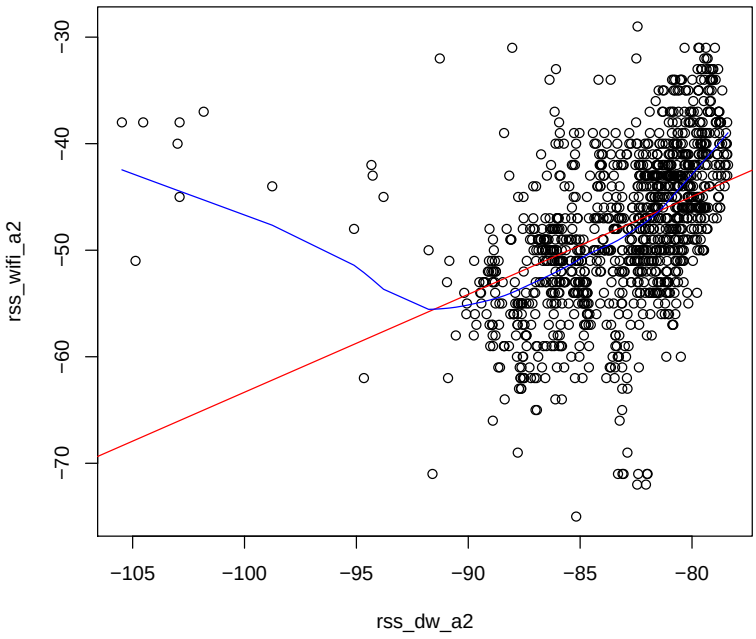


Figure 4.8: RSSI WiFi vs RSSI UWB of NLOS original database for anchor 2.

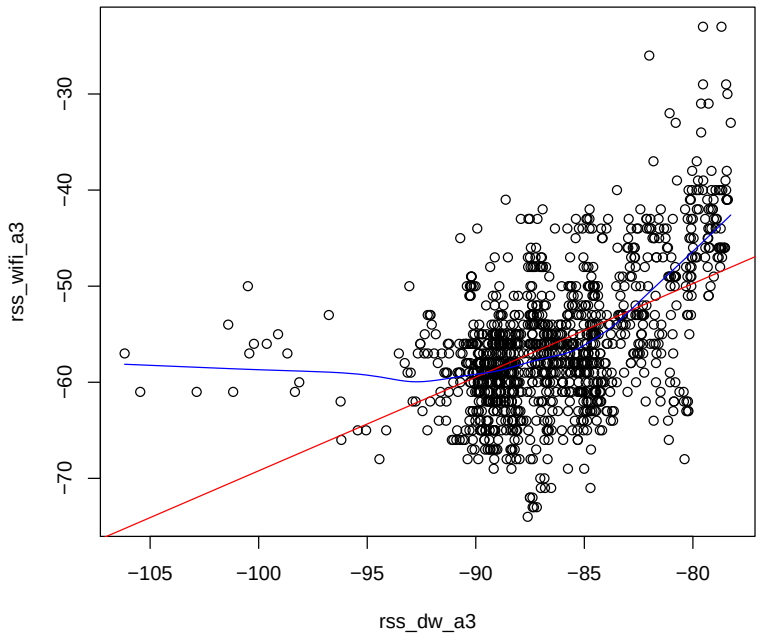


Figure 4.9: RSSI WiFi vs RSSI UWB of NLOS original database for anchor 3.

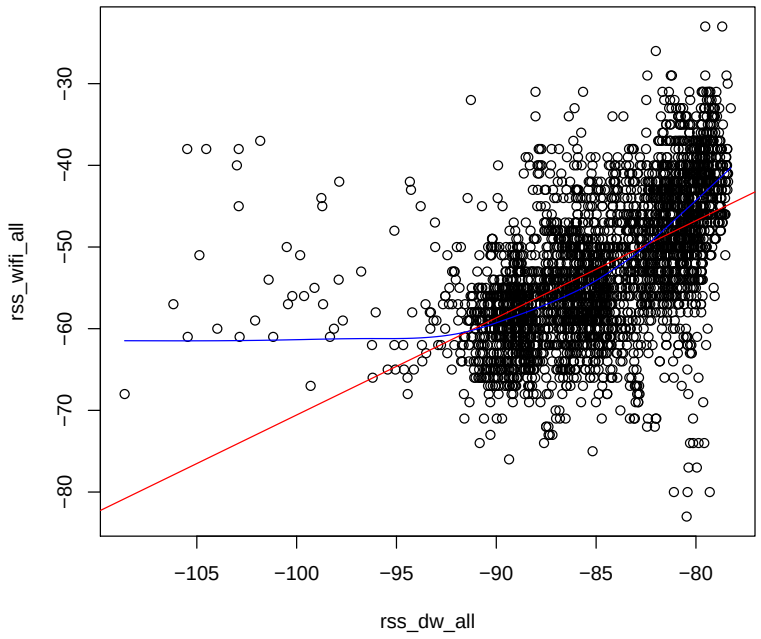


Figure 4.10: RSSI WiFi vs RSSI UWB of NLOS original database for all anchors.

the domain of DecaWave RSSI has a smaller range with respect to the one of the LOS original database, because all the values below -90 dbm are cut out. Furthermore, the correlation is increased for every single anchor and for all the anchors together too. This is because when you do the average of the values of all the entries with the same real coordinates, you eliminate some noise. Finally, if you compare the regression line with the LOWESS in Figure 4.10, you see that the linear trend is a very good approximation for all the range of the x-axis.

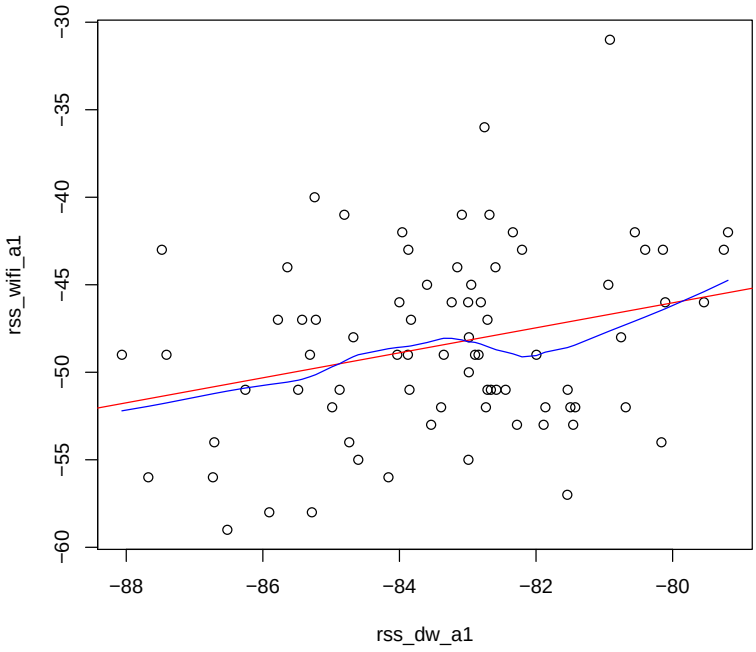


Figure 4.11: RSSI WiFi vs RSSI UWB of LOS condensed database for anchor 1.

In Figures 4.15, 4.16, 4.17 we show the plot derived from the data of the NLOS condensed database for every anchor, while in Figure 4.18 we aggregate the points of all the anchors together. The linear approximation is very good, expecially for anchor 1 and 2. Like in the LOS condensed database case, the correlation is higher with respect to the corresponding original database, and if you compare Figure 4.18 with Figure 4.10 you see that also for the NLOS database in the condensed case the domain of DecaWave RSSI has a smaller range with respect to the corresponding original database.

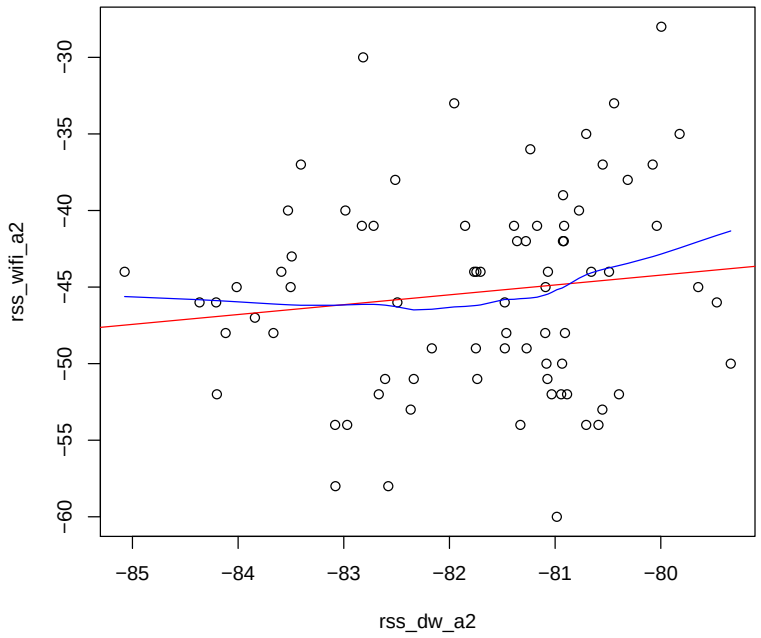


Figure 4.12: RSSI WiFi vs RSSI UWB of LOS condensed database for anchor 2.

