POLITECNICO DI TORINO

Master degree course in Computer Engineering

Master Degree Thesis

Analysis and realization of smart labels for object interaction tracking

Supervisor
prof. Paolo Prinetto

Candidate
Marco Magliona
matricola: S231639

Anno accademico 2016–2017

Summary

The aim of this thesis is to create a “proof of concept” system using existing technologies, like RFID and/or optical recognition, with the purpose of tracking goods during their distribution/selling cycle.

Traditional labels with bar codes are only able to identify an object and when it is bought so, as an enhancement step, particular attention will be payed to identify the type(s) of interaction a potential customer has with the object inside the store, like nearing, touching, moving and finally buying.

The Part I is dedicated to comparing different technologies and how they can be exploited in this scenario: not only performances need to be evaluated in this phase but availability and cost too. The capability of computing the distance transmitter-receiver exploiting the characteristics of a technology is of particular relevance in this phase: most of the interactions with an object involves the movement so a variation of position/distance. The precision obtained in these measurements influences directly the performances of the final system.

In Part II I will concentrate on UHF RFID and study which are the characteristics of this technology and its protocol, particularly which raw data an high end RFID reader offers and how reliable the system is. In this phase I will try to identify the most important intrinsic problems and try to reproduce them in a real life environment.

In Part III I will use the analysis in part two to try to achieve, step by step, the initial aim keeping an eye on already existing papers and studies. In this part I will write about the actual implementation and data manipulation. A little digression will be done on the used artificial intelligence, a feed-forward neural network, I have written from scratch, which has been modified in order to achieve a good precision as a classifier of the different human-object interactions. A comparison with the current “state of art” will conclude this part.

In the Part IV I will propose some possible ideas for further enhancements.

Contents

List of Tables

List of Figures

Introduction

In the last years the retail/consumer sector showed the need of obtaining more comprehensive information about the interest aroused by a good in a potential customer. This is not possible with the current status of the technology employed in the market which is only able to track when a product has been sold and in which quantities. While this is fundamental to properly manage the supply chain and the local stock, does not help to diagnose possible problems in the resell phase that cause customer to disregard a particular good.

If more data about the time spent by a product in a shelf is available, a more deep analysis can be done in order to improve the market uptake of a particular good.

This can be done if the potential customer behavior, with respect to the product, is known. Some possible situations are:

- The customer completely ignores the product
- The customer watches the item, goes near it to better study item
- The customer picks the item, interact with it for some time and
  - buys it
  - decide to put it back on the shelf

While these are some generic examples, knowing the statistics related to them can give some tips to increase the market traction of that particular good and helps in solving the issues that prevents the product to be sold.

A new technology needs to be found and exploited to obtain the same functionalities of the old one and extend them so to satisfy the new needs.

RFID UHF technology can be used to entirely substitute the current system used in the consumer market like barcodes. The features in the old technologies are part of the RFID protocol or are an improved version of them: as an example is possible to have "per object" identification opposed to "per product" identification which allows to mark individual goods as already sold.

The chosen technology has to be able to identify the human-object interactions and give back some statistical data which can be directly related to particular customer behaviors like those listed before.

The aim of this thesis is:

- Identify a suitable technology

- Identify among its capabilities the ones that can be used for the scope
- Find a way to elaborate the data obtained to have the information required
- Test and implement a "proof of concept" system to show the real feasibility of the method
- Propose some improvements to further refine the system and adapt to more specific use cases

The current state of the art «IDSense: A human object interaction detection system based on passive UHF RFID», which uses RFID UHF, has achieved good results in this field but it is affected by some issues that can prevent a widespread usage. The most important problem is the use of the Asian regulations for the employed technology which are, like the Americans, quite lax with respect to those used in other regions like Europe, making it unsuitable for deployment in these more restrictive regions.

Even if the global idea can be kept, the system needs significant alteration in order to work with less environmental data. This is reflected in a generally more complex system, particularly the data elaboration part has to be improved. Not only the data given by the technology has to be processed differently but the artificial intelligence, AI from here on, needs to be more advanced to compensate the loss of input information.

A more complex AI involves more tuning parameters that have to be identified. The AI is the most relevant software factor which influences the final precision: the rest of the system is in charge of preparing the data for the AI and passing the results to the user.

The AI should be able to provide the immediate interaction associated with the data given by the technology. The most general examples of possible interaction are "still, untouched", "touched" or "almost touched", "moved from/to inside of the reading range" and "moved to out of the reading range".

The possible interactions should be available to a user or can be further processed to identify a specific situation like those mentioned before. This additional identification is not part of the aim of this thesis. The only additional step that is done is to have a percentage for each interaction over the total time for each object/product. This last part will be mainly done to analyze the precision of the system in a real scenario but can be still used as a preliminary indicator of the market traction for that product: a good that is touched often can be "visibly appetizing", one which is often moved, goes "out of range" and then return on the shelf can be considered as "interesting" but not "satisfying" for the customer.

Another important point is that the system should be as more modular as possible. This allows the independent development of the various parts and their interchangeability making it adaptable to more particular scenarios.

The implementation of the "proof of concept" system designed here has been done with generality in mind: the final aim is to prove the feasibility of the solution.

Due to the fact that the system has been developed while following the strict European regulations on RFID UHF, it should be possible to use "as is" in regions with more relaxed regulations or extended to use the wider range of frequencies. In the second case, it can be expected an higher precision than that reached with the European frequencies which is around 90 ~95 %, obtained in a real environment (open space office).

Part I

Available technologies

Chapter 1

Visual tracking

1.1 Barcodes

Barcodes are used to identify objects with special labels which contain data encoded as lines or squares and the corresponding reader. Barcodes are today massively used not only in the retail sector but also in logistic and supply chain management and in all those sectors that require a cheap, easy, flexible and fast way to unequivocally identify a product.

The first patent that lead to the creation of modern bar code was filed before the 50s by Woodland and Silver[63] but the first commercial application was only 17 years later and the industrial adoption only started in 1981 at the "United States Department of Defense".

The initial versions of the barcode are defined *linear* or *1-D* since they are able to store information using different line widths and spaces (Figure 1.1a). Some new formats have been defined later on: exploiting two dimensions *2-D* is possible to store more data and add some additional features to a bar code like encryption and error correction. An example of this evolution are the QR codes (Figure 1.1b).

DO NOT REMOVE

* G S J - 2 2 0 0 9 7 *

(a) 1-D (Code 39)

(b) 2-D (QR code)

Figure 1.1: Examples of barcodes

During the years the barcode readers have been implemented in different ways that can be classified as:

- Pentype scanners: the user needs to "scan" the code moving the pen (made of a light source and a photodiode) from one side to the other at a uniform speed; the lines and spaces are decoded like Morse codes and decoded
- Laser scanners use a laser as a light source and a rotating prism to scan the whole code, in the field of view at once
- CCD or led readers use the ambient light to get a "photo" of the code
- Camera based readers are the natural evolution of the previous point: using a normal camera is possible through image elaboration algorithms to decipher the code. These readers can be modified to recognize more tags at once: "large field of view readers" and "omnidirectional readers" are two examples.

1.2 Object optical recognition

The use of visual recognition methodologies has been made possible in the last years by the increase of the computing power. The first algorithms developed in this field allowed to identify the borders of the objects in an image. They have been used for OCR (Optical Character Recognition) to convert an image of text back to editable text. Some examples of edge detection methodologies can be found back in the 80s and an entire literature exists about them (some examples are [10] ,[39] and [37]).

Modern optical recognition systems are able to identify and classify the subject(s) in an image: training a neural network (Convolutional Network with residuals [21]) with a big database of images and classes of objects makes possible to achieve a good precision. Some online cloud services are available and ready to use in these cases; some examples are *Google Vision*[19] and *Amazon Rekognition*[3].

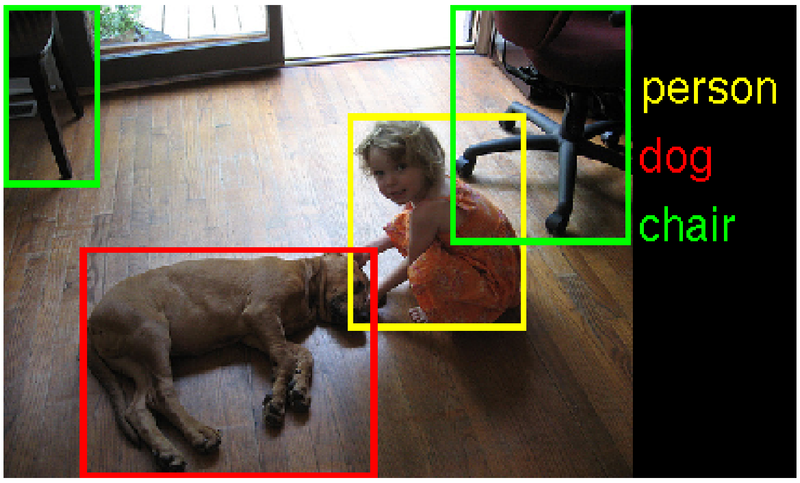


Figure 1.2: A processed image from the Image-Net database

1.3 “Color based” solutions

Different identification solutions have been developed using colors in the visible or non visible spectrum. Some of this systems are used to distinguish between real and counterfeit bills: the authentic bills are marked with inks sensible to the ultraviolet or infrared light which are not visible normally or have a different color.

These special inks are available on the market and, even if they are more expensive than “normal” colors, they can be a affordable solution to mark a product.

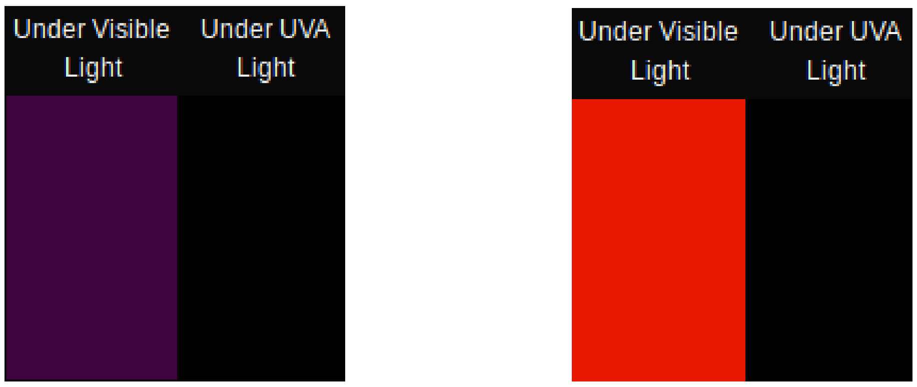


Figure 1.3: Examples of inks with different colors under visible and UV light

Chapter 2

Radio technologies

Radio technologies employ electromagnetic waves to share information between two or more devices without the need of a physical connection or direct line of view. Some of the values used to understand the distance between two devices are:

- RSSI (Received Signal Strength Indication)
- Phase shift between the transmitted and the received signals (only for passive, not powered receivers)
- ToF (Time of Flight) the time required for the signal to travel from the transmitter to the receiver

This values can be obtained from multiple sources (antennas, readers, receivers depending on the technology) and combined to further enhance the precision of the system making it more interference resilient.

2.1 Bluetooth based

The development of Bluetooth started in 1989 at **Ericsson Mobile** as a “short link” radio technology. One Bluetooth master can communicate with up to seven slaves and the maximum distance between two devices increased at each protocol evolution up to 240 meters for Bluetooth 5.0. The current Bluetooth 4.0 maximum range is around 60 meters making it suitable for object distance/position tracking. One of the problems of Bluetooth based devices, power consumption, has been addressed starting in 2004 [22] and has been integrated with the 4.0 specification during 2011 [5], making it suitable for low power, small battery powered devices.

One of the products based on BLE (Bluetooth Low Energy) and suitable for the scenario of this thesis are the *iBeacon* devices. The protocol has been presented by Apple during 2013 and has been implemented by different manufactures and every BLE smartphone is able to support it using a specific app. Each “beacon” has a major identification number, and some minor identification numbers to unequivocally identify it. It is possible, for a master, to filter and consider only some of the devices in range using the major ID number. For distance estimation the RSSI is used and a lot of libraries are available for the various platforms, both mobile and not mobile.

For person location identification it is possible to exploit already existent devices as "beacon" like smartphones. A modern smartphone already has all the hardware required and only need a specific library to behave like a standalone beacon.

This technology has already been used for indoor localization inside hospitals, shopping centers and in general big buildings.

2.2 WiFi based

Different protocols and implementations are part of the "WiFi" technology which is a set of specification also known as **IEEE 802.11**. Starting from 1997 with its initial release, the 802.11 protocol has been constantly updated to be faster, to increase the range of operation and be able to work at different frequencies.

The original aim of a WiFi device is to transmit and receive data but can be used for position estimating. Most of the major techniques are based on RSSI and employ one to many WiFi antennas. Some examples are:

- Trilateration: using the distance, obtained from the RSSI, of multiple access points is possible to find the approximate position of a device [65].
- AoA (Angle of Arrival): using an array of antennas the angle of the object with respect to the array is computed and then combining the information from multiple arrays is possible to find the position of an object. This method is suitable for high accuracy in an indoor environment affected by signal multipaths [30].
- ToF (Time of Flight) uses a completely different approach and exploit high precision timers (nanoseconds) to compute the difference between the time at which the transmitter sends and the time a response is received. Since the signals travel at almost the speed of light it is possible to compute the distance between the two devices with an accuracy of 2~3 meters [33].

2.3 RFID

RFID (Radio Frequency IDentification) is a family of technologies which uses electromagnetic waves and microwaves to identify objects with a special tag on or inside them. The previous analyzed WiFi and Bluetooth are part of this category but due to their radically different frequencies and initial purposes have been separated in this analysis.

RFID technologies can be divided in different classes by their tag type, active (battery powered) and passive (power taken from the waves emitted by the reader) or by their frequencies. Performances, cost, maximum distance and maximum stored data depends on those two factors. Active tags have an higher transfer rate, greater transmit distance but are bulkier, more expensive and need a battery with respect to passive tags which can be really small (the size of a fingernail or the depth of a sheet of paper), don't need a power supply and have a minimum cost of few cents.

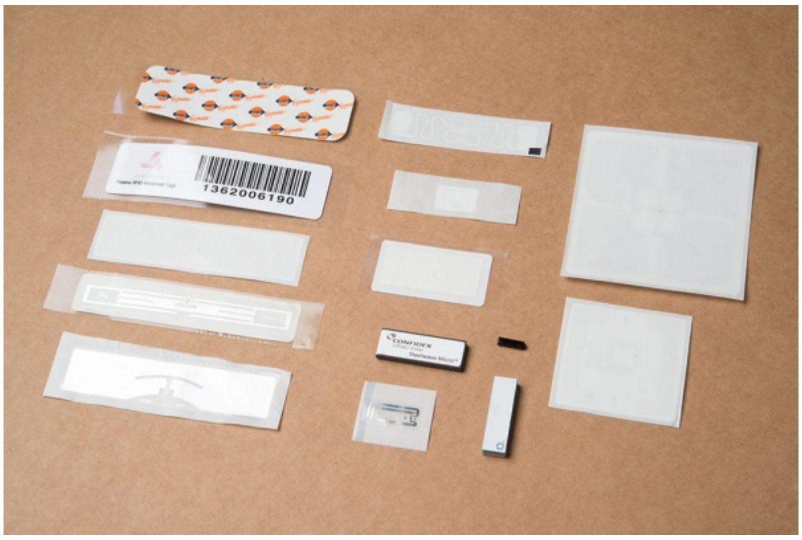


Figure 2.1: Some different kinds of passive tags

The frequencies at which the RFID family works are:

- LF (Low Frequency) with a range of some centimeters and low data speed
- HF (High Frequency) with a range of up to 1 meter
- UHF (Ultra High Frequency) with a range of some meters, up to 10~12 meters depending on the current regulations

The characteristics of the UHF tags and readers may vary depending on the region due to different regulations that allow different frequencies: in Europe 865.7~867.50 MHz and in North America 902~928 MHz.

A little mention here about the NFC which is strictly related with the other RFID protocols but allows a bidirectional communication at low distance.

The typical system is composed of a reader, from one up to some dozens of antennas and from one up to thousands of tags. The reader can be both handled or fixed but the performances are really different: the fixed version allows more antennas, has better signal stability, better range, higher reading speed and the antenna(s) can be chosen depending on the use case. The antennas for UHF readers can be "Near-Field" and "Far-Field". The two kinds can be connected to any UHF reader, so the type of the antenna is transparent to the reader. They have different ranges when reading the same kind of tag using the same transmitting power: some centimeters for the former and some meters for the latter. The comparison between the implementation and characteristics of the two antennas can be found in «UHF RFID Reader Antenna for Near-Field and Far-Field Operations»[53]. "Near-Field" antennas where initially created to identify near objects in the presence of metals and liquids that impact the reading capabilities of a "Far-Field" antenna. In recent years some "metal mountable" tags has been developed and are available on the market. This tags are usually rugged and can be inserted into metal objects with little to none impact on readability.

Chapter 3

Usability analysis

3.1 Functional requirements on the technology

Before starting a comparison between the previous mentioned technologies, it is better to identify which are the characteristics of an ideal solution.

The system should work in the following situation:

1. Real indoor environment with people and object in movement
2. A readable 3-D space with a surface up to some tens of meters and some meters (4 or less) from the floor to the highest readable position; dividing the space in zones is also doable
3. All the objects are traceable and can be identified inside the aforementioned space/zone
4. The number of objects can be in the order of some thousands
5. After buying, the object can be marked as sold and the system should ignore it

Due to the high number of objects the solution should have as requirement the lowest possible cost for each tag (if any). Each tag should have a long operative time without the need of substituting itself or a battery, if any. The tag should not be to invasive. The tag should be available in very high quantities and in various formats depending on the object it is going to be applied on.

Due to the wide space/room of a normal factory or shop the reader should have the highest possible range or at least the possibility to divide the room in zones, each with an independent reader without the risk of creating conflict or interference among each other. Additionally, since in a real case scenario the environment is quite noisy (lots of moving parts, possibility of electronic devices), the technology should be noise-resilient to some extent. An high number of tags should be able to be read by the same reader.

The chosen technology should be fast enough to get meaningful data: an user-object interaction in a shop can last for some seconds (picking, moving) and in that time frame the system should be able to get the data necessary for the analysis. The processing of the data can be done offline.

3.2 Reasons behind the choice of RFID UHF

The technology that best fits the requirements is RFID UHF with passive tags. An UHF tag can cost between some cents to some euros depending on its size, format and performances. For the scenario even the cheapest tags, under one dollar per piece, are usable.

A fixed reader can cost up to 2000$ but it supports up to 32 antennas, each on of them positionable in a zone with a radius of some meters. Multiple readers can be used without conflicts between them and the result is a tag read by each reader in range: a central database should be able to filter out the “conflict”. The tag of a sold object can be “killed” so it will not be recognized anymore by the system (proper authentication is required for this step).

Some tags can be protected with a password or their information encrypted making them relatively secure from cloning and tampering.

The visual methods have been discarded mainly because of the need of a free line of view between a reader and the object: this is clearly difficult to achieve in a shop with shelves and the constant presence of people moving to/from the shelves.

The Bluetooth solution is interesting but the high cost for a single device makes it unsuitable for object tracking and more usable if applied to people which usually have another wearable device (like a smartphone or smartband). The need of a battery is another demerit point for Bluetooth and for WiFi which is affected by the same problems. WiFi could have been an interesting choice due to the high range and its presence in the environment for other purposes (internal WiFi or free WiFi access).

Technology	Range	Speed	Active tags	Cost per tag	Receiver cost	Notes
Visual (barcode, QR-code)	Line of sight (a few centimeters)	N/A	No	Printing cost	Standalone reader 100 $ ~1400 $	–
Bluetooth	1-2 m (iBeacon 10 m, up to 70m)	High	Yes	25 $ down to a few $ for high quantities	25 $ or a device with BLE	–
WiFi	100 m	High	Yes	Device with embedded Wifi	Hotspot cost (10 $ ~few hundreds $)	Present hotspots can be used, max clients 10~300
RFID(LF)	10 cm	Low	–	1 $	100 $ - 300 $	Tags read by proximity, one tag per read
RFID(HF)	Up to 1 m	Low	Both active passive	0.50 $ to 5 $	600 $ - 800 $	The higher the number of tags the higher the reading time
RFID(UHF)	Up to 12 m (EU frequencies)	Mod.	Both active and passive	0.15 $	800 $ - 2000 $	Same as Rfid(HF) with lower reading time

Table 3.1: Comparison of the various technologies

Part II

RFID UHF

Chapter 4

RFID UHF protocol

4.1 Introduction

The latest RFID UHF protocol is called EPC Gen-2 and it is actually at version 2.0[16]. The protocol defines the physical and logical requirements that a tag and a reader should follow. The system is of defined "Interrogator-Talks-First" (ITF): the tags receive both the commands/information and operating energy from the reader.

The protocol describes the communication as :

> An Interrogator receives information from a Tag by transmitting a continuous-wave (CW) RF signal to the Tag; the Tag responds by modulating the reflection coefficient of its antenna, thereby backscattering an information signal to the Interrogator. The system is ITF, meaning that a Tag modulates its antenna reflection coefficient with an information signal only after being directed to do so by an Interrogator. ([16])

4.2 Communication

The communication is half-duplex so if the reader is talking the tag is listening and vice-versa. Only one tag at a time can answer to the reader, and before exchanging information all the tags need to be identified. The identification is divided in three key parts:

1. **Select** Some of the tags in range have to be selected. Each group of tags are defined a population. A population is defined by common data or common authentication shared among the members.

2. **Inventory** Among the selected population each tag needs to be identified separately. A *Query* command is sent to the population, one or more tags reply but only one tag is correctly identified at each session. The ID of each tag is called EPC. The process is iterated until all the tags in the population are identified.

3. **Access** This phase is done only if requested by the system: it is possible to read or write at a particular memory address of a tag, authenticate, block or kill a tag.

After each packet of data, sent or received, from/to the reader an error detection code is inserted. The length of the error detection code varies with the length of the packet but a CRC algorithm is always employed. Two CRC algorithms are used by the protocol: CRC-16 and CRC-5. In both cases the receiver should always check the integrity of the received packet using the appended CRC.

No error correction mechanism is expected by the protocol.

The collisions caused by multiple answering tags are managed by a random-slotted probabilistic algorithm. Even if the protocol doesn't specify a particular algorithm for collision resolution usually a "slotted aloha" based one is used. Lots of studies has been done about possible improvements of the original algorithm with the intent of keeping a constant inventory time regardless the number of tags. Some of them are:

- Bayesian estimation in dynamic framed slotted ALOHA algorithm [57]
- Optimal dynamic frame slotted aloha (DGFSA) [42]
- Group improved enhanced dynamic frame slotted ALOHA (GroupIEDFSA) [60]

A comparison between these algorithms can be seen in chapter 5.

Each tag should implement a random/pseudo-random number generator to be used for the slotted ALOHA algorithm. The specification forces some constraints on the characteristics of the generator:

- Each obtained number should have a very specific range of possible probabilities
- For a population of 10.000 tags the probability that two or more tags generate the same number should be less than 0.1%
- The second random number obtained after 10 milliseconds since the first one should be less predictable with a probability less than 0.025%.

Each tag should have enough memory to store two 16-bit random numbers.

4.3 Security features

Some security features based on passwords are implemented on the tags. These features may or may not be present on a tag but exists the possibility to query a tag about its capabilities.

4.3.1 Lockable fields

Some bits allow to lock some memory sections with an access password. The lockable regions are:

- Kill password
- EPC

- Access password
- TID memory

The *Lock* command should always be implemented in all the tags. Each memory region has two lock bits: one which blocks read or read/write depending on the region and one which makes the locks permanent. If a permanent lock bit is asserted it cannot be de-asserted later.

4.3.2 Killable tags

Some tags can be killed so to not respond anymore to the reader requests. Not all tags have these feature but only those with the K bit. These tags have a reserved memory which stores a kill password: when the correct command which the correct password is send to a tag, the tag acknowledge the success and then stops responding. The killed status is permanent, even after subsequent power-ups, and forces the tag to not react to the reader commands.

Not killable tags have the memory reserved to the kill password hard-wired to all zeros and it must be always accessible.

4.3.3 Restrict access

A tag can have an access password. This feature is optional.

Using the *Access* command

The access to a protected memory can be done only after the *Access* command has been sent together with the correct password: the tag enters in **secured** status and the reader can read/write not permanently locked memory regions.

If the wrong password is issued the tags reset the process and cannot answer to successive *Access* commands until an internal security timeout has finished.

Using the *Authenticate* command

A further security enhancement is using the *Authenticate* command which uses a cryptographic suite. Each tag can implement none to a few cryptographic methods. The method used for the communication is agreed with the reader. The communication follows a flow defined by the specific cryptographic suite.

A cryptographic suite can enforce some more strict requirements to the random/pseudo-random number generator than those specified in section 4.2.

Chapter 5

Performances

As said before the time required to identify a tag increases as the number of tags increases. In Figure 5.1 can be seen how the reading time scales with different algorithms with a different number of tags.

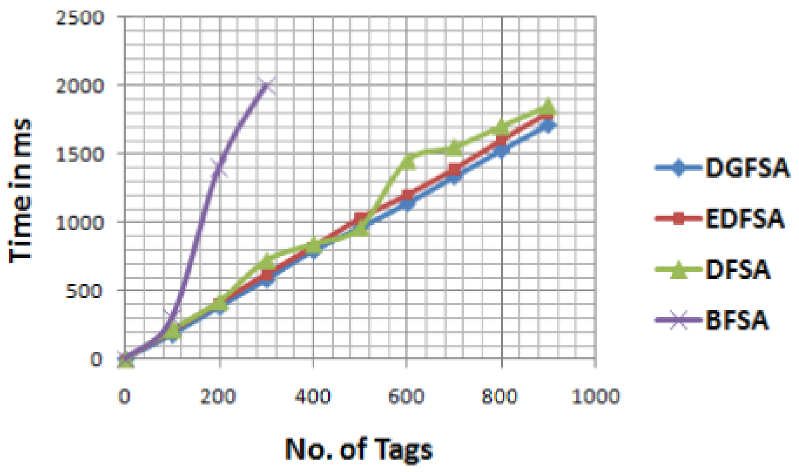


Figure 5.1: Comparison between collision algorithms. On the x-axis the number of tags, on the y-axis the time required to identify all the tags. See [42]

For the required scenario the time between the readings of each tag should be around 1 second or less to be sensible to short interactions (brief touch) when the object return to the initial position. In a real case a potential buyer usually takes more than a couple of seconds (the time needed to identify 900 tags) to take an object, observe it and then decide, so and extended reading frame should not cause any misreading.

The reading time in the scenario has been reduced to a minimum because the third step of the communication, **Access** is not required.

The reader can be set to only give back the data required. The only information read from the tag is its EPC for identification purposes, all the other data is computed by the reader. To summarize for each tag are needed:

- The EPC code
- The RSSI (in dBm): the strength of the backscatter signal received by the reader
- The phase difference (in radians) between the reader signal and the backscatter signal
- The frequency (in MHz) of the RF signal at which the previous two values has been taken

In some cases has been observed that a tag fails to be read at a particular inventory session without a significant environment change since the previous session. This can lead to possible mistakes during the processing phase if not properly managed.

Since the data can be elaborated with some delay or offline, the timings and throughput of transferring the measurements from the reader to the PC are not important as long as the communication is reliable. For the scenario a fixed reader, Speedway Revolution R420, will be used which is connected to a PC through an Ethernet cable.