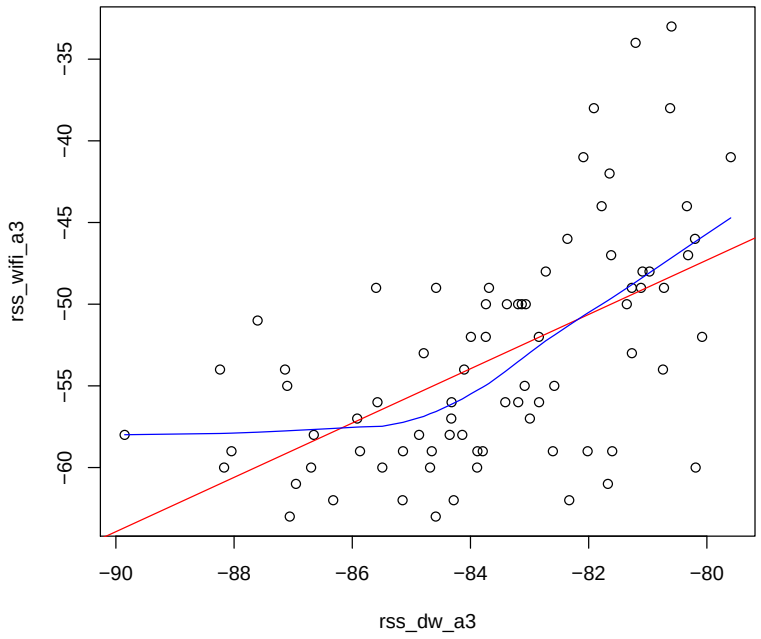


Figure 4.13: RSSI WiFi vs RSSI UWB of LOS condensed database for anchor 3.

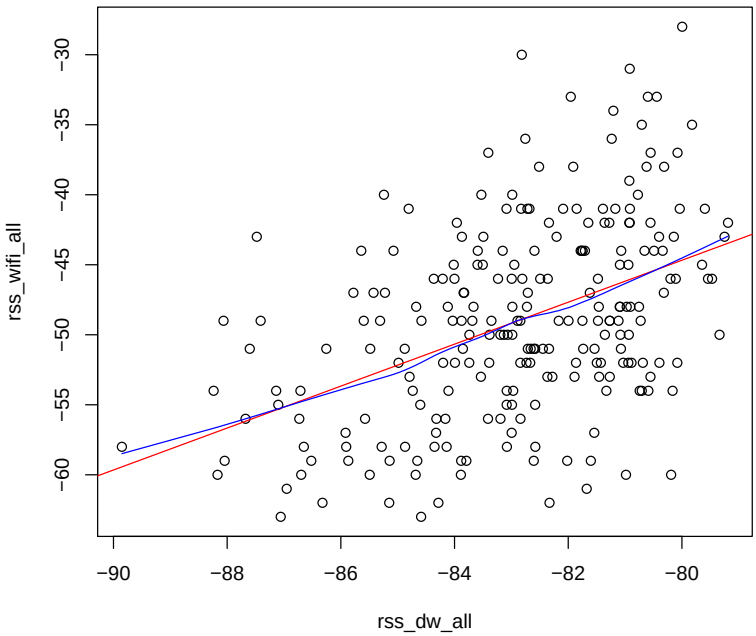


Figure 4.14: RSSI WiFi vs RSSI UWB of LOS condensed database for all anchors.

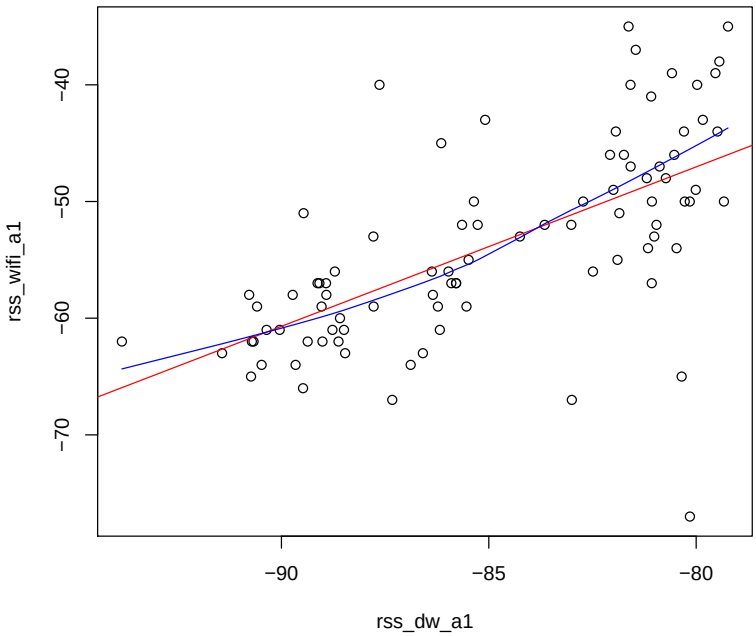


Figure 4.15: RSSI WiFi vs RSSI UWB of NLOS condensed database for anchor 1.

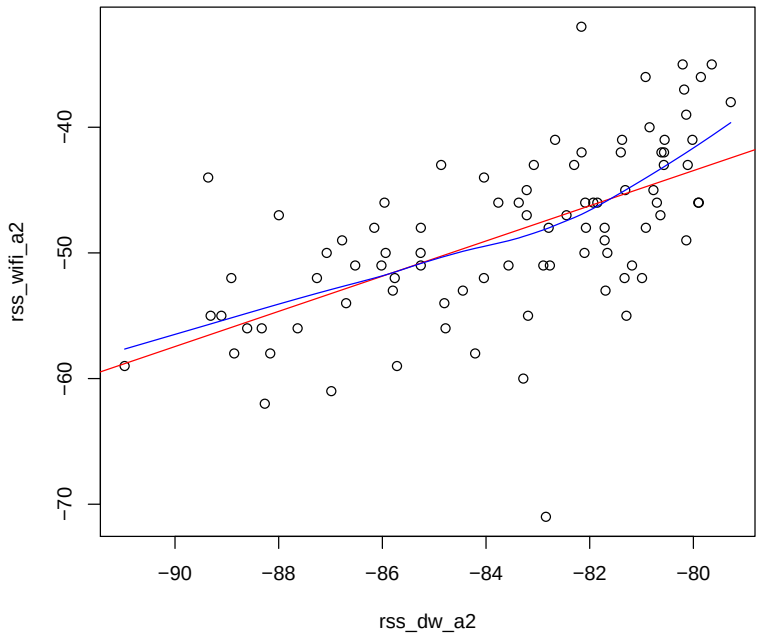


Figure 4.16: RSSI WiFi vs RSSI UWB of NLOS condensed database for anchor 2.

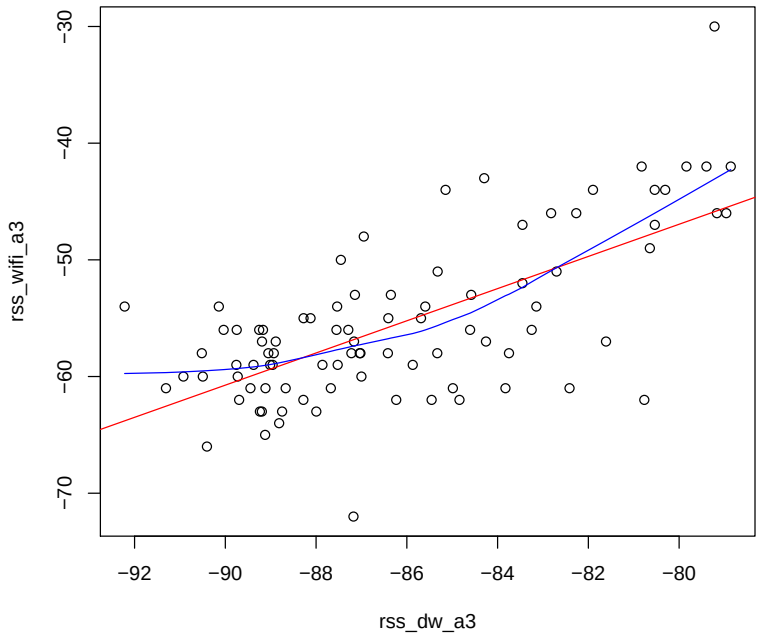


Figure 4.17: RSSI WiFi vs RSSI UWB of NLOS condensed database for anchor 3.

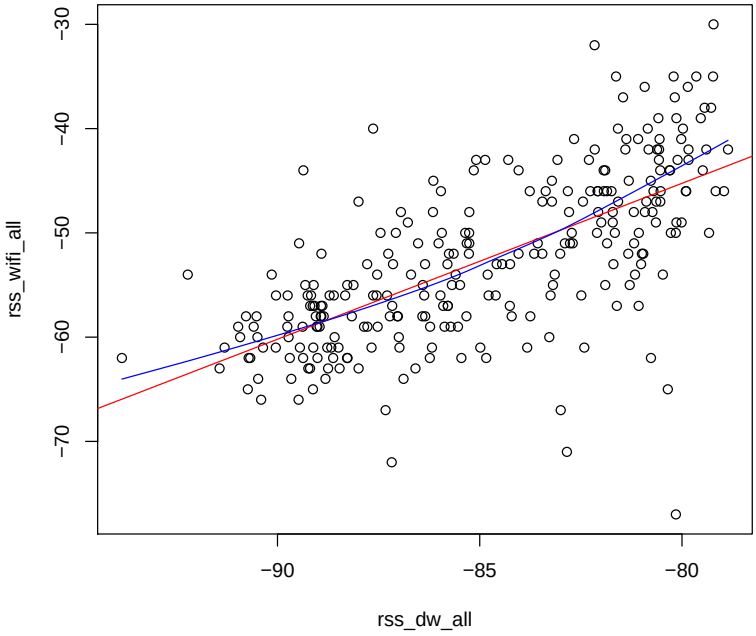


Figure 4.18: RSSI WiFi vs RSSI UWB of NLOS condensed database for all anchors.