Another important point of interest is the stability of measurements: even if a tag is not read at each inventory phase, between successive measurements with a stationary situation (the tag has not been moved) the data provided by the reader should be constant as much as possible.

For the fixed reader this is mostly true: the standard deviation of the phases at each frequency is almost always zero or a very low value, the standard deviation of the RSSI is usually a very low value.

The RSSI measurements are more subject to interference, multi-paths and other materials between the tag and the reader than phase difference. This fact has been observed in different papers about distance measurements exploiting UHF RFID tags. As an example in Figure 5.2 can be seen the absolute error distribution of the distance tag-reader computed using RSSI and phase difference.

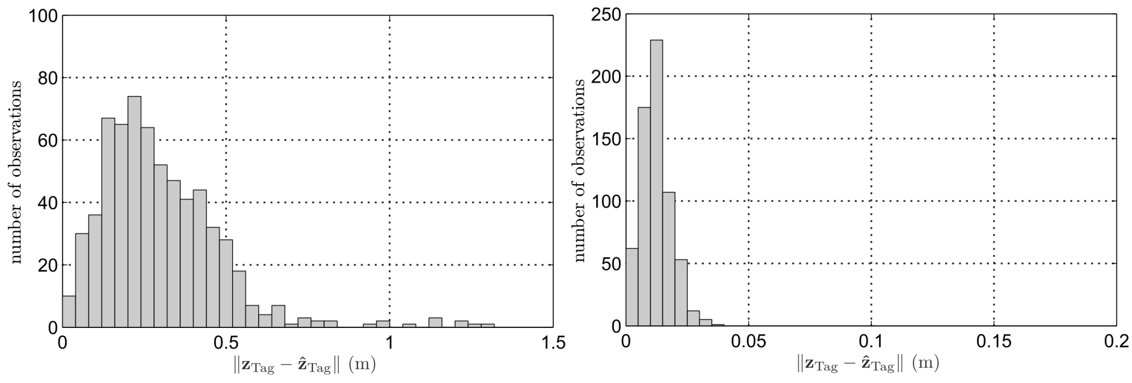


Figure 5.2: Distribution errors of distances estimation using RSSI (on the left) and phase difference (on the right). Taken from [49]

The performances of an handled portable reader are not considered here but are discussed later on.

5.1 Interference effects

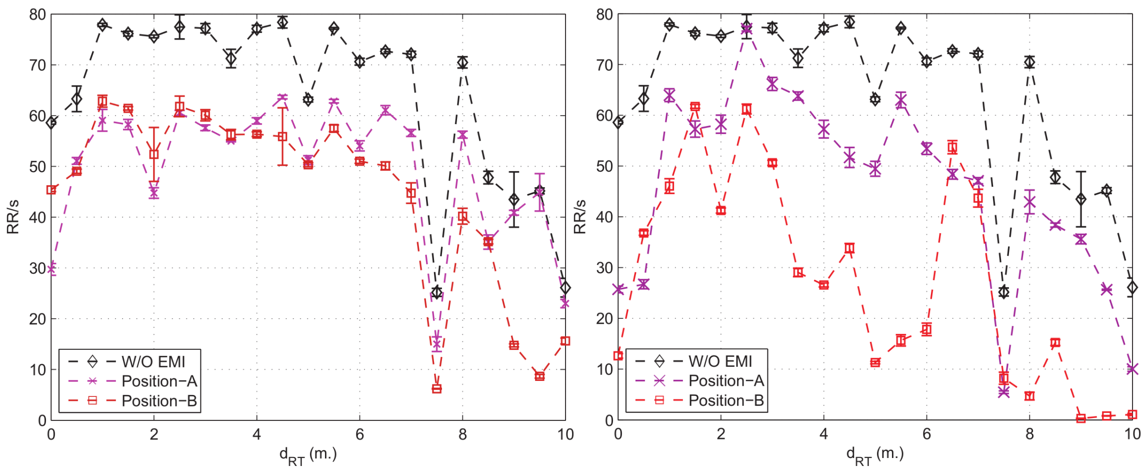
A very important factor that can heavily influence the performances of the system is environmental interference.

5.1.1 Electromagnetic interference

The effects of various kinds of electromagnetic interference has been analyzed in [2]. The reading rate (reads per seconds) has been used as the measure for the performances.

The first test has been done using an impulsive interference with a frequency similar to that of the RFID system. The source of the interference has been moved to two location near (Position-A) and far from the antenna (Position-B) Figure 5.3a. The reading rate decreases of more than 40 % at some tag-reader distances like 2 meters.

The impact is greater when using a Gaussian noise and depends on the distance between the noise source and the antenna. The performance loss is more relevant when the noise source is far from the antenna where the antenna power is quite low Figure 5.3b: for greater distances the reading rate falls to very low values so making the tags mostly unreadable.



(a) Impulsive interference (b) Gaussian interference

Figure 5.3: Reading rate (RR/s) on the y-axis, tag-reader distance on the x-axis. Impulsive and Gaussian noises. Taken from [2]

The performances are even worse when dealing with a Rayleigh interference Figure 5.4: for tag-reader distances greater than 7 meters the reading rate falls below 10 RR/s.

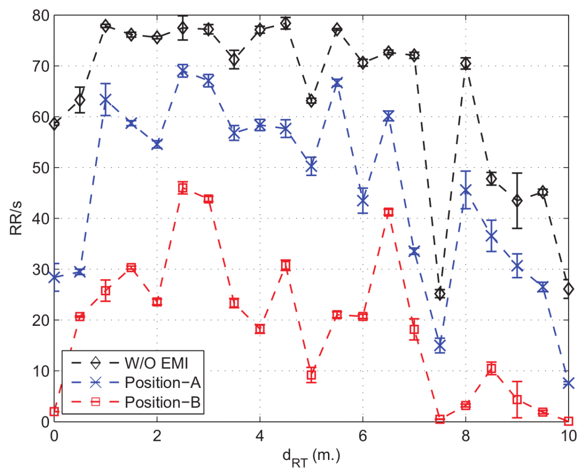


Figure 5.4: Reading rate (RR/s) on the y-axis, tag-reader distance on the x-axis. Rayleigh noise. Taken from [2]

The effect of electromagnetic interference is more evident with high tag-antenna and antenna-noise source distances. For the intended scenario of this thesis the distances tags-antenna are usually lower than 4/5 meters when considering antennas in the shelves so the effects of interference is mitigated.

5.1.2 Metal and liquid interference

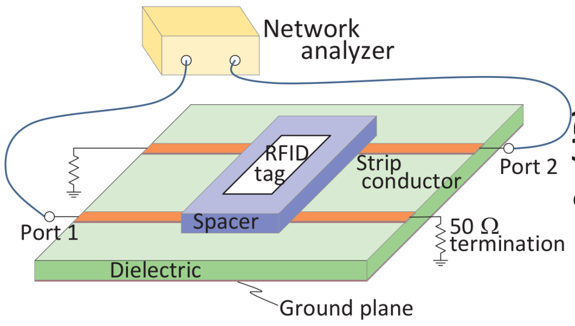
The performances of in an environment with liquids and metal object has been analyzed in [47]. The tests were run using Asian/North American frequencies 902-928 MHz.

A plastic water container and various metals has been used for the experiment including aluminum, copper, iron, nickel. The test has been conducted in the near region, tag-antenna distance less than 90 centimeters, and in the far region, tag-antenna distance around 1.5 meters, and showed that there is no particular impact on the reading rate.

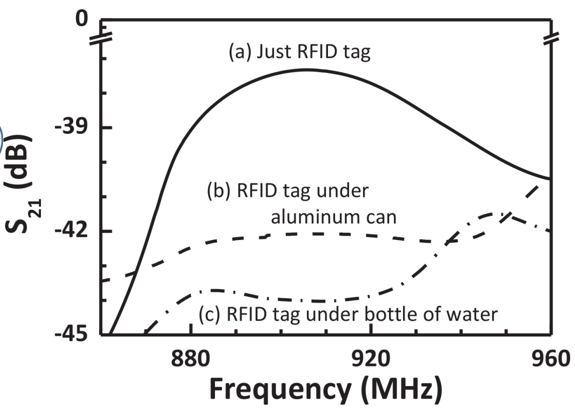
This results are encouraging for the purpose of this thesis in which mounting the antennas on shelves made of metal sheets should be doable.

During my tests I have noticed that the thickness of the metal sheet can indeed reduce the readability of a tag, making it unreadable if multiple sheets are one on the top of each other with a thickness of more than 2~4 millimeters.

Another paper [20] exploited the interference caused by metallic and liquid materials to identify the presence of an object. The position of the elements is different from [47] since in that case the metal/liquid was between the tag and the antenna and in this case the metal/liquid is “behind” the tag as showed in Figure 5.5a.



(a) Antenna structure for the experiment



(b) RSSI with respect to frequency with different object

Figure 5.5: Antenna and interference of different objects.
Taken from [20]

The results in Figure 5.5b show the effects of positioning metal objects or liquids on top of the tag.

The RSSI of the tag is impacted by the presence and type of object. The system is suitable for stationary situation recognition (presence/not presence) since the interference caused by the object is replicable and depends on the specific communication frequency. This kind of interference causes an attenuation of the RSSI which only reduces the reading range and should not compromise the possibility of using the RFID for the purpose of this thesis.

Chapter 6

Regulations

Some constraints apply due to regional regulation. In this chapter only the European specification, ETSI EN 302 208 [17], is considered. The document imposes some constraints that are also reported on the Speedway R420 manual [55].

6.1 Frequency

The regulation specify not only the frequency range, but how the channels should be separated inside the range, the stability, the spurious emissions and the error in the transmitted frequency (sections 4.3.1, 4.3.2, 4.3.6 of [55]). Since the used reader has been certified the only important part is the frequencies and the channels.

For the use case having different phases at different frequencies can be useful to separate actual signal variation from interference. The reader supports all the four European channels from 865.7 to 867.5 MHz and they will be used in all the experiments (frequency hopping).

6.2 Timing

The reader should not “listen before talk”: an already powered tag cannot be read by a different reader which hasn’t before issued a command to it. The reader should always open the communication (section 4.3.7 of [17]).

The reader cannot occupy the channel for longer than a time frame of four seconds, so it is forced off for 100 milliseconds before reusing the same channel. The reader can stay on as long as it changes the transmitting channel. During the experiments has been observed that when frequency hopping the data sent to the PC is slower than when using a single frequency; this is probably due to the time required to re-set the transmitting hardware to the new frequency.

6.3 Antenna

Some constraints are specified for the antennas related to the beam-width (Section 4.3.4 of [17]).

For all the experiments, commercial available antennas have been used so no further checks are needed for this part.

6.4 Power

The total transmitted power should never exceed 33 dBm ERP (Effective Radiated Power) as specified in section 4.3.3 of [17]. The reader manual [55] suggested formula to compute the transmitting power is :

$$Max_{power}[dBm] = 33[dBm] - Gain_{Antenna}[dBd] + Loss_{Cable}[dB]$$

The used antennas for the experiments are **Laird Technologies' S8658PCx** with a gain of 3.85 dBd and assuming a cable loss of 2 dB the maximum allowed power is 31.5 dBm. Since 31.5 dBm is the maximum output power of the reader (constrained to 30 dBm in Europe) the reader can be set to its maximum output power.

During the experiments, in some cases, the power of the reader has been reduced: in the presence of metals between tag and antenna or if a tag is too near the antenna, reducing the power resulted in better reading speeds. For distances ≥ 1 m, keeping the maximum transmitting power gave faster and more reliable readings (less spurious unread tags during inventory). This fact has been observed in Figure 5.3, black line: the reading rate decreases when the tag is too near the antenna.

Part III

Implementation

Chapter 7

Introduction

The first part of the implementation (chapter 8) has been occupied by the study of the Speedway R420 SDK called "Octane" and exploring the capabilities of it. The Octane SDK is the PC side but exists the possibility of using the embedded capabilities of the reader. The embedded development makes use of the ARM processor, the FPGA inside the Speedway reader, the Linux system and gives access to the low level raw data.

For the case scenario I have only used "Octane" (Java version) to get the data from the reader. Part of the initial processing of the data (like grouping values by EPC) has been inserted in the Java program together with the SDK (section 15.6). The rest of the logic has been implemented in C++ and compiled GCC 7.1.1 exploiting its auto-vectorization features [4] to get an optimized binary (subsection 13.3.2).

The tests than moved to an handled device a "CAEN R1240I qID". Due to the noisy output of this reader I have tried various filtering techniques from the simplest one, an exponential moving average filter, to a median windowed filter ending with a little neural network as non linear filter.

The next step has been identification: identify an object with a tag confronting the tag's code with a database of products. A fully functional "smart table" has been developed, able to identify an object and to show the related information on a screen. In this step the problem of "tags not always read" has surfaced and has been solved for that particular case.

A further attempt has been made to compute the approximated position of the object on the table exploiting the antenna characteristics.

The next step has been storing samples of the various human-tag interactions in different situations like:

- various antenna position
- various antennas models, different antennas same model
- various tags position
- single tag in range vs variable number of tags in range

The longest step has been finding a way to process the raw values in order to obtain meaningful data and then choosing the correct neural network parameters. In this step are included all the improvements and additions inserted in the network in order to have a good precision in the output data.

Particular attention should be payed on the choice of the outputs values due to the range limitation of a neural network and then how to interpret them.

Chapter 8

Octane SDK

The reader can be connected to a PC through a serial or an Ethernet cable. The serial interface is mainly for debugging purposes and in the case the reader is not reachable anymore, if it doesn't answer when connected to the network. The reader has a local DNS resolution service which allows connecting to it without the need of knowing its IP address. The resolution is quite slow and due to the many tests resulting in continuous connections and disconnection to the reader, a static IP address was set. The reader-PC connection can be remote: during the various tests, with the two sides across the Internet, there were no particular visible lags which caused malfunctions. In the following section only the actual used functionalities are reported. The whole documentation can be found at [44].

8.1 Basic settings

The first setting is the IP address or the hostname of the reader on the network. It is possible to scan and get a list of all the readers on the LAN but for the scenario required is more convenient to keep a list of the readers addresses. In all the experiments a single reader has been used.

8.2 Antennas settings

In the SDK all the antennas settings are grouped in class **AntennaConfigGroup**. Through it is possible to obtain the list of effectively connected antennas and set various parameters:

- Receiving sensitivity
- Whether or not to use the maximum transmitting power
- Specify a transmitting power in dBm (the list of possible values can be downloaded from the reader)

8.3 Report settings

In this section are included all the data that the reader should compute, store and then send back to the PC. If a particular value is needed, it is required to set the reader before starting reading the tags. The possible given values for each tag are:

- First seen time in milliseconds
- Last seen time in milliseconds
- Seen count
- Peak RSSI
- Phase angle
- Channel used in MHz
- Number of the antenna that did the read

Some other values can be obtained but were not supported by the particular version of the onboard firmware like Doppler frequency.

8.4 Start the readings

After sending the settings to the reader some more steps are required. The SDK is fully asynchronous so it exposes some “Listeners” to call when an inventory session is concluded. Only one listener can be associated at a time which receives the list of read tags after the whole inventory session is finished. After issuing the start command the reader starts interrogating for tags in range. The list of tags given to the listener contains all the data set as enabled in the report settings. The more data is required the more time is required to conclude the inventory phase and send back to the PC.

8.5 Extracting the data

Each tag read creates an instance of the *Tag* class in the list given to the listener. Each value requested is usually a primitive type (double, string) which needs to be extracted and stored separately for processing. The EPC code of each tag is given as a hexadecimal string which can be converted to a number. The RSSI, phase measurements and channel are given as double precision numbers.

Some additional functions are given to check if a value is valid or not. All the values are always given but only those requested in the report section are really meaningful.

Chapter 9

CAEN reader

This chapter reports the various experiments with the handled CAEN reader. This reader will not be used anymore after this if not to study the repeatably of its readings.

The "CAEN R1240I qID" could have been an interesting solution due to its compact size, the fact that includes the antenna and its Bluetooth connection. An SDK is given for this reader in Java both for a PC and Android.

The initial connection has been quite problematic both using a USB cable connection with the PC and Bluetooth using the given application on Android. The example application is quite old (Android 2.3+) and works barely acceptable with a modern smartphone requirements; many attempts are needed to obtain a successful connection. After the connection the reader works flawless.

Using a PC both an USB and Bluetooth connection has been used but, due to the degraded performances with a not fully charged battery, the majority of test has been done using a wired connection.

The reader communicates using a serial through USB and is seen as a modem by the PC. The given SDK is in charge of sending, receiving and decoding the data from/to the reader.

9.1 Configuring the reader

Since the connection is done through serial, the serial port used needs to be given to the SDK. The other important settings are:

- The transmitting power (in mW)
- The regulation (in case of Europe is ETSI 302 208)
- Which tags to select and eventually enable a filter
- The tags filter settings
- The Q value for the identification algorithm (See [42])
- The "source" number or antenna number (the specific model has only one antenna)

- The session number (See below).

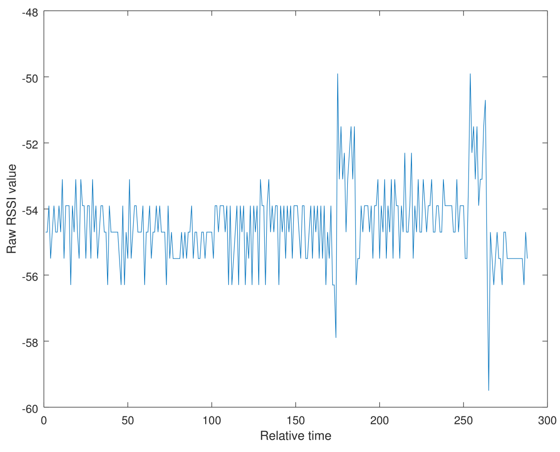
The session number is a particular feature in the reader that allows to not re-inventory tags recently inventoried. The aim of this mechanism is to reduce the inventory time due to the reduced number of tags to read assuming a mostly constant population of tags in range. The session are as follows:

- Session 0: Read always all the tags in range
- Session 1: Read the tags not already read in the previous 5 seconds
- Session 2 and 3: All the tags "older" than 5 seconds

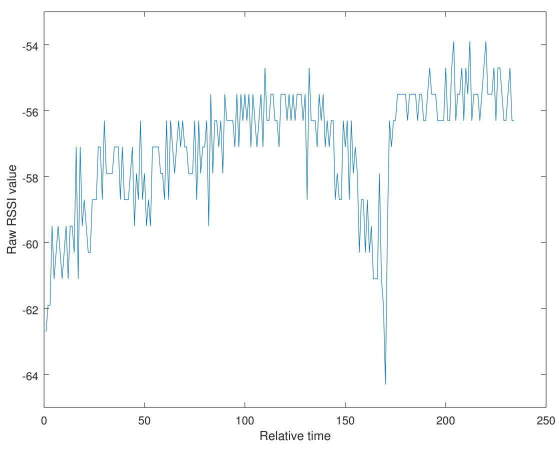
For the tests only session 0 has been enabled due the reduced number of tags in range and the need of checking the stability of the reader.

9.2 Performances

By default the reader gives only a very reduced set of data about a tag (only its EPC). The reader is quite fast or at least seems to be faster than the Speedway with a small number of tags and the default settings. The CAEN reader does not support phase measurements and the RSSI values given are really noisy.



(a) Still tag. Variance of data is 1.07



(b) Moving tag. Variance of data is 1.35

Figure 9.1: Examples of RSSI over time using the CAEN reader with a tag within 1 meter

For comparison the Speedway reader gives an RSSI with a variance of 0.4~0.5 and a phase with a variance of 0.1~0.2 for a still tag near the border of the reading range.

9.2.1 Filtering the data

The raw data obtained is characterized by lots of high frequency noise (spikes). To obtain a "softer" curve, filtering techniques can be used but they have too be as simple as possible due to the high quantity of data that need to be processed. The values for each tags should be stored and processed separately. I have selected two possible algorithms:

- Median filter: suitable to spikes removal
- Exponential filter: suitable to "smoothen" the data

Median filter

A median filter requires only one parameter: the window size. A bigger window gives smoother data but decreases the variance introduced by a real change (not noise) in the data. The median filter substitutes a value of the signal with the median of the window: as an example, when using a window of 3, a value in a time series is substituted with the median computed on a window made of the value, the previous and next ones.

The results on the Caen reader are as expected Figure 9.2. The falling spike between time 150 and 200 is more attenuated the more the window is increased, but the overall stability of the signal improves. This is reflected in the standard deviation of the whole signal which decreases from 1.86, for the unfiltered data, to 1.71 when using a window size of 2, to 1.46 when using a window size of 20.

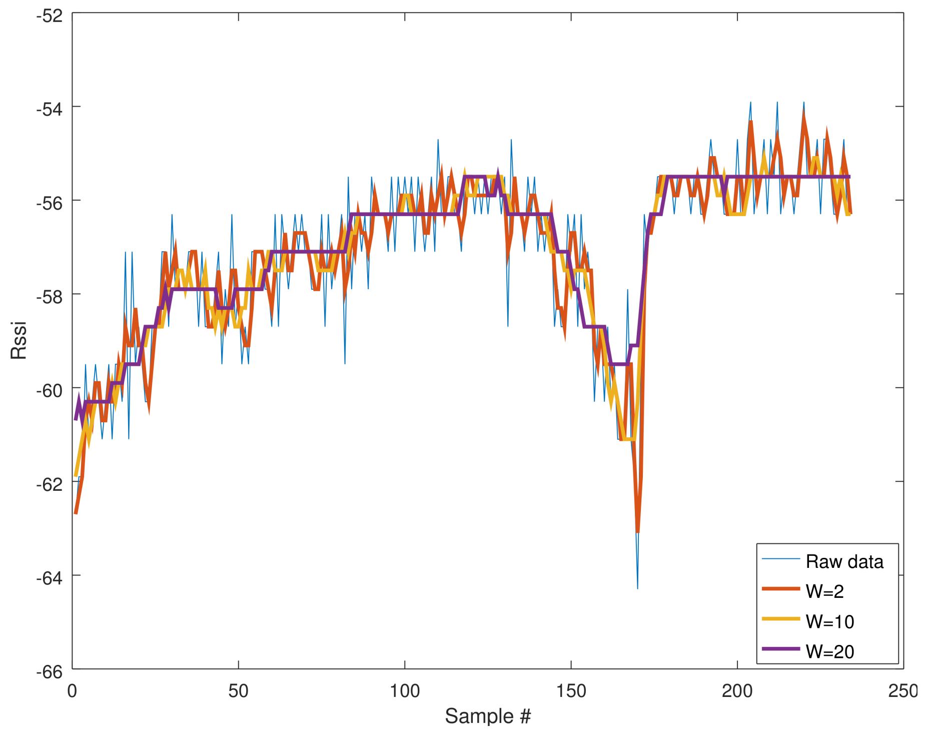


Figure 9.2: Median filtered data using different window sizes.

Exponential average filter

One of the other commonly used filter is exponential average: no window is present here and the only required values are:

- A coefficient between 0 and 1 (the only parameter)
- The previous average value
- The new raw value

The filtered value is obtained with:

$$Avg_i = \alpha \cdot x_i + (1 - \alpha) \cdot Avg_{i-1}$$

where α is the coefficient, x_i is the raw value and Avg_i is the average computed at x_i. It is common to put $Avg_0 = x_0$;

The α coefficient is the weight of the new value: a greater α gives an average more similar to the original signal (purple curve), a lower α a "smoother" signal (orange curve).

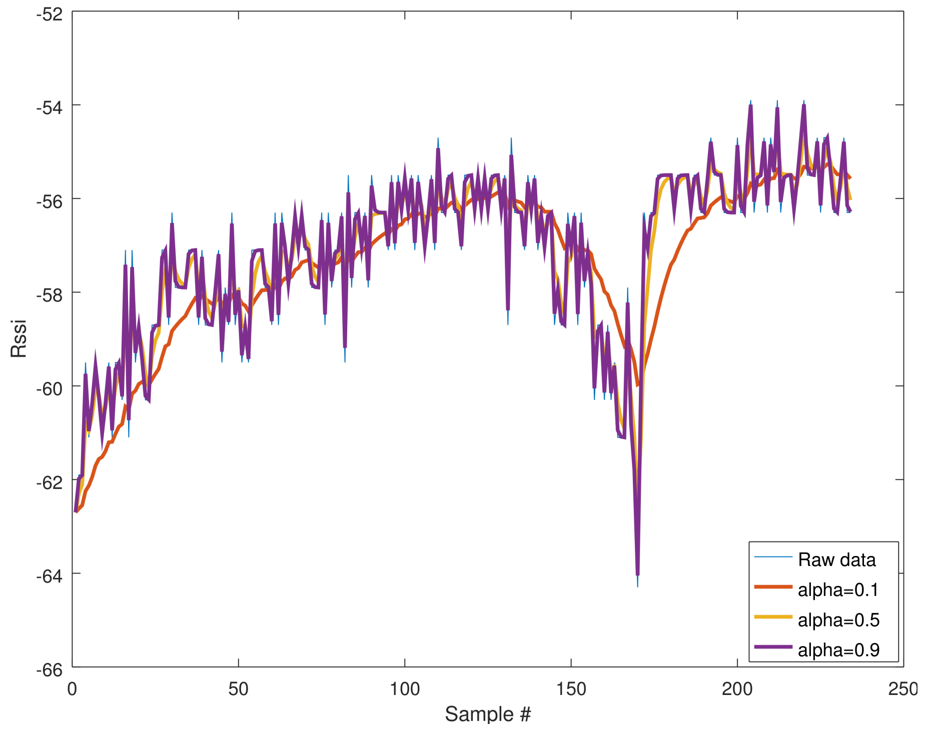


Figure 9.3: Exponential average as filter with different α

Chapter 10

Identifying objects

One of the important points for the case scenario is distinguishing objects from each other. Each tag has an identification code called EPC which in some cases is programmable. For this part is sufficient to consider the EPC code as the equivalent of a bar code.

A database of products keeps track of the equivalences between EPC code of a tag and the object it is attached to. The normal procedure is:

1. Take an object
2. Scan the object to read the tag
3. Get the tag's code
4. Look for the code in the database
5. Get the data about the object

Since any tag has a different code it necessary to program a group of tag, or request them already programmed from the factory, to have the same EPC code so a group of equal products end up with the same identification number.

The aim of this chapter is to talk about the realization of a "smart table" able to identify the object on top and to display the information related to it.

10.1 Requirements

Before going on with the implementation, it is better to formalize the list of requirements:

- The object placed on the table should be recognized regardless its position or orientation
- The object should be identified as fast as possible
- The non-presence of an object should be identified
- Only one object can be identified at a time

- The data about the objects should be easily editable
- Arbitrary data can be associated to the object
- The obtained data should have a very specific output format

The whole system is basically made of two parts: one is in charge of the previous requirements list and one is in charge of displaying, presenting the data to a potential customer. The part I have done and I write about here is the former. The last requirement is one of the most important due to the fact that the first part needs to be interfaced with a separate program done by a separate team.