4.3 BOXPLOTS

In the following section we are going to predict the RSSI WiFi values, therefore it is important to analyze the data in order to check if there are outliers. We choose to use boxplots for this purpose.

In Figures 4.19 and 4.20 we are going to see boxplots respectively for LOS and NLOS databases.

As we can see, the outliers are not numerous, therefore we decided to ignore them and leave the datasets as they are.

4.4 PREDICTION

Prediction may be very important when you are considering hybrid approaches that involve different technologies, because by predicting the results of one technology from the results of the other ones with a reasonable accuracy you may avoid to implement physically that technology in your system.

In this section we want to predict the RSSI value of every anchor measured from the Raspberry Pi in order to examine the relationship between the WiFi RSSI and the other data of the databases. We are going to do this analysis in the following two cases:

1. We have all the data of the dataset;

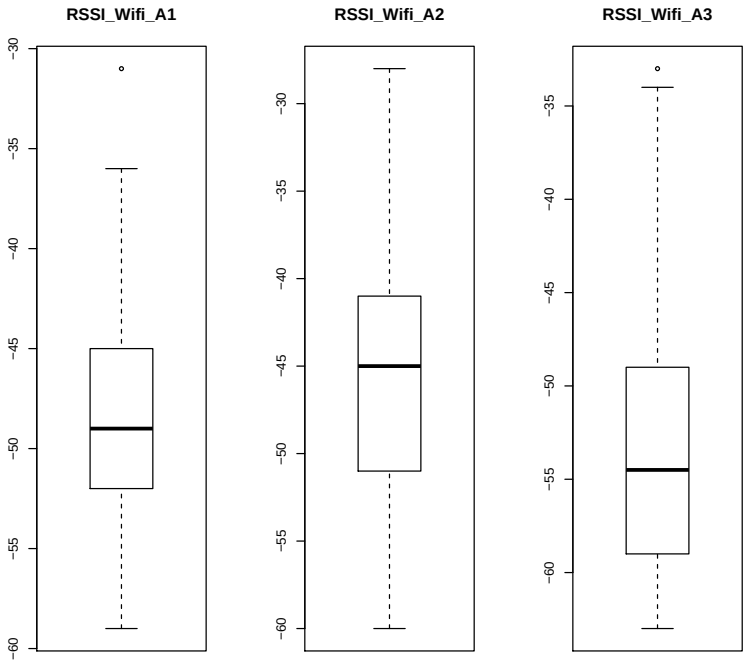


Figure 4.19: Boxplots: RSSI WiFi, LOS database.

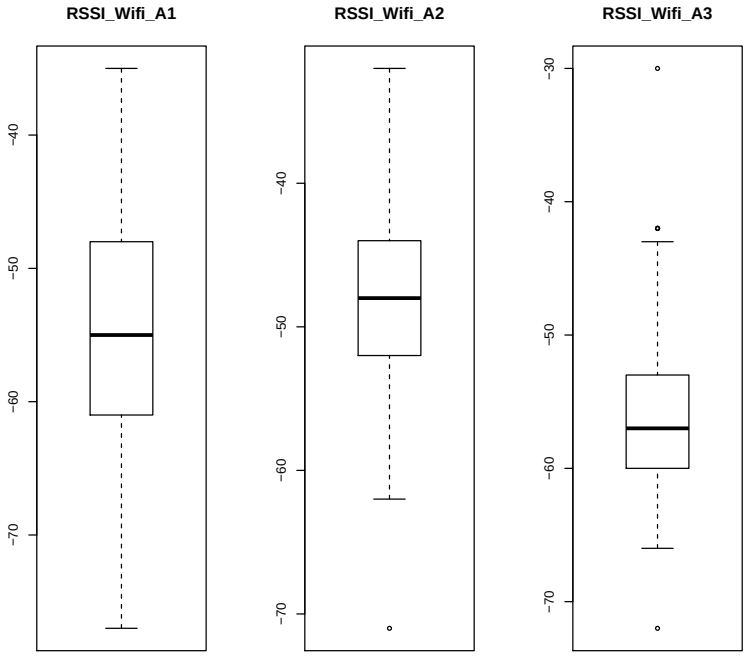


Figure 4.20: Boxplots: RSSI WiFi, NLOS database.

2. We have only the RSSI estimated by the DecaWave system.

The datasets were split into a 70:30 sample (training:test) and the values were sampled randomly, with a provided seed, in order to be able to replicate the same split in all the other experiments. Furthermore, we eliminated the `real_z` from the database, because it was almost always the same (the tag was positioned at the same height).

4.4.1 *Evaluation*

Using the model that we obtain, we evaluate the estimated RSSIs by comparing them with the real ones. In order to understand the accuracy of the predictor we use:

MEAN ABSOLUTE ERROR

$$MAE = mean\left(abs\left(predicteds - actuals\right)\right) \tag{4.1}$$

CORRELATION

$$\rho = \frac{cov(X, Y)}{\sigma_x \sigma_y} \tag{4.2}$$

MIN MAX ACCURACY

$$MMA = mean\left(\frac{min\left(actuals, predicteds\right)}{max\left(actuals, predicteds\right)}\right) \tag{4.3}$$

MEAN ABSOLUTE PERCENTAGE ERROR

$$MAPE = mean\left(abs\left(\frac{predicteds - actuals}{actuals}\right)\right) \tag{4.4}$$

ROOT MEAN SQUARE ERROR

$$RMSE = \sqrt{mean((predicteds - actuals)^2)} \tag{4.5}$$

Overall, for what regards the LOS dataset, we obtain the best results by using decision tree with all the data (case 1) and random forest with both cases, while we obtain the best results for the NLOS dataset by using linear regression in case 2 and random forest in case 1.

4.4.2 *Linear regression*

The results of linear regression prediction for case 1 are showed in Tables 4.3 and 4.4 respectively for LOS and NLOS datasets, while the results for case 2 are showed in Tables 4.5 and 4.6, respectively for LOS and NLOS datasets.

As you can see by the data showed in the tables, with linear regression we obtain better results in case 2.

Anchor	MAE	Correlation	MMA	MAPE	RMSE
a1	6.871	-0.372	1.164	0.157	8.515
a2	5.275	0.329	1.127	0.117	6.332
a3	5.406	0.672	1.117	0.114	6.457

Table 4.3: WiFi RSSI prediction with linear regression: case 1, LOS database.

Anchor	MAE	Correlation	MMA	MAPE	RMSE
a1	4.740	0.574	1.097	0.091	5.584
a2	4.993	0.543	1.118	0.113	6.348
a3	4.396	0.645	1.088	0.084	5.069

Table 4.4: WiFi RSSI prediction with linear regression: case 1, NLOS database.

Anchor	MAE	Correlation	MMA	MAPE	RMSE
a1	5.672	0.654	1.133	0.126	6.733
a2	5.333	0.019	1.127	0.117	6.263
a3	4.931	0.740	1.106	0.101	5.939

Table 4.5: WiFi RSSI prediction with linear regression: case 2, LOS database.

Anchor	MAE	Correlation	MMA	MAPE	RMSE
a1	3.156	0.741	1.065	0.062	4.374
a2	4.037	0.694	1.097	0.094	5.033
a3	3.721	0.744	1.072	0.071	4.412

Table 4.6: WiFi RSSI prediction with linear regression: case 2, NLOS database.

4.4.3 *Decision tree*

The results of decision tree prediction for case 1 are showed in Tables 4.7 and 4.8 respectively for LOS and NLOS datasets. As we can see, in case 1 almost all the statistics evaluation improved with respect to the linear regression method.

The results of decision tree prediction for case 2 are showed in Tables 4.9 and 4.10 respectively for LOS and NLOS datasets. As you can see by the data showed in the tables, with decision tree technique we obtain better results in case 1.

Finally, as opposed to case 1, in case 2 all the statistic evaluations are worse with respect to the linear regression method, except for anchor 2 in the LOS dataset.

Anchor	MAE	Correlation	MMA	MAPE	RMSE
a1	5.308	0.390	1.125	0.117	6.409
a2	4.815	0.392	1.114	0.105	5.937
a3	5.228	0.602	1.115	0.108	6.657

Table 4.7: WiFi RSSI prediction with decision tree: case 1, LOS dataset.

Anchor	MAE	Correlation	MMA	MAPE	RMSE
a1	3.467	0.761	1.070	0.068	4.650
a2	4.354	0.754	1.104	0.103	5.581
a3	4.358	0.654	1.086	0.085	5.297

Table 4.8: WiFi RSSI prediction with decision tree: case 1, NLOS dataset.

Anchor	MAE	Correlation	MMA	MAPE	RMSE
a1	6.820	-0.282	1.160	0.148	7.838
a2	4.165	0.504	1.099	0.093	5.160
a3	5.583	0.576	1.122	0.114	6.803

Table 4.9: WiFi RSSI prediction with decision tree: case 2, LOS dataset.

4.4.4 *Random Forest*

The results of random forest prediction for case 1 are showed in Tables 4.11 and 4.12 respectively for LOS and NLOS datasets, while the results for case 2 are showed in Tables 4.13 and 4.14 respectively for LOS and NLOS datasets.

Overall, with random forest we obtain better results in case 1. As you see, all the statistic evaluations are better then the results of the

Anchor	MAE	Correlation	MMA	MAPE	RMSE
a1	3.913	0.635	1.083	0.076	5.553
a2	5.031	0.505	1.122	0.116	6.171
a3	4.690	0.553	1.092	0.090	5.740

Table 4.10: WiFi RSSI prediction with decision tree: case 2, NLOS dataset.

tree method, except for the anchor 2 of LOS dataset. However, they are still worse than the linear regression method, except for anchor 2 of LOS dataset.

Anchor	MAE	Correlation	MMA	MAPE	RMSE
a1	6.212	-0.070	1.147	0.138	7.297
a2	4.578	0.474	1.108	0.098	5.560
a3	4.444	0.755	1.097	0.093	5.615

Table 4.11: WiFi RSSI prediction with random forest: case 1, LOS dataset.

Anchor	MAE	Correlation	MMA	MAPE	RMSE
a1	3.297	0.727	1.067	0.065	4.521
a2	4.174	0.747	1.101	0.100	5.442
a3	3.462	0.748	1.067	0.066	4.167

Table 4.12: WiFi RSSI prediction with random forest: case 1, NLOS dataset.

4.5 ASSESSING THE LOCALIZATION ACCURACY

In this section we want to evaluate the localization accuracy in different scenarios, by using different methods, features and technologies.

In the following, we are going to estimate the real position of the tag in six different cases:

1. We have all the data except the DecaWave estimated position;
2. We have only the RSSI of the Raspberry Pi's;
3. We have all the data except the DecaWave estimated position and the RSSI of the Raspberry Pi's;
4. We have all the data in the database.
5. We have all the data in the database except the RSSI of the Raspberry Pi's.
6. We have all the data in the database except both RSSIs

Anchor	MAE	Correlation	MMA	MAPE	RMSE
a1	5.868	-0.053	1.138	0.130	7.083
a2	4.479	0.445	1.106	0.098	5.486
a3	5.007	0.706	1.109	0.103	6.126

Table 4.13: WiFi RSSI prediction with random forest: case 2, LOS dataset.

Anchor	MAE	Correlation	MMA	MAPE	RMSE
a1	3.774	0.674	1.078	0.074	4.845
a2	4.210	0.706	1.103	0.102	5.623
a3	4.116	0.644	1.081	0.079	4.930

Table 4.14: WiFi RSSI prediction with random forest: case 2, NLOS dataset.

We summarise all the cases in Table 4.15:

In order to compare the results that we will obtain in this section with the DecaWave estimated positions performances, in the Table 4.16 we present the mean and the standard deviation of the distance between the real position and the one estimated by the DecaWave system, calculated over the same test dataset that we are using for the evaluations of this section.

As we mentioned earlier, in fact, we split the original database in two parts, the first one for the training (70%) and the second one for the test (30%). Therefore, we will evaluate the error mean and the error standard deviation as we did for Table 4.2, but only for the test dataset. Furthermore, by setting the same seed, we are able to reproduce the same 30% portion of the initial database in both use cases.

In Figure 4.21 we show the normal distribution of localization error for both the datasets.

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
DW position				X	X	X
DW distance	X		X	X	X	X
DW tof	X		X	X	X	X
DW RSSI	X		X	X	X	
WiFi RSSI	X	X		X		

Table 4.15: Localization cases summary.

dataset	error mean	error standard deviation
LOS	0.184	0.069
NLOS	0.415	0.306

Table 4.16: Localization error mean and standard deviation of test database.

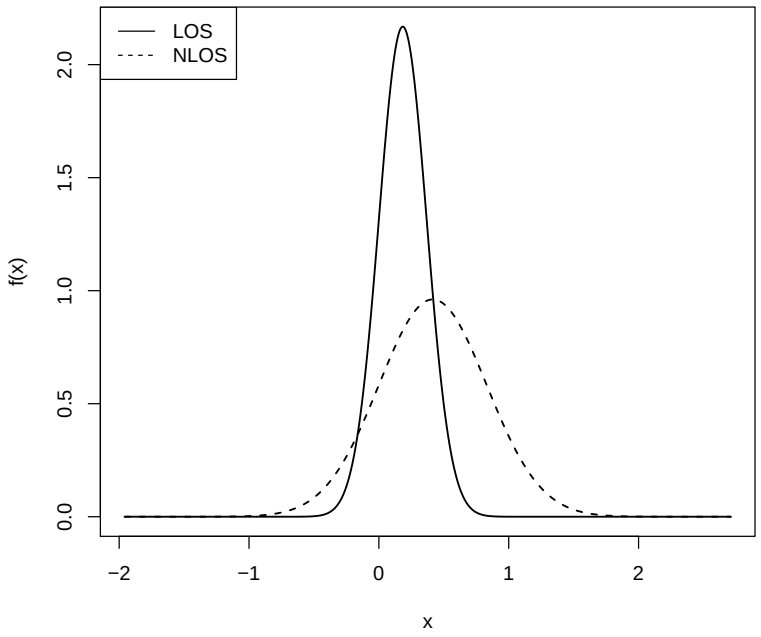


Figure 4.21: Normal distribution of condensed LOS and NLOS test datasets.

4.5.1 *Linear regression*

In this section we explain how we used linear regression in order to localize and we show the results of this operation.

In the following, we summarize the main steps of the R code that we wrote to localize with linear regression:

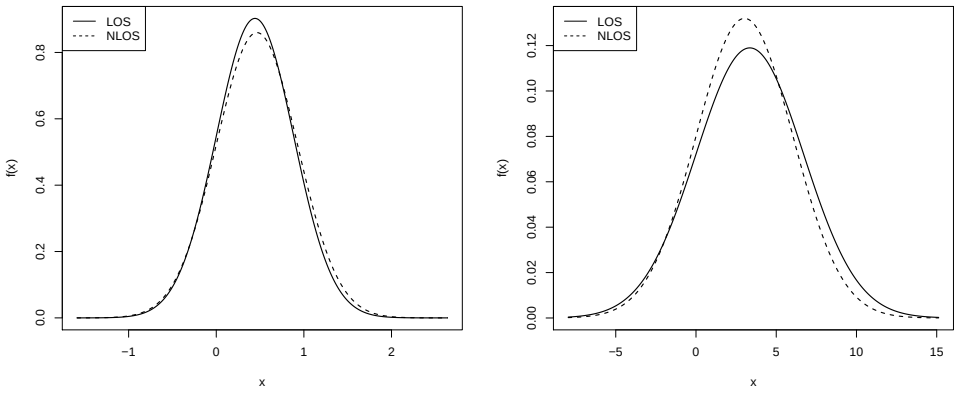
1. set a seed in order to be able to replicate the experiment;
2. split the dataset into a 70:30 sample (training:test);
3. use `lm` function to fit two linear models (one for the x-axis and one for the y-axis) with the data of the training set that vary depending on the case;
4. predict the position using the linear model with the data of the test set;
5. for each point calculate the distance between the estimated position and the real one;
6. evaluate error mean and standard deviation of the distances obtained in the previous point.

In Table 4.17 we report the results of localization with linear regression for all the 6 cases described previously. In order to facilitate the comparison with the DecaWave localization system, in the last entry of the table we report also the values from table 4.16. We indicate with EM the error mean and with ESD the error standard deviation. As you can see, for the LOS database we obtain the best results in case 6, while for the NLOS database we obtain the best results in case 5.

	LOS EM	LOS ESD	NLOS EM	NLOS ESD
Case 1	0.442	0.277	0.464	0.290
Case 2	3.353	1.564	3.023	1.569
Case 3	0.393	0.253	0.453	0.294
Case 4	0.091	0.062	0.331	0.240
Case 5	0.087	0.065	0.313	0.238
Case 6	0.070	0.067	0.328	0.267
DecaWave	0.184	0.069	0.415	0.306

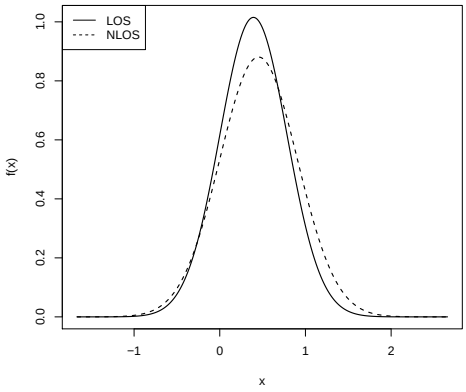
Table 4.17: Localization error mean and standard deviation with linear regression: summary.

In Figure 4.22 we plot the normal distribution of localization error (distance between estimated position and the real one) with linear regression applied both to LOS and NLOS databases, for each of the six cases.

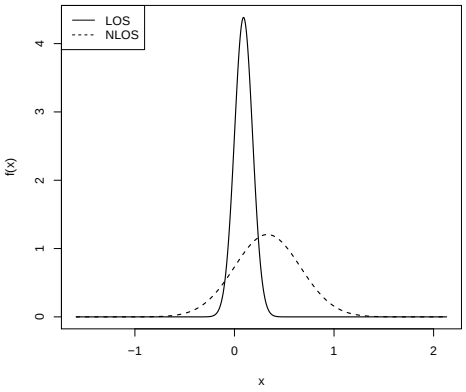


(a) Case 1.

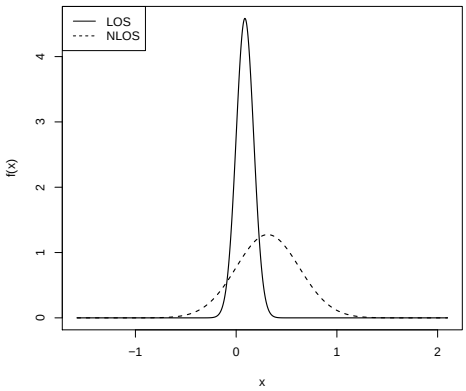
(b) Case 2.