10.2 Outline of the implementation

The main program has to interface directly with the SDK. The SDK is set to give only the EPC of the tag read. The main program pseudo code is:

```
function INIT(IP, TXPower, DBFile, OutFile)                 ▷ Entry point
    Connect to the reader
    Send settings to the reader
    Open DBFile
    Create OutFile
    Set the listener method to be called when the inventory has ended
    Start reader
end function
function MAINLOOP                                           ▷ Wait for user to exit
    while !KeyPressed do
    end while
    Disconnect reader
    Exit
end function
function LISTENER(TagsList)         ▷ Called after each successfully inventory
    Get read tag from TagsList
    Get read tag EPC
    Compare with the entries in DBFile
    if Entry found then
        Update OutFile                                      ▷ Read tag in database
    else
        Clear OutFile                                       ▷ No tag or unknown tag
    end if
end function
```

10.3 Workaround the problems of the technology

10.3.1 Spurious unread

As said before in chapter 5, the technology does not assure that a tag is read at each inventory phase, but in some random cases a tag in range could not be reported.

The solution to this issue has been solved using a time window: if a tag is read in the current time frame than it is considered as present on the table; when the time frame expires the object has to be considered as removed.

This straightforward solution results in quite a complex implementation caused by the asynchronous report given by the SDK. The report of an inventory phase does not arrive at specific intervals when the main program is running.

A separate class implements the access to the output file (**XmlGenerator**) and each read tag is directly passed to it. This class does most of the required management. The main logic does:

```
function NewTagFound($NewTag$)                  ▷ This function is called after a read tag
    if $NewTag \neq OldTag$ then                 ▷ The read tag is different than the previous
        Stop timer thread
        if $NewTag$ in Database then             ▷ Check if the tag is one of the registered
            $OldTag \leftarrow NewTag$
            Start timer thread
            Write($NewTag$)                      ▷ Write the output file
        else                                     ▷ Tag not recognized. Consider it as no tag in range
            $OldTag \leftarrow NoTag$
            Write($NoTag$)
        end if
    else                                         ▷ The same tag has been read. Reset the timeout timer
        Restart timer thread
    end if
end function
```

```
function TimerThread($Timeout$)   ▷ This thread behaves like a non-blocking timer
    Sleep for $Timeout$ seconds
    Write($NoTag$)
end function
```

Notice that, using this algorithm, the object recognition is immediate (as far as the technology allows) but absence of object is strictly dependent on the chosen timeout. The timeout is related to various environmental factors and needs some tries to find an acceptable value. Some of the factors are:

- Interference/noise levels

- Distance antenna/table surface
- Table characteristics (material, thickness)

With a wood table with a couple of centimeters of thickness, the antenna at 10 centimeters from the bottom of the table surface and in an office environment, one second is a safe value for the timeout, with 400~500 milliseconds as the lowest limit.

A timeout too short can cause the *NoTag* event even when an object is still on the table; a timeout too long makes the system to feel slow when an object is removed.

10.3.2 Antenna transmitting power

The used antenna is a far-field antenna with a range of around 10 meters which is too wide for the required application. The reader allows to reduce the transmitting power thus reducing the reading area. The ideal sensible zone is the up to 10 centimeters above the surface of the table. A more extended zone can cause misreadings with objects near the table but not on the surface. The antenna radiation pattern is usually a semi-sphere: the higher the transmitting power the higher is the radius of the semi-sphere. Through empirical tests the radiation pattern of the antenna keeps its shape even when passing through non metallic surfaces like a wood and/or glass table.

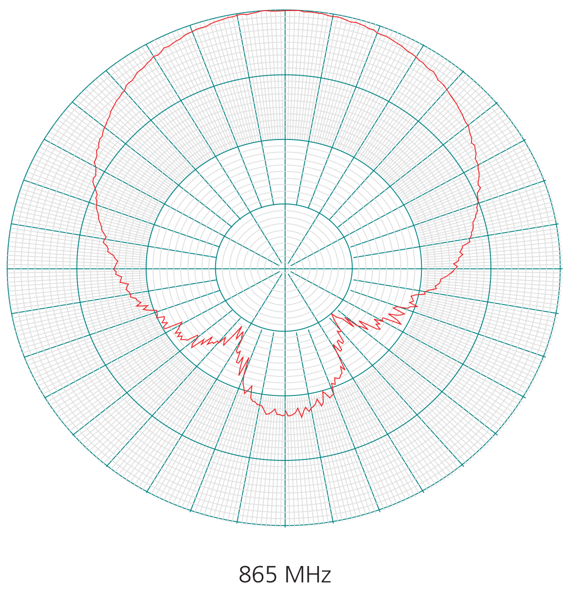


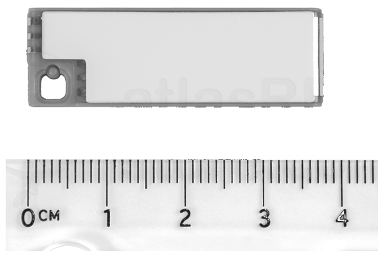
Figure 10.1: Radiation pattern of the antenna s8658p used. Notice the almost semi-spherical shape of the pattern above the antenna. The back lobe, below the central horizontal line, can be ignored for the use case.

To maximize the sensible surface of the table while keeping the output power low, the antenna can be inserted some centimeters below the surface. This is equivalent to consider only a portion of the radiation pattern: a good choice can be setting the antenna so to have only 2/3 of the shape above the table surface.

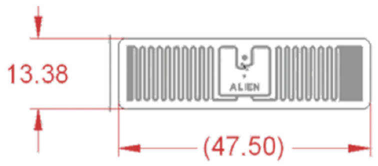
10.4 Choice of tags

Choosing a tag type implies doing some tests on site. Some tags have an higher sensibility allowing them to be read more easily than others. This kind of tags usually works best when the distance tag-reader is quite high, but combining a good choice of transmitting power on the reader (which reduces the range) with a good quality tag, makes possible to reduce the timeout so to obtain a more responsive system with less misreadings of far tags.

Rigid tags have usually better "readability" but have a much higher cost which is acceptable for limited quantities. Moreover a rigid tag can be removed from the object upon buying, re-programmed and reused for other products. Instead a flexible tag is not re-usable after placing. A rigid tag is suitable for large products, like clothes while a flexible one can be directly glued to small objects.



(a) Rigid re-usable tag. Cost is around 4 $ per piece.



(b) Flexible tag. Cost is 0.24 $ per piece with an order of 1000 pieces.

Figure 10.2: Comparison of different tags types with similar sizes

Chapter 11

Finding the position of an object on a surface

A lot of studies has been done to find the position of a tag only using the RFID technology capabilities. To identify the distance reader-tag a single antenna can be used [59, 38]. More antennas can be used to increase the precision in a real environment [36], or antenna arrays [31] and/or multiple reference tags [12] are possible solutions.

In this part I have tried to exploit the radiation pattern of the antenna in Figure 10.1 to identify the angle between the center of the table, which corresponds to the center of the antenna, and a tag on the table surface.

Using the diagram and assuming a constant linear distance between the tag and the antenna, it is possible to estimate the angle between the perpendicular at the antenna's center and the imaginary line that links the tag at the center of the antenna. Due to the fact that the distance tag-antenna is not constant across the surface some corrections are needed especially at the borders of the sensible area. Without the correction the error increases as the tag goes far from the center up to around 5 ~8 degrees.

11.1 Relation between RSSI and angle

The first step here is to extract the radiation pattern values from the picture in the antenna datasheet (Figure 10.1). This can be done using a digitilizer which gives back the coordinates of the points in the plot. Using the value extracted it is possible to plot back the data in an x-y Cartesian plot, and after a 90° rotation the plot in Figure 11.1 is obtained.

As can be seen the curve is not a mathematical function but it can be separated into four different curves which are functions and to which polynomial fitting can be applied in order to obtain analytical expressions.

The obtained analytical expressions allow to compute the angle aforementioned knowing the read RSSI and the maximum RSSI obtained: those expressions define the relations between attenuation and angle. The best fitting polynomial in this case is of degree four but a linear regression fitting gives good approximation. The highest value of RSSI is

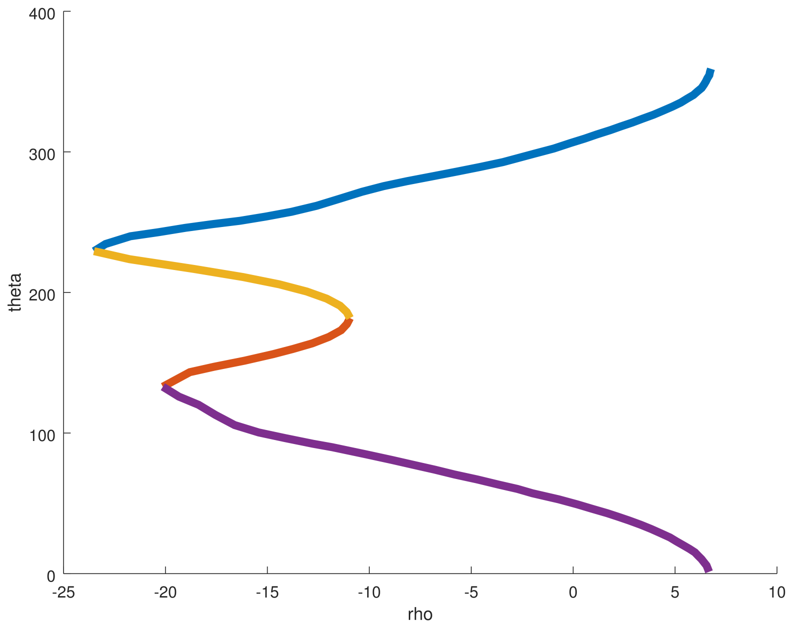


Figure 11.1: Radiation pattern of the antenna s8658p on a cartesian plot.

obtained at angle 0° or 360° (in Figure 11.1 are the highest point of the blue curve and the lowest of the purple one).

The plot should be symmetric at 180° but, as various experiments have proved, the antenna is not completely symmetric itself thus increasing the uncertain of the measurements.

At each RSSI value corresponds two to four possible angles that identify some circular zones (annulus) on the surface with center the center of the antenna.

11.1.1 Implementation

The final application needs some a priori data:

- The highest RSSI value at 0°
- The four equations corresponding to the four curves

The final angle can be the average of the two/four angles obtained. It should be possible to compute the uncertainty of the value combining the two/four angles if required.

11.1.2 Proposed improvements

The previous implementation is meant to be a "proof of concept". Lots of improvements are possible to increase the precision of the system and make it usable.

One interesting proposal is to compute the error introduced by the partially wrong initial assumption, constant tag-reader distance, to obtain the required correction.

Adding more antennas should be possible to pinpoint the real position of the tag on the surface, using already existing techniques like trilateration (intersections of circles).

11.2 Relation between RSSI, phase and tag position

The starting point of the previous section is the datasheet of the antenna. In this section I try to use not only RSSI, but phase too, to confirm the results obtained before using real data taken using the reader.

Instead of using angles, relative position will be used here. As in the previous section, one antenna is not enough to obtain the coordinates of a tag on a surface so only possible "zones" are given by the experiment configuration.

All the tests here have been done with an antenna below the surface of a table. A square of 1x1 meters has been drawn on the table with its center corresponding to the center of the antenna.

The measurements have been taken using 9 different tags positioned at known intervals inside the square. The RSSI has been shifted in order to have a value of 1 at the center of the antenna while the phase has been taken "as is" and due to the low tag-antenna distance ranges between -3 and 1.6.

Rssi and position

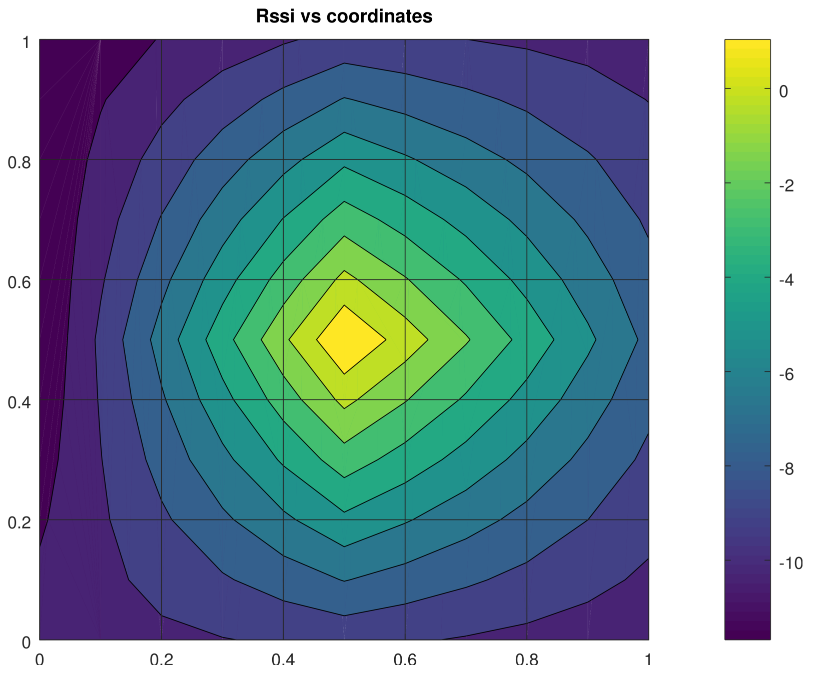


Figure 11.2: RSSI values measured on a surface

As can be seen in Figure 11.2, the not perfect symmetry of the antenna is confirmed as well as the shapes of the zones identified in the previous section.