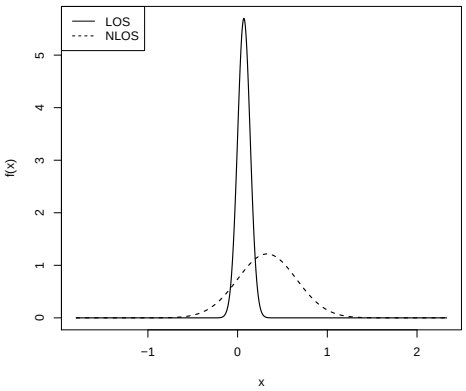
(c) Case 3.



(d) Case 4.



(e) Case 5.



(f) Case 6.

Figure 4.22: Normal distribution of localization error with linear regression.

If we compare the results of case 1 with the ones estimated by the DecaWave location system, we can see that even when we add the RSSI of the Raspberry Pi's, if we do not consider the DecaWave estimated position too, the position estimated with linear regression is less precise that the one estimated by the DecaWave localization software, since the only parameter that slightly improves is the error standard deviation of the NLOS database, while all the other parameters are lower.

As you can see from the results of case 2, removing the DecaWave information and estimating the location from WiFi RSSI only, with linear regression gives really bad results.

In case 3, with respect to case 1, we see that by removing the RSSI of the Raspberry Pi you improve the accuracy of the localization when you use linear regression.

If you compare the results of case 4 with the ones estimated by the DecaWave location system, we can see that by using the position estimated by the DecaWave system together with all the collected data, when we use linear regression we notice that there is a very good accuracy improvement. In fact, if we take LOS case as an example, the error mean and standard deviation are almost halved.

As we noticed by comparing case 3 with case 1, also for case 5 removing the WiFi RSSI leads to some improvements. Here we have another confirmation of this, because if we compare the results that we obtained with the one of case 4, we can see that all the evaluation results improved, except for error standard deviation of LOS dataset.

In case 6 we notice that by removing DecaWave RSSI with respect to case 5, the error mean of LOS dataset improves, while all the other results got worse.

4.5.2 *Decision tree*

In this section we show the results of localization using the decision tree technique. The R piece of code used for this purpose is almost the same of the one used for linear regression localization explained in Section 4.5.1, except for the `lm` function, which is replaced with the `rpart` function, which stands for "Recursive Partitioning And Regression Trees", with "anova" assigned to the `method` parameter, that specifies that the predicted value is numerical.

In Table 4.18 we show the results of localization with decision tree in case 4, 5 and 6, i.e. the cases which give better performances with linear regression, in order to try to obtain some improvements.

The results are the same for all the three cases and they are really bad compared to the performances obtained with linear regression.

In Figure 4.23 we plot the normal distribution of localization error for both LOS and NLOS databases.

dataset	error mean	error standard deviation
LOS	1.051	0.391
NLOS	1.067	0.488

Table 4.18: Localization error mean and standard deviation with decision tree: cases 4, 5 and 6.

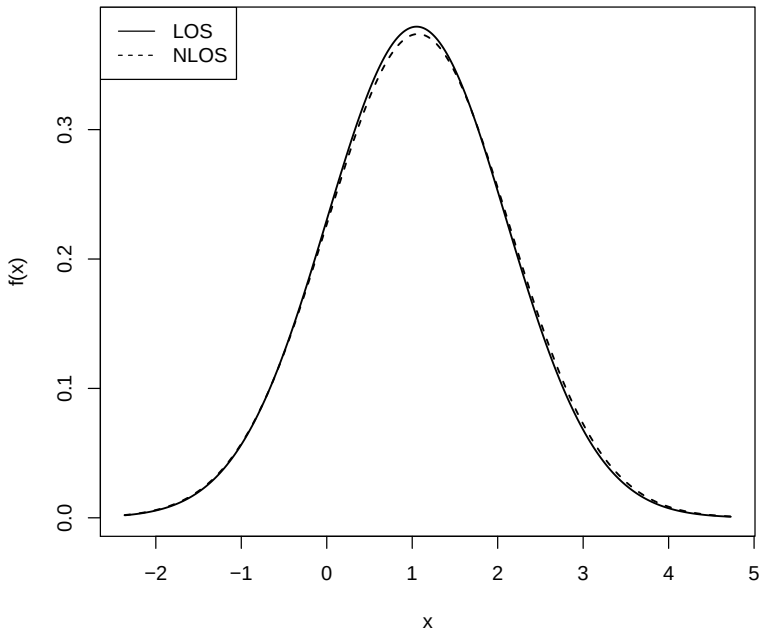


Figure 4.23: Normal distribution of localization error with decision tree: case 4, 5 and 6.

4.5.3 *Fingerprinting*

In this section we are going to estimate the location of the tag by using the fingerprint technique. The latter consists in comparing the RSSI vector measured by the Raspberry Pi's with the RSSI vectors stored in the database, finding the closest match and returning the corresponding coordinates.

Of course we cannot apply the fingerprinting algorithm over the condensed databases, because we have only one entry per point and therefore during the testing phase, for every match there would be an error of at least 1 meter.

Because of this, in the experiments done in this section we used the original LOS and NLOS databases, in which there are different entries per each real point and, as we did before, the databases were split into a 70:30 sample (training:test).

You may compare the values that we obtained in this section with the ones of case 2 from Table 4.17 in order to see the difference between this technique and linear regression. In general, we notice that this technique outperforms linear regression if the only information that we have from the tag is the WiFi RSSI vector.

A scenario like the previous one is not extremely rare, because notoriously UWB technology is more expensive than the one based on WiFi RSS. Therefore, you may have a situation in which you do not have the budget to buy enough UWB tags to be connected to all the entities that you want to localize, but instead you can afford enough WiFi RSS tags to do so. In this case, it may be convenient to use the fingerprinting technique, by using both UWB and RSS in the first phase in which you have to collect data from the environment, and RSS only in the localization phase.

In this way, during the first phase you collect fingerprints using both the UWB and the RSS tags and you fill a database with a structure like the one that we are considering in Table 3.2, that keeps track of both WiFi and UWB information. Together with these data, you have to store in the database also the estimated position, which for instance may be obtained with the linear regression method, as we have seen in Section 4.5.1. Next, during the second phase, in the reference scenario you have only an RSS tag for each entity and you use the RSSI vector measured by the tags to access the previously built database with the fingerprinting method and retrieve the corresponding estimated position. Therefore, instead of evaluating the position with a pure RSS algorithm, you are using the RSSI vector to obtain a position that was derived by exploiting both WiFi and UWB techniques.

In order to access the database and retrieve the position, we are going to use two algorithms: k-nearest neighbor and nearest neighbor, which is the special case of k-nearest neighbor with k=1.

4.5.3.1 *Nearest neighbor*

First, we apply a simple nearest neighbor algorithm i.e., given the WiFi RSSI vector measured by the tag, we find the entry of the database that has the closest RSSI vector with the given one and we return its position.

We report the mean and the standard deviation of the distance between the real position and the estimated one in Table 4.19.

dataset	error mean	error standard deviation
LOS	2.622	2.931
NLOS	2.021	2.329

Table 4.19: Localization error mean and standard deviation with nearest neighbor fingerprinting.

In Figure 4.24 we plot the normal distribution of localization error for both LOS and NLOS databases.

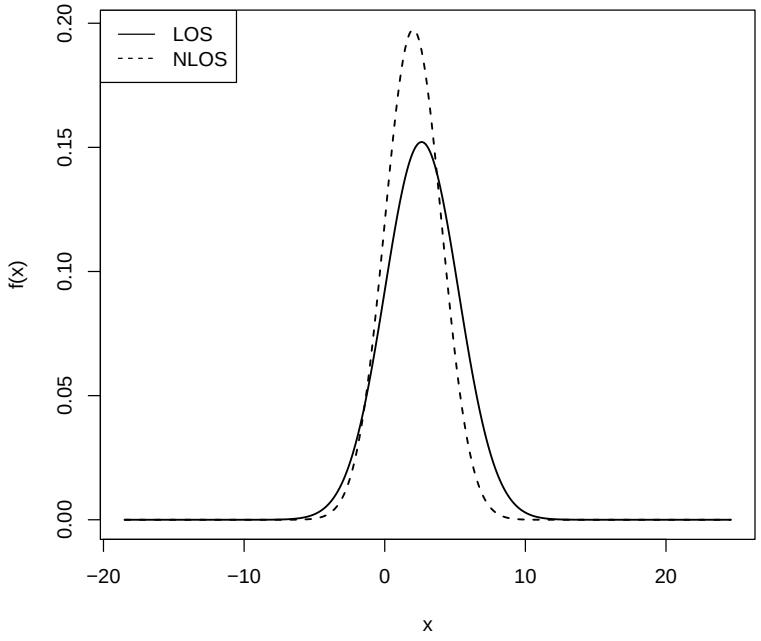


Figure 4.24: Normal distribution of localization error with fingerprinting: nearest neighbor.

4.5.3.2 *K-nearest neighbor regression*

With k-nearest neighbor regression algorithm, we do not limit our search to the nearest neighbor in the database, but we find the k nearest neighbors, i.e. the k pointsin the dataset that are closer to

the RSSI vector of the tag. At this point, the estimated position is calculated from the average of all the coordinates of the k points that were found.

In Figures 4.25 and 4.26 we report the error mean variation in function of k for both LOS and NLOS databases.

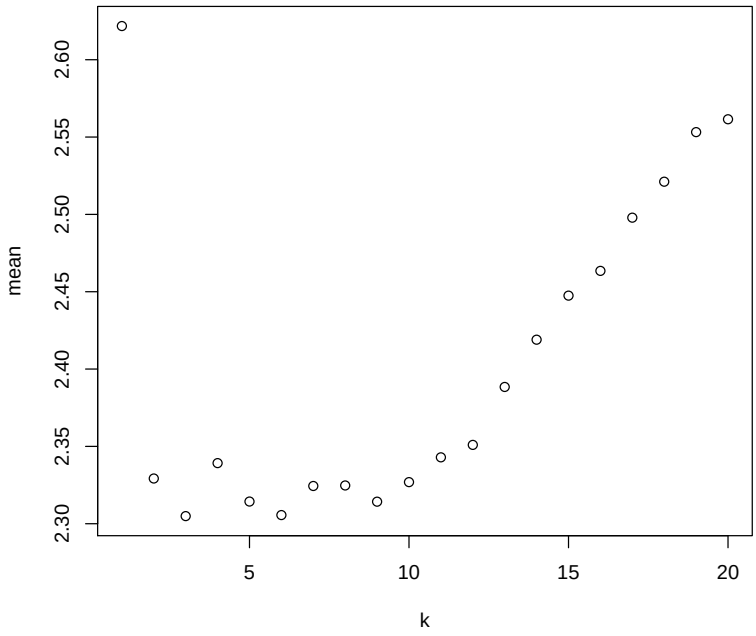


Figure 4.25: Localization error mean variation in function of k for LOS database.

In Table 4.20 we report the results of the best error mean that we obtained with this algorithm, together with the corresponding standard deviation and value of k.

dataset	error mean	error standard deviation	k
LOS	2.305	2.110	3
NLOS	1.945	1.900	2

Table 4.20: Localization error mean and standard deviation with k-nearest neighbor fingerprinting.

In Figure 4.27 we plot the normal distribution of localization error for both LOS and NLOS databases.

4.5.4 *RSSI Interpolation*

In this section we are going to simulate a situation in which we want to localize with WiFi RSS only, but we have a poorly populated

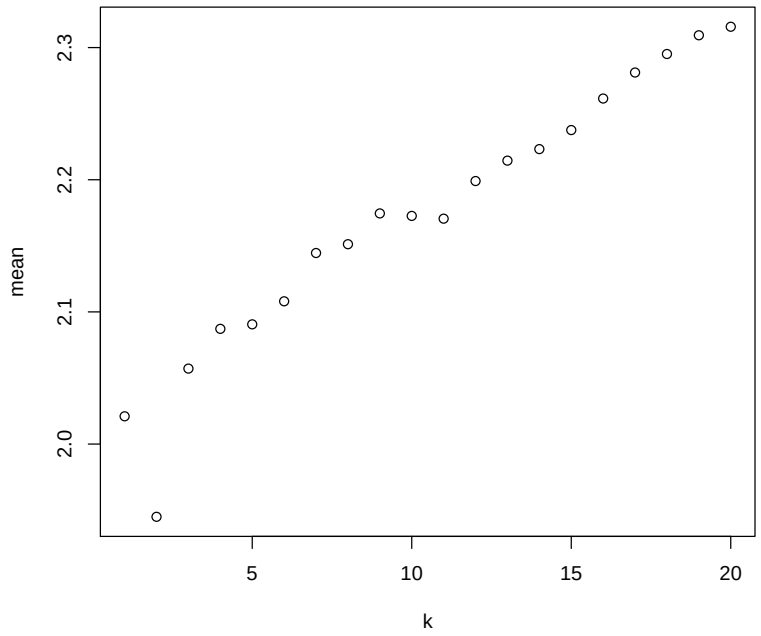


Figure 4.26: Localization error mean variation in function of k for NLOS database.

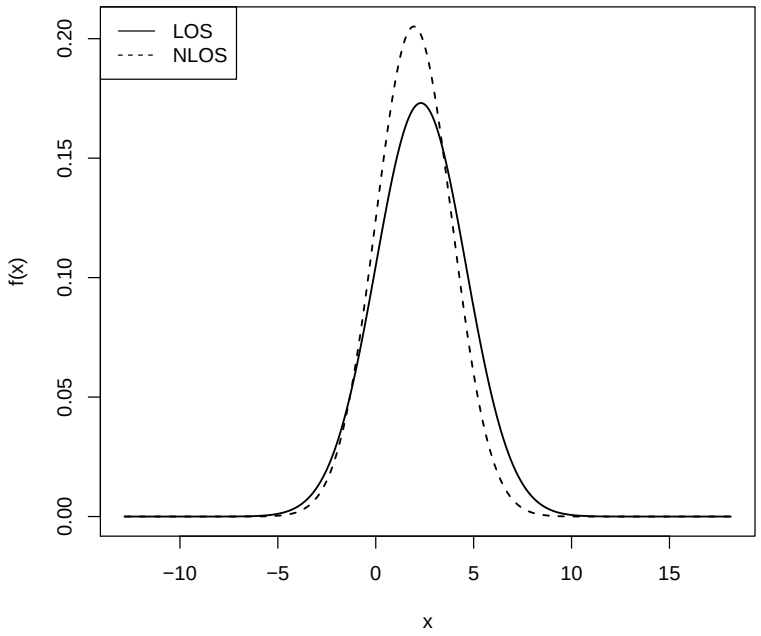


Figure 4.27: Normal distribution of localization error with fingerprinting: k-nearest neighbor.

database, i.e. in which the initial data collection phase was not done properly. We are going to see how localization accuracy changes if we fill some "missing entries" of this database with predicted RSSI vectors.

First, we extract from the training set only the points that have even `real_x` and `real_y`, like (0,0), (2,0), (0,2) and so on. In this way we reduce the LOS training set from 54 to 16 entries and the NLOS one from 62 to 18 entries. At this point we perform a first localization over this database and we evaluate the results that we obtain. We report the results of this operation in Table 4.21, while in Figure 4.28 we plot the normal distribution of localization error for both LOS and NLOS databases.

dataset	error mean	error standard deviation
LOS	3.142	1.788
NLOS	3.125	1.481

Table 4.21: Localization error mean and standard deviation on even coordinate points dataset.

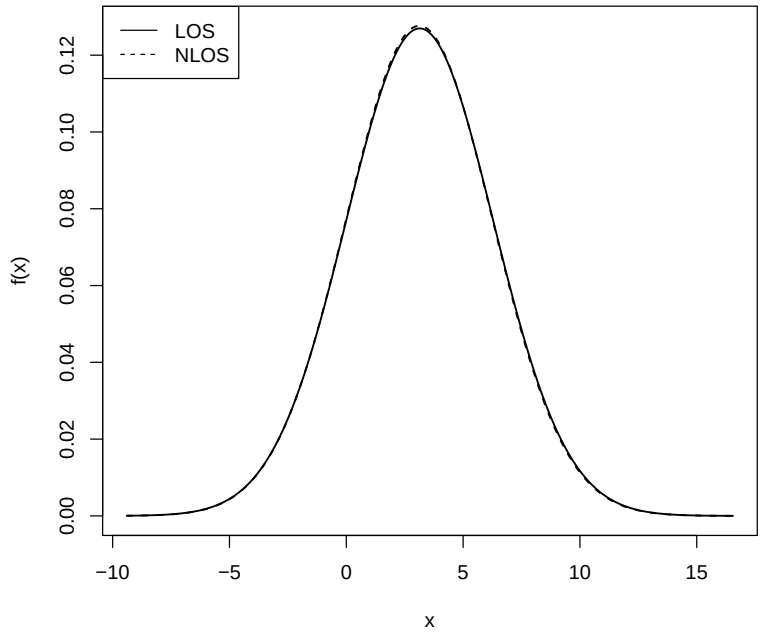


Figure 4.28: Normal distribution of localization error: even coordinate points dataset.

You may compare these results with the ones of case 2 from Table 4.17, in order to see what is the effect of removing more than one third of the values from the training set.

Next, for every missing point, we fill the database with the WiFi RSSI values predicted in the 2 scenarios that we considered in Section 4.4, i.e. case 1 in which we consider all the fields of the database except `real_z` and case 2 in which we predict with DecaWave RSSI only. The prediction operation will be executed with all the techniques that we used previously, i.e. linear regression, decision tree and random forest.

In the following you can see the results of the two cases considering the new database where missing points were filled.

As you can see for the LOS dataset the only result that improves is the error standard deviation in case 2, while for the NLOS dataset the only value that does not improve is the standard deviation in case 1.

4.5.4.1 *Linear regression*

The results of localization with linear regression prediction for case 1 are showed in Table 4.22, while in Figure 4.29 we plot the normal distribution of localization error for both LOS and NLOS databases.

dataset	error mean	error standard deviation
LOS	3.716	1.959
NLOS	3.018	1.513

Table 4.22: RSSI interpolation: localization error mean and standard deviation with linear regression, case 1.

The results of localization with linear regression prediction for case 2 are showed in Table 4.23, while in Figure 4.30 we plot the normal distribution of localization error for both LOS and NLOS databases.

dataset	error mean	error standard deviation
LOS	3.483	1.552
NLOS	3.114	1.439

Table 4.23: RSSI interpolation: localization error mean and standard deviation with linear regression, case 2.

4.5.4.2 *Decision tree*

The results of localization with decision tree prediction for case 1 are showed in Table 4.24, while in Figure 4.31 we plot the normal distribution of localization error for both LOS and NLOS databases.

The results of localization with decision tree prediction for case 2 are showed in Table 4.25, while in Figure 4.32 we plot the normal distribution of localization error for both LOS and NLOS databases.