Phase and position

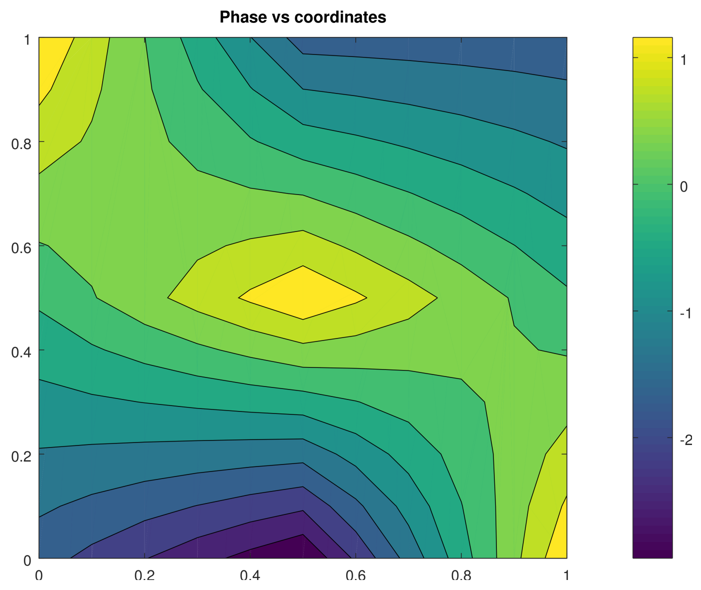


Figure 11.3: Phase values measured on a surface

The phase shows no particular shapes in Figure 11.3. Only one particular characteristic of phase measurements is showed: periodicity. The phase difference is periodic with respect to the wavelength: two tags at different distances from the antenna can have the same reported phase. The Speedway reader reports the phase difference in radiants and when considering the European frequencies, moving a tag 5 cm away or towards the antenna implies a phase change of around 1 rad. This means that after a movement of a distance equal to the wavelength, the tag has the same phase difference as when it was at the starting point. This causes ambiguity in the real tag-antenna distance (yellow zones in the picture are an example). The maximum unambiguous range is defined as

$$R = \frac{c}{2f}$$

where c is the speed of light and f is the transmitting frequency. Some techniques know as PDoA (Phase Domain of Arrival) extend the unambiguous range. One of the most affordable is the Frequency Domain PDoA which exploits multiple phase differences at multiple frequencies:

$$d = \frac{c}{4\pi} \cdot \frac{\Delta\theta}{\Delta f}$$

where d is the tag-antenna distance, $\Delta\theta$ is the difference between the phases and Δf is the difference between the frequencies. The unambiguous range formula becomes

$$R = \frac{c}{2\Delta f}$$

As an example a $\Delta f = 1MHz$ gives an unambiguous range of 150 meters which is much higher than the wavelength and the maximum range of the technology [48].

Chapter 12

Recognizing movement and touch

One important step toward the aim of this thesis is to understand how the values given by a reader change when a tag is moved.

Trying to find the position of an object at a specific time and then later on in order to identify a possible movement is possible but requires too much complexity, both in hardware and in software. A much more simple approach can be applied in this scenario: observing the changes in the values to indirectly observe changes of tag position.

A common computed number to measure the "diversity" of the values in a sample of a population is the variance or optionally the standard deviation.

In chapter 11 has been observed that small changes of position cause important changes in the measured RSSI and phases.

For the next experiment the transmitting frequency has been fixed to a single value and the reader has been left on for some time. A single tag has been kept still for some time and then suddenly moved: the idea was to get a plot with "steps" and observe if the environment noise could be separated from the actual movement.

Another important issue is how to compute the variance of a partially known population which changes at each sample. Three solutions are possible here:

- Use a sliding window and compute the variance on the window
- Use an online algorithm which can compute the variance knowing some of the previously computed values.
- A combination of the two

12.1 The window approach

For this choice it is necessary to keep in memory n values, where n is the size of the window, for RSSI and phase. The variance can be computed on the whole window using the classical two-pass algorithm:

1. Compute the mean of the window $\bar{x} = \frac{\sum_n x_i}{n}$

2. Compute the variance with $\sigma^2 = \frac{\sum_n (x_i - \bar{x})^2}{n-1}$

12.2 Online variance

Online variance does not require a two-pass computation on the values but can directly compute the mean and the variance with a single pass.

Various formulas have been studied in order to obtain numerical stability. One of them is the Welford algorithm [62] which has been thoroughly studied and tested.

Taken from [61]

```
function OnlineVariance(data)                ▷ Online variance using Welford algorithm
    $Mean \leftarrow 0.0$
    $M2 \leftarrow 0.0$
    $n \leftarrow 0$
    for $x$ in $data$ do
        $n \leftarrow n + 1$
        $delta \leftarrow x - mean$
        $mean \leftarrow mean + delta/n$
        $delta2 \leftarrow x - mean$
        $M2 \leftarrow M2 + delta * delta2$
    end for
    return $M2/(n-1)$
end function
```

12.3 Combining online variance with window

The previous algorithm can be applied sample by sample storing only the mean, m2 and the number of samples. The number of samples should be stored in variable with a sufficient number of bits to prevent overflow since n is incremented at each new sample: this should not be the case for an RFID reader that gave samples relatively slow. If we consider a border case with 10 values per seconds and n as an unsigned integer of 32 bits the system goes on for more than 10 years before overflowing.

12.4 Results

For the experiments the second solution has been used. Rssi and phase variances have been update at each new sample using the Welford algorithm. The result are showed in Figure 12.1.

The RSSI is more noisy than the phase but the difference between noise and movement is evident. The variance is very near to 0 with a still tag and increases when a movement is detected.

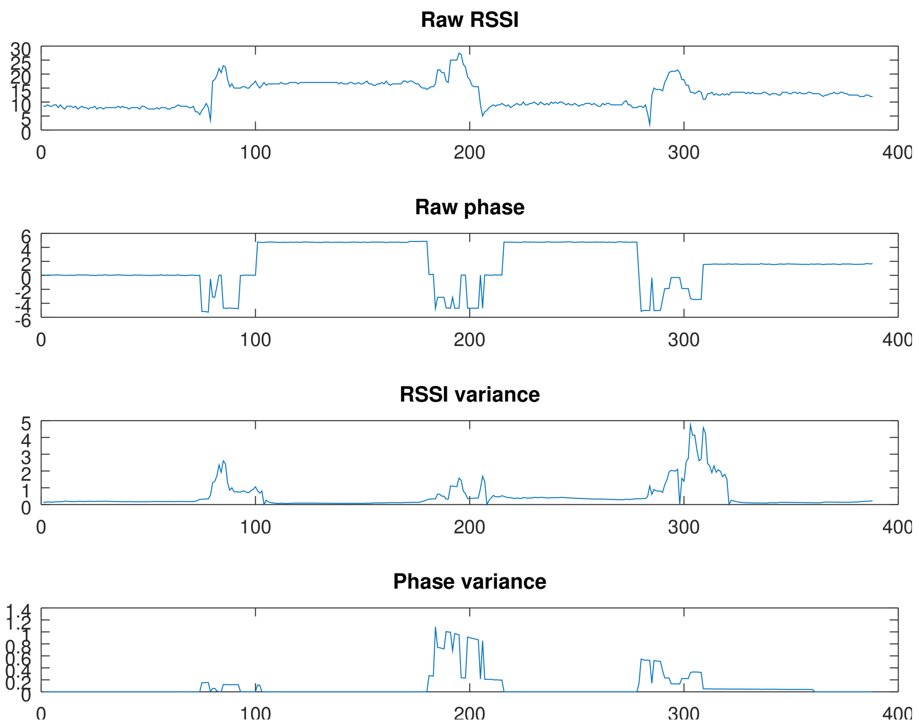


Figure 12.1: Sample number on x-axis. The movements are at around 100, 200 and 300

A debouncer is needed for both variances. For the tests a simple software debounce has been used based on window: after a sudden increase of variance do not consider any more movements for *s* samples. More refined algorithms can be used if necessary but they were beyond the scope.

The parameters that can be tuned are:

- The debouncer parameters
- The threshold of the variance above which a movement is detected

This short test shows some interesting results:

- A tag movement is clearly distinguishable from noise
- Touching or nearing a tag influences the system: some spikes are clearly noticeable during the beginning and the end of a movement while the tag is touched
- Rssi and phase are influenced in different ways during the interaction making them complementary: the first movement has been "noticed" better by the Rssi variance while the second by the phase

These results give another confirmation about the feasibility of the system that this thesis wishes to achieve.

Chapter 13

Artificial Intelligence

The aim of this chapter is a brief introduction on artificial intelligence. Two kinds of “machine learning” will be introduced here: the first one is a Support Vector Machine (SVM) and the second one is a neural network. The former is the method used for the current state of art [34] regarding the scenario of this thesis and the latter is the one used here with the attempt to outperform the first one. Some similar papers has worked on a similar scenario which resulted in a different approach, like [46], but all of them have a single common problem: how to classify the interaction with a tagged object starting from the raw data given by the reader.

Using a classical “algorithmic approach” is quite problematic since

- There is not a clear connection between inputs(reader data) and outputs(interaction)
- Interactions of the same type can give very different inputs

An artificial intelligence can be trained to behave as requested if some conditions are met:

- A relation between inputs and outputs must exists
- A sufficient number of data (inputs and correct outputs) for training is available
- The hyper-parameters of the AI are chosen carefully

13.1 Glossary

- **Training tuple**: an ordered list of inputs and their corresponding ordered list of correct outputs
- **Training set**: a list of **training tuples** used, one by one, to train the network
- **Validation set**: a list of *tuples* used to check if the network responds as expected. Usually a different set than the **training set**
- **Mini-batch**: a subset of *tuples* usually randomly chosen from the **training set**

13.2 SVM

An SVM is used for both classification and regression analysis. The original SVM is a linear classifier. A linear classifier can be explained using a graphical representation. To simplify only two classes of input data exist. If the data to be classified is represented by points of different colors in a plane, the SVM ideally gives a straight line that divides the plane into two parts each one containing only the points of the same color. Each point is an input to the SVM, the color is the class of the point and the “line” given by the SVM identify all the points in that class. The SVM should be able to classify points not present in the original set using their position with respect to the line.

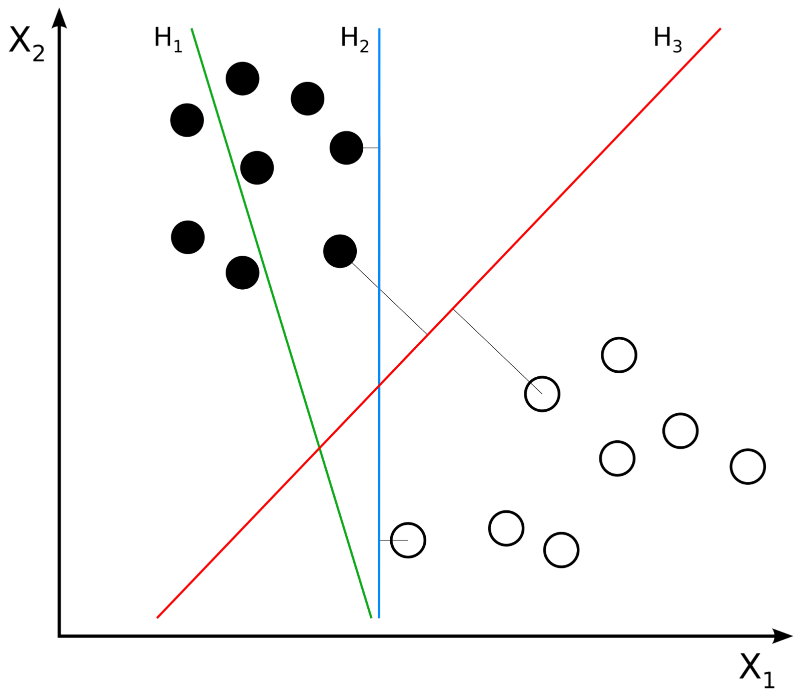


Figure 13.1: Three possible separations (H1, H2, H3) representing three SVGs. Source Wikipedia

The graphical interpretation can be extended to three dimensions and, in that case, instead of lines we have planes. As generalization it is possible to extend to even more dimensions and consider more planes if more than two classes are present.

The SVM has been extended over the years to behave as a non-linear classifier and one of the methodologies used is combining SVM with RBF kernels [50]. This is the solution chosen in [34]. One important problem of SVM with RBF kernel is that they do not scale well with an high number of samples and fails to generalize the data. SVMs are one of the first machine learning approaches which are still used today both in their original formulation and their evolutions.

13.3 Feed-forward neural network

Feed-forward neural networks are a class of neural networks made of nodes or neurons and connections or synapses. Their main characteristic is that each neuron forwards its value to the following neurons from the inputs to the outputs. In this kinds of networks there are no cycles so they have no memory of the previous status but the outputs are directly function of the inputs. Some network topologies have cycles in it and they are part of the Recurrent Neural Networks (RNN) which is a class on its own.

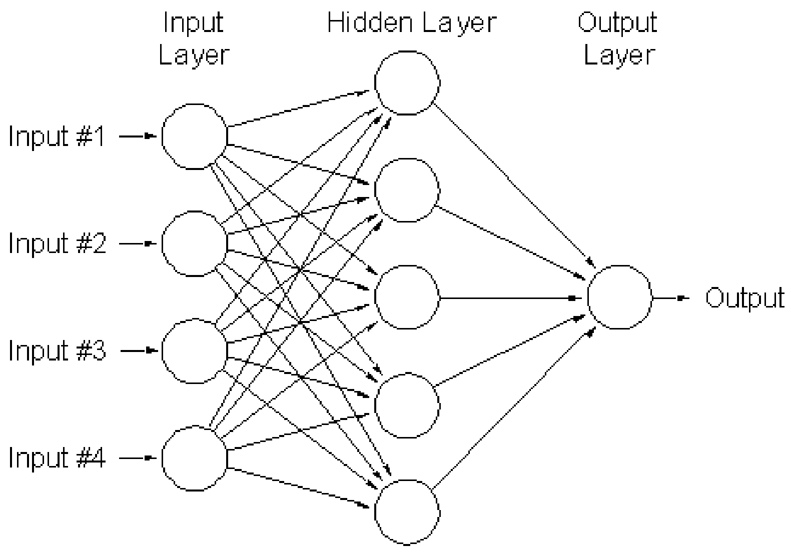


Figure 13.2: Typical feed-forward network [18]

13.3.1 Neuron internal structure

An artificial neuron, known also as perceptron, can be considered as a simplified mathematical model of a biological neuron.