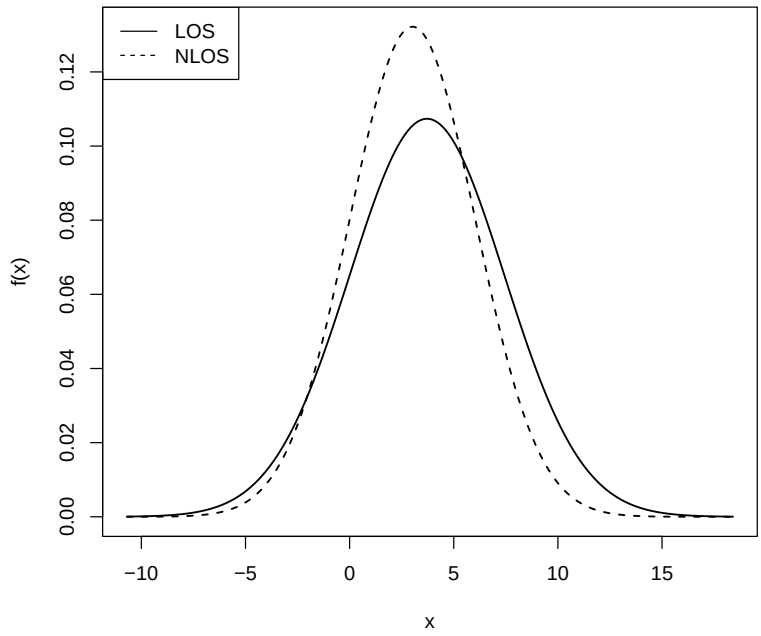


Figure 4.29: RSSI interpolation: normal distribution of localization error with linear regression, case 1.

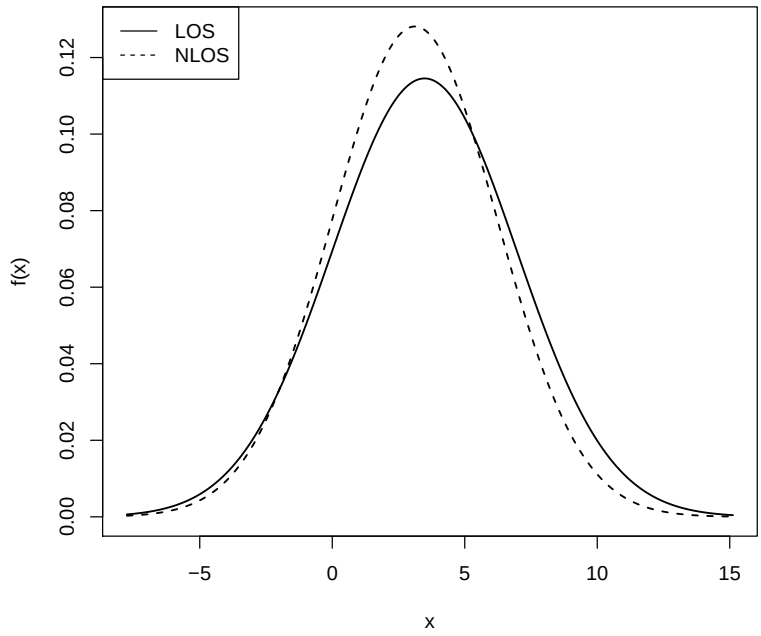


Figure 4.30: RSSI interpolation: normal distribution of localization error with linear regression, case 2.

dataset	error mean	error standard deviation
LOS	3.466	1.434
NLOS	3.179	1.538

Table 4.24: RSSI interpolation: localization error mean and standard deviation with decision tree, case 1.

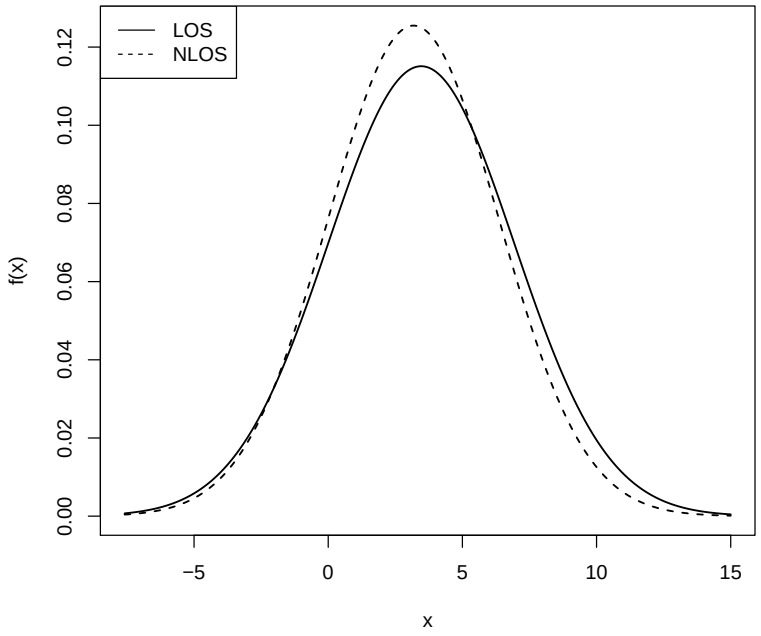


Figure 4.31: RSSI interpolation: normal distribution of localization error with decision tree, case 1.

dataset	error mean	error standard deviation
LOS	3.421	1.350
NLOS	3.035	1.425

Table 4.25: RSSI interpolation: localization error mean and standard deviation with decision tree, case 2.

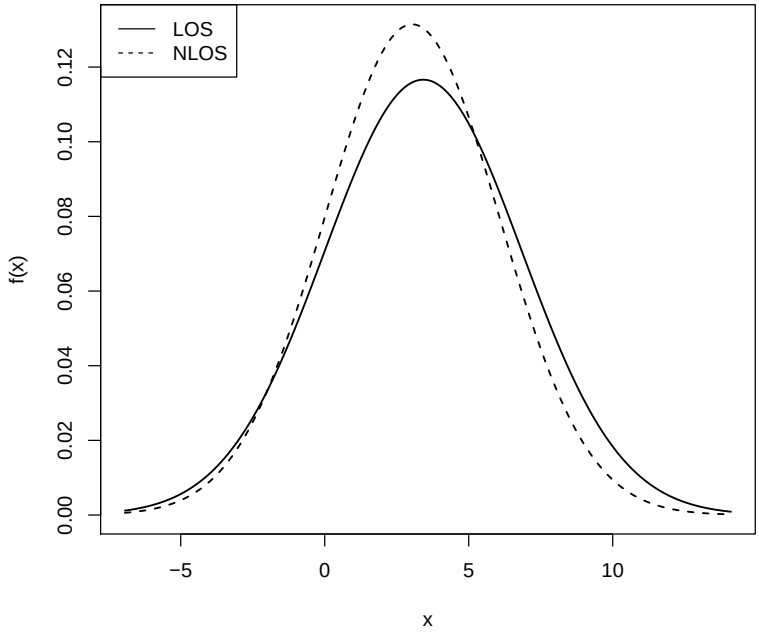


Figure 4.32: RSSI interpolation: normal distribution of localization error with decision tree, case 2.

4.5.4.3 *Random forest*

The results of localization with random forest prediction for case 1 are showed in Table 4.26, while in Figure 4.33 we plot the normal distribution of localization error for both LOS and NLOS databases.

dataset	error mean	error standard deviation
LOS	3.433	1.718
NLOS	3.059	1.548

Table 4.26: RSSI interpolation: localization error mean and standard deviation with random forest, case 1.

The results of localization with random forest prediction for case 2 are showed in Table 4.27, in Figure 4.34 we plot the normal distribution of localization error for both LOS and NLOS databases.

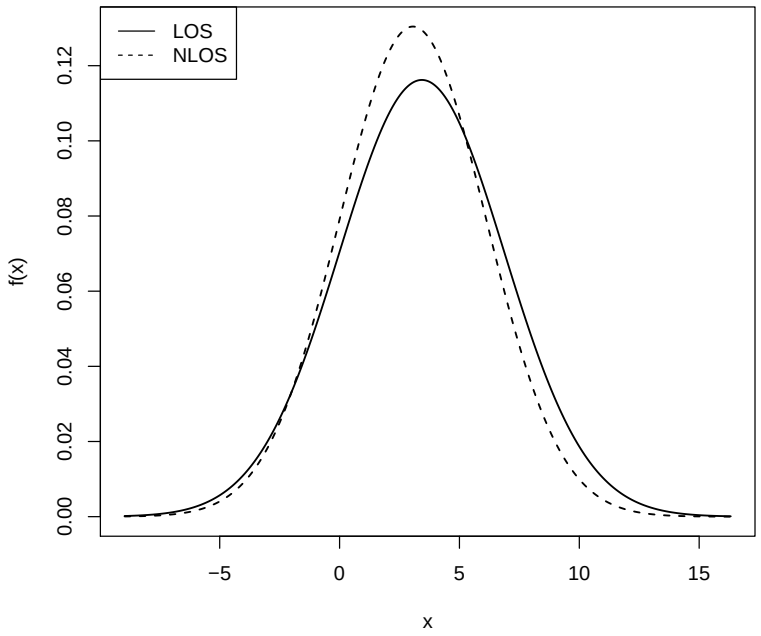


Figure 4.33: RSSI interpolation: normal distribution of localization error with random forest, case 1.

dataset	error mean	error standard deviation
LOS	3.453	1.525
NLOS	3.067	1.499

Table 4.27: RSSI interpolation: localization error mean and standard deviation with random forest, case 2.

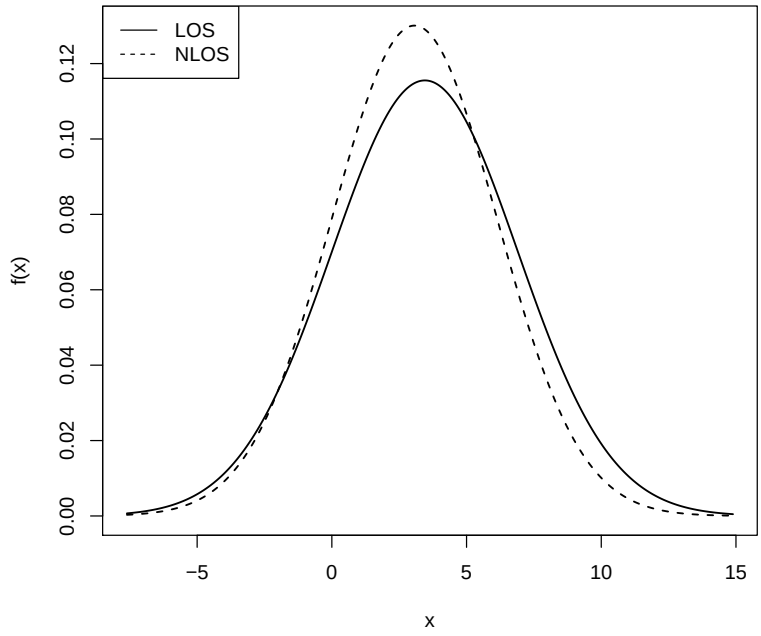


Figure 4.34: RSSI interpolation: normal distribution of localization error with random forest, case 2.

5

CONCLUSION

The work presented in this thesis is related to IPSs, a field where there is not a standard yet and whose importance nowadays is constantly growing. One of the main challenge in the industry 4.0 era, in fact, is to develop an IPS which is scalable, accurate and cost-effective.

In the thesis we evaluated the performances of the DecaWave localization system, one of the most important commercial IPSs on the market, and we combined different localization techniques with each other, in different scenarios and with different features, in order to define some algorithms that result in better performances or that achieve a better tread-off between accuracy and cost.

For this purpose, we developed a testbed for hybrid indoor localization that is able to collect data by exploiting both UWB and WiFi wireless protocols, respectively with RSS and TWR localization techniques. We carried out the measurements at the IoT Lab (Laboratorio di ricerca applicata del Politecnico di Milano) [23], in which we were able to reproduce both LOS and NLOS conditions thanks to the presence of wide areas and different industrial machines. In particular, the testbed that we developed is able to collect: UWB distance, UWB TOF, UWB RSSI and WiFi RSSI. The UWB information are provided by the DecaWave TREK1000 evaluation kit, while the WiFi information are given by four Raspberry Pi 3 model B.

After the measurement process, we analyzed the data collected and we evaluated different localization algorithms. In particular, we found an algorithm based on linear regression that is able to considerably increase the accuracy of the DecaWave localization system and another one based on fingerprinting that reduces the cost of UWB localization, by achieving an accuracy which is almost in the middle between the one obtained in our experiments by using WiFi RSSI only and the one given by the DecaWave localization system.

The first step of the first algorithm that we mentioned consists in dividing the area that we used to perform localization into a grid with squares of 1 meter x 1 meter and we taking measurements samples at each point. The measurements sample should include real position of the tag, UWB distance, UWB TOF and UWB RSSI (the latter only if you are in LOS conditions). After you collected these data you use the DecaWave localization system to evaluate the positions of each point and you add the obtained coordinates to the dataset. Finally, you perform linear regression by building a predictor for each coordinate of the position. When you have to perform localization, you give the measurements that you obtain from the UWB and WiFi tags as input

to the predictors and you obtain the estimated position as output. So, with this method, it is possible to improve the accuracy of the DecaWave localization system up to double it without adding extra-hardware.

If your goal is not to have the highest possible accuracy, but instead to have a good tread-off between accuracy and cost, you may use the following hybrid localization algorithm, which is based on fingerprinting and both WiFI and UWB wireless protocols. The first steps of this algorithm are similar to the previous ones, but instead of collecting only real position of the tag, UWB distance, UWB TOF and UWB RSSI, your measurement sample should include also WiFi RSSI. Therefore, in this phase you need to build a database with both WiFi and UWB devices, while in the previous algorithm we did not used WiFi devices at all. Together with these data, you have to store in the database also the estimated position, which may be obtained with the linear regression method by using UWB data only, like in the previous algorithm. Then, during the localization phase, you use the WiFi RSSI measured by the tag to access the database that you have built with the fingerprinting technique. With this method, you need one WiFi tag per each entity that you want to localize, but for what regards the UWB technology you only need a basic evaluation kit, that is used in the data collection phase only. Therefore, since WiFi infrastructures is notoriously cheaper than UWB ones, with this method you obtain a cheaper system, but at the same time you get access to a position which was estimated by using UWB technology.

However, as you may have noticed, both the algorithms that we showed have a not negligible drawback: they do not work out of the box like the DecaWave localization system, but they require an initial data collection procedure in which you analyze the environment in which you want to perform localization.

BIBLIOGRAPHY

[1] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1067–1080, 2007.

[2] Yuan Zhou, Choi Look Law, Yong Liang Guan, and Francois Chin. Indoor elliptical localization based on asynchronous uwb range measurement. *IEEE Transactions on Instrumentation and Measurement*, 60(1):248–257, 2011.

[3] Fadhel M Ghannouchi, Donglin Wang, and Smita Tiwari. Accurate wireless indoor position estimation by using hybrid tdoa/rss algorithm. In *Vehicular Electronics and Safety (ICVES), 2012 IEEE International Conference on*, pages 437–441. IEEE, 2012.

[4] Konstantin Mikhaylov, Antti Tikanmäki, Juha Petäjäjärvi, Matti Hämäläinen, and Ryuji Kohno. On the selection of protocol and parameters for uwb-based wireless indoors localization. In *Medical Information and Communication Technology (ISMICT), 2016 10th International Symposium on*, pages 1–5. IEEE, 2016.

[5] Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen. A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi. In *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, pages 46–51. Ieee, 2007.

[6] Davide Dardari, Pau Closas, and Petar M Djurić. Indoor tracking: Theory, methods, and technologies. *IEEE Transactions on Vehicular Technology*, 64(4):1263–1278, 2015.

[7] Mohamed Laaraiedh, Stéphane Avrillon, and Bernard Uguen. Hybrid data fusion techniques for localization in uwb networks. In *Positioning, Navigation and Communication, 2009. WPNC 2009. 6th Workshop on*, pages 51–57. IEEE, 2009.

[8] Sinan Gezici, Zhi Tian, Georgios B Giannakis, Hisashi Kobayashi, Andreas F Molisch, H Vincent Poor, and Zafer Sahinoglu. Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. *IEEE signal processing magazine*, 22(4):70–84, 2005.

[9] Karthikeyan Gururaj, Anojh Kumaran Rajendra, Yang Song, Choi Look Law, and Guofa Cai. Real-time identification of nlos range measurements for enhanced uwb localization. In *Indoor*

Positioning and Indoor Navigation (IPIN), 2017 International Conference on, pages 1–7. IEEE, 2017.

[10] Davide Dardari, Andrea Conti, Ulric Ferner, Andrea Giorgetti, and Moe Z Win. Ranging with ultrawide bandwidth signals in multipath environments. *Proceedings of the IEEE*, 97(2):404–426, 2009.

[11] Sichun Wang, Brad R Jackson, and Robert Inkol. Hybrid rss/aoa emitter location estimation based on least squares and maximum likelihood criteria. In *Communications (QBSC), 2012 26th Biennial Symposium on*, pages 24–29. IEEE, 2012.

[12] Rajika Kumarasiri, Khair Alshamaileh, Nghi H Tran, and Vijay Devabhaktuni. An improved hybrid rss/tdoa wireless sensors localization technique utilizing wi-fi networks. *Mobile Networks and Applications*, 21(2):286–295, 2016.

[13] Mohammad Reza Gholami, Reza Monir Vaghefi, and Erik G Ström. Rss-based sensor localization in the presence of unknown channel parameters. *IEEE Transactions on Signal Processing*, 61(15):3752–3759, 2013.

[14] Decawave. *TREK1000: Two-Way-Ranging (TWR) RTLS IC Evaluation Kit*. `https://www.decawave.com/sites/default/files/trek1000_product_brief.pdf`.

[15] Decawave. *PRODUCT INFORMATION: EVB1000*. `https://www.decawave.com/sites/default/files/resources/evb1000-product-brief_3.pdf`.

[16] Decawave. *PRODUCT INFORMATION: DW1000*. `https://www.decawave.com/sites/default/files/dw1000-product-brief_2.pdf`.

[17] Decawave. *DW1000 USER MANUAL: how to use, configure and program the dw1000 uwb transceiver*, 2.13 edition. `https://www.decawave.com/sites/default/files/dw1000_user_manual_2.13.pdf`.

[18] Raspberry pi website. `https://www.raspberrypi.org/`. Accessed: 2018-06-03.

[19] Raspberry pi 3 model b webpage. `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/`. Accessed: 2018-05-31.

[20] Raspbian website. `https://www.raspbian.org/`. Accessed: 2018-05-31.

[21] St-link/v2 webpage. http://www.st.com/en/development-tools/st-link-v2.html. Accessed: 2018-05-31.

[22] How to use your raspberry pi as a wireless access point. https://thepi.io/how-to-use-your-raspberry-pi-as-a-wireless-access-point/, November 2017. Accessed: 2018-05-31.

[23] Iot lab website. http://www.iotlab.it/. Accessed: 2018-06-03.

[24] Decawave. *TREK1000 User Manual*, 1.06 edition. https://www.decawave.com/sites/default/files/trek1000_user_manual_1.06_1.pdf.