An artificial neuron computes its value doing a weighted sum of its inputs: each one of its input connections has a weight w_i specific to the connection and a value x_i obtained from the input of the connection.

In some cases one or more bias neurons are inserted: a bias neuron has a fixed value and allows the network to output non zero values even when all the inputs are zero. Their role is to add a constant to the weighted sum (b in Figure 13.3). Since a bias neuron is usually considered as a normal neuron, it is connected to other neurons through a synapse with a trainable weight: if a bias is not really needed, is eventually disabled by giving a weight near zero at its output synapses (equivalent to $b \approx 0$).

Activation functions

Since all the mathematical operations involved are linear, their combination is linear too. So a network made of this neurons is only able to approximate linear functions. To extend

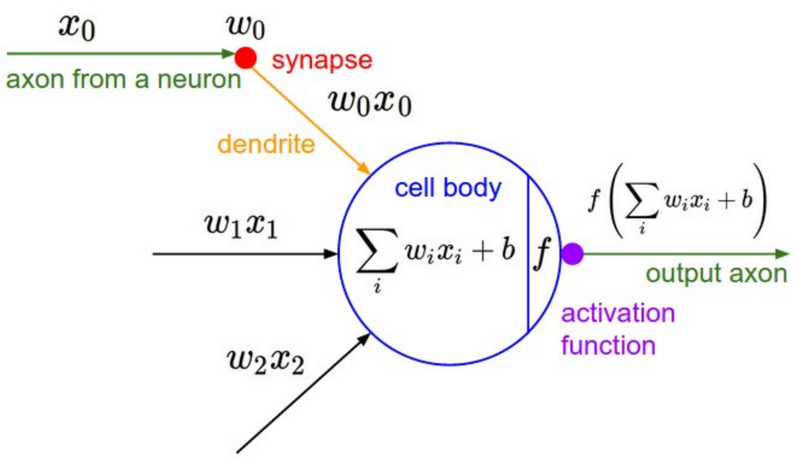


Figure 13.3: Neuron model [13]

the network to be non linear (if necessary) a non linearity must be inserted: the weighted sum of inputs became the parameter of a non linear function f and the result is the final output of the neuron. The function f is called activation function and historically has always been a sigmoid function. The functions which belong to this class are **S** shaped, tends to saturate, are continuous and are usually derivable. In recent years some more functions have been used like the Relu [41] which gives much higher performances (faster learning, less computing cost) but need some additional care: they usually do not saturate so during learning the values can grow until overflow. Some techniques have been studied to avoid this fact like batch-normalization [24].

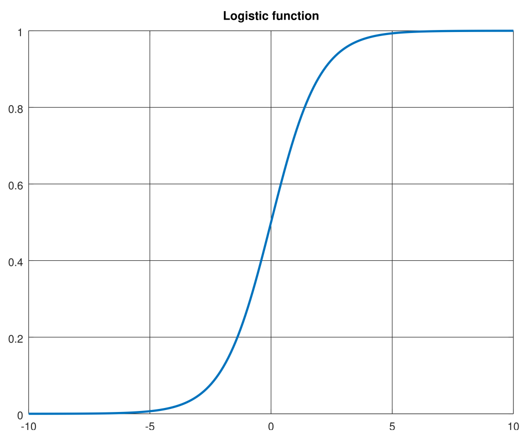
Batch normalization is quite computing expensive so some effort has been spent to create “self normalizing” functions which are able to give values, iteration after iteration, with some specific variance and mean. This studies resulted in the SELU (Scaled Exponential Linear Units) function which, with the correct coefficients (λ and α), has self normalization properties [29].

$$selu(x) = \lambda \cdot \begin{cases} x & x > 0 \\ \alpha \cdot e^x - \alpha & x \leq 0 \end{cases}$$

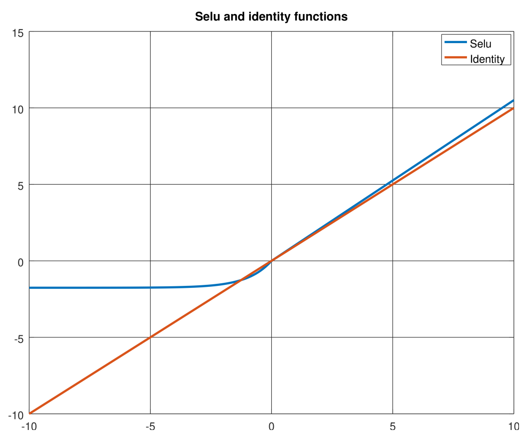
For a feed-forward network the ideal situation is having a variance of 1 and a mean of 0 for the inputs of each layer (Figure 13.4b).

13.3.2 Layered approach

In a feed-forward network is common to “layer” the neurons. The first layer is the input layer: every neuron in this layer has a value given by outside the network. All the neurons that have as inputs only synapses starting from the input layer make the first hidden layer and so on until the output layer. The neurons in the output layer does not have output synapses.



(a) A logistic function as an example of sigmoid curve



(b) Selu function which normalize to mean 0 and variance 1.
An identity function for comparison

Figure 13.4: A classical S-shaped function and a Selu function

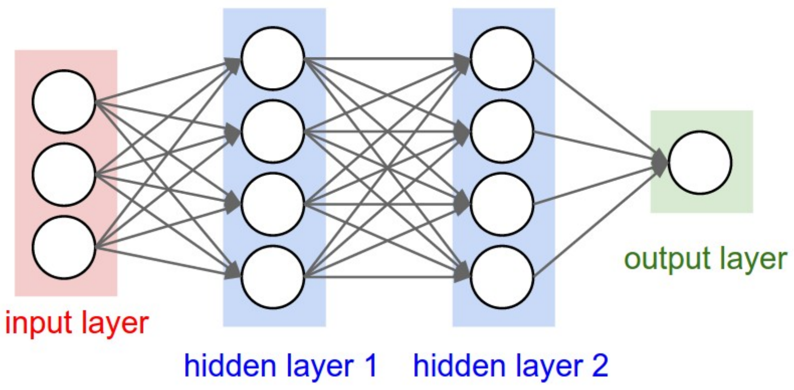


Figure 13.5: Layer structure [13]

The layered approach is particularly interesting because it allows to exploit vectorial instructions to compute the values for all the neurons in the next layer directly.

Due to the fact that using the vectorial instructions is more abstract than computing the next value one neuron at a time, in my implementation I have decided to start from a classical implementation (using loops to sum up all the inputs) and then in a second moment implement the operations using matrices and vectorializable loops.

The “matrix” implementation consists in

- Organize each layer of neurons in a vector
- Organize the synapses connecting two successive layers in a matrix

If we consider two successive layer x_i x_{i+1} and W^T the matrix of the synapses where i is the position of the layer starting from 0 (the position of the input layer) to n (the position of the output layer) we have

$$x_{i+1} = W^T \cdot x_i$$

Repeating the operation $n - 1$ times we get the outputs. The matrix of the weights W needs to be transposed W^T to match the sizes and get a vector of the correct size as result.

The performance gaining of this approach is huge due to the fact that vectorial instructions use an hardware module on the CPU to compute all the output values at once instead of one at a time. This particular architecture is also know as Single Instruction Multiple Data (SIMD). As a reference, on an mobile low power Intel Core processor, doing 250000 training iterations (each of which requires two feedforwards) takes around 50~60 seconds to finish while using the optimized matrix version only takes a little more than 4 seconds.

13.4 Learning

In the previous section I have often spoken about training or learning. An artificial intelligence is a mathematical model and "learning" is equivalent to adjust some of its numerical parameters.

In a feed-forward network this parameters are the weights of the synapses which manipulate the final output of the network. A whole literature exists about training so this section is a simple attempt to give a brief introduction to supervised learning.

The training step can be considered as the opposite of the feed-forward: going from the required outputs back to the inputs with the attempt to fit the two parts.

Each training step consists of:

1. Feed-forward the inputs so to update the values of the neurons
2. Compute the error at the outputs
3. Back propagate the errors to the inputs
4. Update the weights using the back propagated error

The key word here is back propagation: using a gradient descend method the error at the outputs is propagated back to the synapses deciding how to distribute it across them.

Lots of parameters are involved:

- The error function at the outputs should be chosen depending on how the results are interpreted. The standard choice here is the square of the Euclidean distance between each output and its expected output
- The so called hyper-parameters which influence the computing of the correction. Some examples here are the learning factor (or scaling factor), the momentum, the number of iterations, whether or not apply regularization and its own parameters

Back propagation assumes that the error is a function with a minimum and applies gradient descend to attempt to find that global minimum. In real case scenario, with lots of neurons, the error function has some local minimum at which the backprop can be stuck: this is one of the biggest issues of this method and various extensions, like momentum, have been created in order to avoid or mitigate this unwanted behavior.

Another import issue is over-fitting: a network is trained to approximate so well the training set that it is unable to react to unknown inputs. Another way of considering over-fitting is: if the learning rate is too high the network is trained to fit the last inputs/required outputs tuple and “forgets” the previous training steps.

The original back propagation has been improved using “adaptive learning rates” that are dynamically modified during training. Some of this methods are:

- ADAGRAD [15]
- RMSPROP [56]
- ADAM [28]

Each of this methods are able to shape the learning rate during training. They over-perform the vanilla back propagation algorithm but introduce more hyper-parameters that have to be set manually. Luckily some of this parameters have some recommended values that work in most of the cases and can be further optimized if necessary after the initial training.

13.4.1 The particular case of this thesis

For the purpose of this thesis, the standard hyper-parameters for ADAM have given the best results and the only “problematic value” has been the initial learning rate.

The initialization of the network is non deterministic even when fixing all the parameters involved like the number and type of neurons and how they are connected. The synapses’ weights are initialized using random values and this leads to different error functions and starting points on them. In order to test a new hyper-parameter values the whole network needs to be reset and retrained at least 4-5 times: using the same validation set of samples the network has an error rate that may vary of some percent points (up to 10% difference has been observed).

13.5 Organize the data

To improve the accuracy and/or the training time, the inputs must be manipulated and the outputs must be coded in a reasonable way. What has been said in 13.3.1 about the hidden layers ideal inputs is valid for the input layer too. When dealing with activation functions which do not saturate (Relu, Selu) it is important to keep the internal values as little as possible while having the ideal conditions. So giving small inputs to the network helps to satisfy the previous statement. The same discussion can be done on the outputs: they output range is limited and it usually better to keep it constrained in a specific range like [0,1] or [−1,1] even if the network can go slightly outside them.

In some cases modifying the activation functions of the last layer can gives some improvements: some good choices are the sigmoid functions like the logistic function or some more complex like a Softmax layer. In the former case we can get some values constrained between 0 and 1 which can be interpreted as independent probabilities or in the latter case an Softmax layer helps to get as outputs a list of related probabilities which sums up to around one.

In this category we can insert the organization of the training data. The training set is usually a big list of samples without any particular order. Due to its size, training with a whole set at once can be problematic. The strategy in this case is to divide the set in mini sets, also called batches, and to train on each set individually. Usually using online weights update (update the weights after each individual sample) gives better results but it is a sequential process. If a lot of hardware resources are available, it is possible to train each batch separately and in parallel, then average or sum the corrections before updating the weights.

When dealing to batches, it is fundamental to stop updating the weights if the minimum acceptable error has been reached for the current batch to prevent over-fitting. I have seen slightly improvements when randomly dividing the training set in small batches, training on each one of them and then re-dividing the whole set in different batches than before and so on until reaching a good precision.

Chapter 14

Current state of art

The current state of art has been achieved in 2015 with «IDSense: A human object interaction detection system based on passive UHF RFID» [34]. In the paper they have analyzed three different scenarios and only the last one is related to this thesis scenario, but since both their approach and mine are not dependent on the specific application, the system can be used to recognize the different movements involved in different scenarios as long as the technology can give enough data to distinguish between them.

14.1 Raw data pre-processing

Before inputting the data into an artificial intelligence it is required some conditioning. In [34] they have proposed some values:

- The standard deviation of the RSSI
- The mean of the RSSI standard deviation withing each frequency
- The mean of difference between neighboring RSSI
- The median of the CFPR
- The sum of the absolute values of the CFPR
- The standard deviation of the CFPR
- The standard deviation of the VFPR
- Read rate

The CFPR (Constant Frequency Phase Rate) is defined as

$$CFPR = Phase[i+1] - Phase[i]$$

where i is the current sample and $i+1$ the following (at the same frequency) and the VFPR (Variable Frequency Phase Rate) as

$$\frac{Phase[i+1]-Phase[i]}{Frequency[i+1]-Frequency[i]}$$

where i is the phase at a frequency and $i+1$ is the phase at the following frequency. Since the experiment has been conducted in Asia, the number of total channels/frequencies is 50 (in Europe only 4 are allowed).

Notice as all the values tries to identify difference of status for the same tag in different moments:

- The RSSI is the strength of the backscatter signal and depends not only on the distance tag-reader but also on the multipaths.
- The CFPR is the difference between to neighbor frequencies and due to the frequency hoping time required by the reader can identify changes on the tag-reader distance
- The VFPR is proportional to the distance tag-reader [43]
- The reading rate is directly related to the RSSI and decreases as the RSSI decreases

The window size is two seconds, considered sufficient for a normal human-object interaction.

14.2 Results

Some interactions have been proven to not be distinguishable like rotation and movement so they have been grouped. The total number of classes is four and they are still, motion, swipe and cover with an accuracy of around 95 % in the best case with only one tag. This precision has been achieved using an SVM classifier with a validation set of 1600 entries.

In a real retail environment (clothes shop), the precision falls to 92.5 % with 50 tags and ten moving people.

Different environments lead to different performances and different people can interact in different ways with the objects. In the paper they assume a person that does the different interaction for 10 minutes so to obtain enough training data for basic learning. To further refine the classifier, lots of data is needed taken in various environments, different situations with different people.

They conclude saying that the method is applicable to various scenarios with various aims. This technology is relevant since it is able to make sensing human interaction "easy and unobtrusive, by minimally augmenting objects with low-cost and long-lived RFID tags".

Chapter 15

Improving the state of art

The proposed system in [34] can be further improved to scale better for higher tags number. I have identified some “choices” that can improve the feasibility of the system even if reducing the precision:

- Try to reduce the number of the operations required for the AI inputs
- Change the AI to a feed-forward type
- Do not involve timings (2 seconds window) during data accumulation but use inventory iteration as discrete time

In the next section various attempts to improve the precision of the solution using the European frequencies are done. The system “as is” is not usable in Europe and some changes are required in order to works with the different regulations.

The samples used for training and validation are taken in not ideal conditions including:

- Antenna positioning
- Lots of humans/objects in range even between the antenna and the tags
- Tags near the edge of the range
- Around 20/30 tags for different vendors, sizes, type (metal and non metal mountable) always in range

The aim of this part is to increase the precision of the system in an environment worse than the one in which the system will be used, trying to amplify the interference of a real scenario.

For these tests all the tags in range are read but only one is processed and used to obtain the following results.

15.1 Pre-processing the AI inputs

The experiment in [34] has been conducted in Asia which allows up to 50 different frequencies for UHF RFID. Since in Europe only 4 channels are allowed, the steps required to compute the values are a lot less then in the paper case. The frequencies in Europe are nearer so:

- It has been observed that the RSSI changes between the European channels are very few if none
- The phase is much more affected by the little changes in frequency but much less than the paper case

This lead to two forced choices:

- the only value, based on RSSI, that will be passed is the standard deviation of it (across all 4 frequencies).
- The median of the CFPR is often zero due to the CFPR being zero very often making this values useless

The reduced number of channels has decreased the precision (60~70 %) of the system but has radically reduced the data pre-processing required resources. Nothing can be done on the number of channels, so it is needed to process the data in other ways in order to obtain some meaningful values.

The sliding window size for the next experiments is 10 samples. Each time a new sample is available, the window is updated and its median or average taken and stored separately (Figure 15.1). Ten “medians” or ten “averages” are then used in the following proposals: each value can be taken at a specific time interval to slow down the system so to obtain an acceptable trade off between reactivity (time between the interaction and the response) and instability (response oscillation between an interaction and uncertain).

This procedure is independently applied to each value, the RSSI and the four phases.

15.1.1 First proposal

The first test have been done with the following changed inputs:

- The median of the CFPR has been substituted with the average of the CFPR
- The standard deviation of RSSI has been divided by 10 so to not become greater than 1
- The sum of CFPR has been divided by 10
- The standard deviation of VFPR has been divided by 10
- The CFPR has not been calculated using successive samples but doing a jump of 2

Using a training set of 462 samples and a threshold of 5% (an output below 5% from the network is considered as “uncertain response”) results in an average error between 22~26 %. That percentage groups both errors and uncertainties as errors. The uncertain percentage is usually between 4% and 18% depending on the network training parameters and the particular training session.

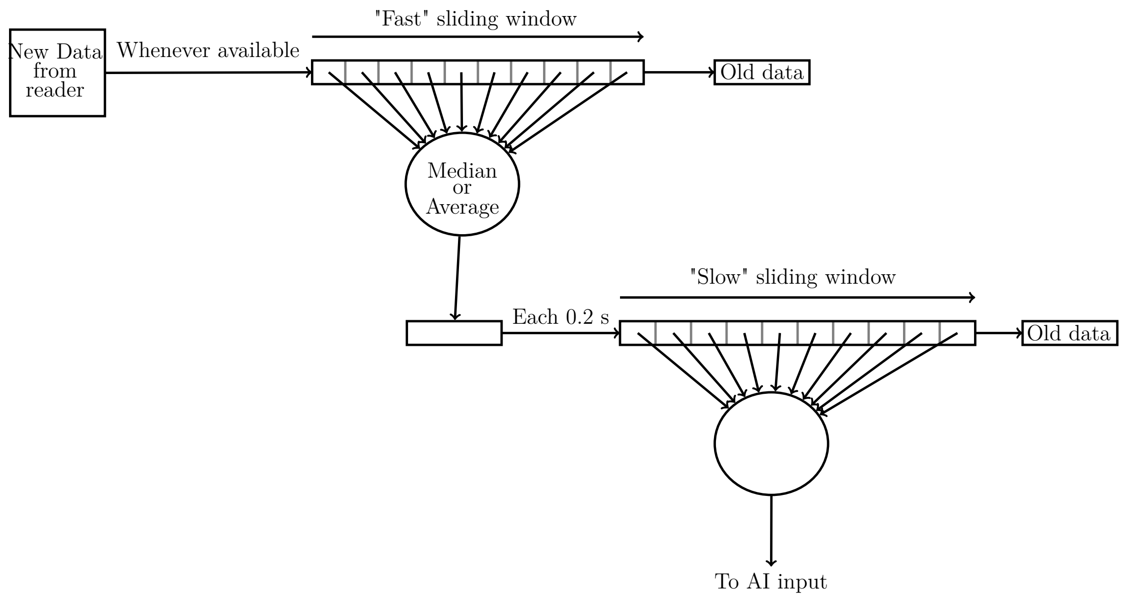


Figure 15.1: Sliding windows used to compute the inputs.
5 "fast" sliding windows (RSSI and the 4 phases at each frequency) and one "slow" sliding window for each AI inputs are kept

15.1.2 Second proposal

The second test has been done settings as inputs the whole window values so the values given to the network are:

- 10 RSSI values
- 10 phase values for each frequency

totaling to 50 inputs. The values needs normalization (a simple division is enough) to avoid too big values. The precision is a little worse than in the first proposal with an error rate between 30~40 % and the size of the network increases resulting in higher training and response time. This proposal will not be used anymore and the first proposal will be kept for further experiments.

15.2 Feed-forward network settings

A feed-forward network is a very flexible tool but its own flexibility results in an higher complexity in choosing the right hyper-parameters. The procedure I have applied here is:

- Start from some reasonable parameters
- Fix all parameters but one to be tuned
- Repeat the previous step for all the parameters

- Try to change two or three parameters at once trying to improve the precision a bit more

15.2.1 Network size

The size of the whole network can be modified pretty easily. The only fixed parts are the number of inputs and the number of outputs. I have chosen to add one additional input, a bias neuron, with a constant value of 1.

The aim is to reach the minimum network size while keeping a mostly constant precision. The test for each network needs to be performed different times due to the unpredictability of the starting weights.

The main program has been modified to test all the possible network topologies starting from no hidden layer up to 13 hidden layers each of 1 to 13 neurons. Each topology has been tested 5 times starting from an untrained network.

All the others parameters are fixed for all the iterations.

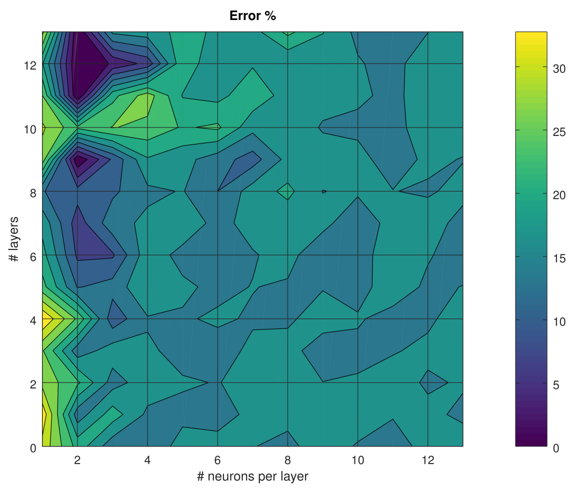
The training has been done randomly dividing the 462 samples in 10 mini-sets for 10 times. The network trains on each mini-set for at most 500 times or until it reaches a mean squared error of 0.005 on its outputs before switching to next batch.

This training is not sufficient to obtain the best error rate, which stays larger than the one mentioned in the previous section, but can give an idea of how the network size and topology influences the performances.

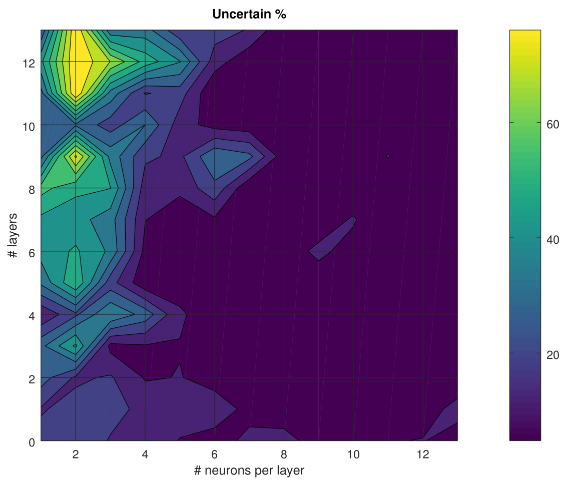
In some cases the error falls below some points % but the uncertainty rate of more than 90% makes the network unable to recognize the interactions. To get more relevant statistical data the training is repeated multiple times (5 as mentioned before): for each topology the average errors and uncertainties as well as their standard deviation is taken into account.

The obtained results are in Figure 15.2.

The zones where the error is 0% are also the zones with the greater uncertainty. Smaller networks (on the left of the two plots) are more prone to low confidence: they are not complex enough to distinguish between the interactions. Merging the data that has generated the two plots, which equals to considering the uncertain responses as errors gives a better idea of the performances Figure 15.3.



(a) Contour plot of the error in %. Range is 0(purple) to 36(yellow) %



(b) Contour plot of the uncertain samples in %. Range is 5(purple) to 83 %

Figure 15.2: Comparison of precision of the network with different sizes

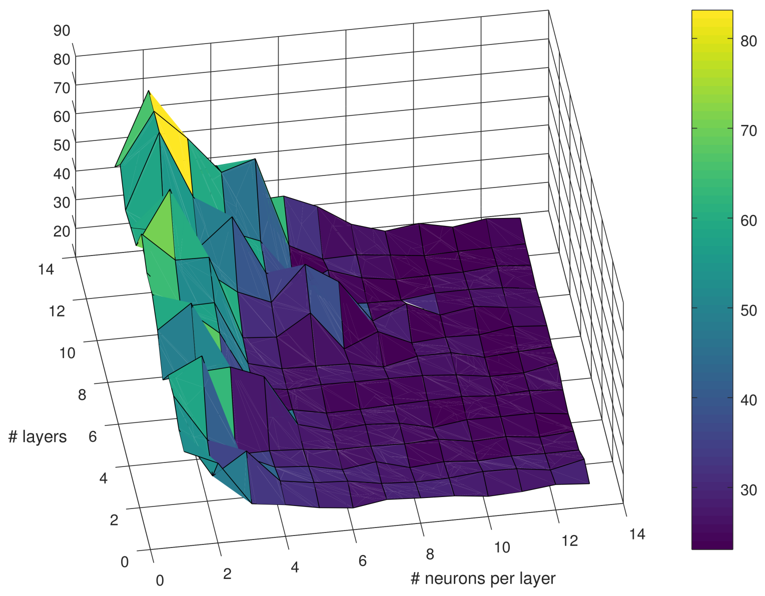
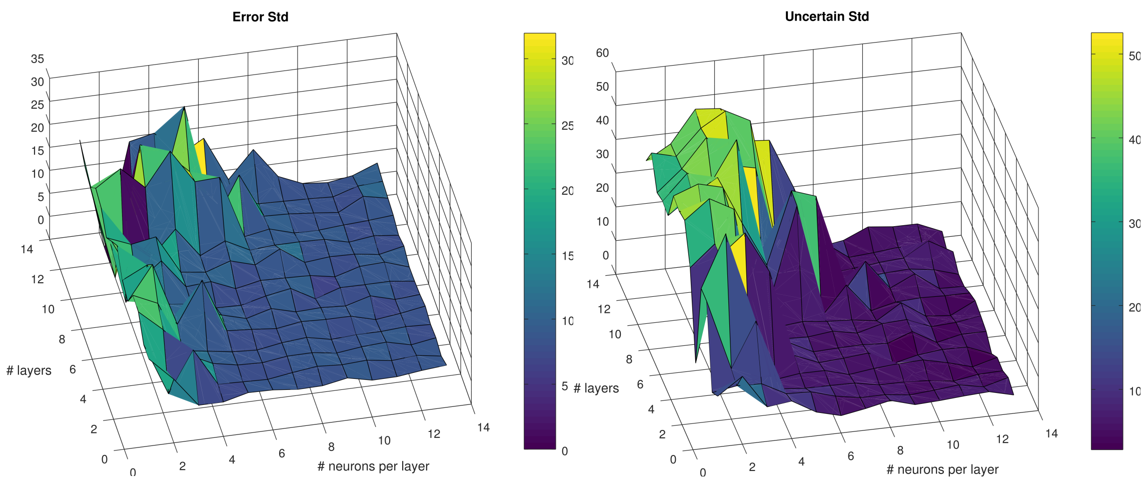


Figure 15.3: Errors and uncertainties considered as errors.
Range is 23(purple) to 83(yellow) %

Another tested parameter is the reproducibility of the precision of a particular topology. Each network start with random weights which than needs to be adjusted to fit the aim of the network, so for reproducibility here is intended the capacity of the network to reach the expected precision after a specific number of learning iterations starting from a random initial situation.



(a) Error (b) Uncertain

Figure 15.4: Comparison of the standard deviation of the parameters which define the performances of a network topology

The results are similar: the decisiveness of a bigger network is reflected in its performances reproducibility.

Another interesting characteristic displayed is that it is better to keep the number of layers as small as possible and instead increase the number of neurons per layer. This is known as the “Deep learning problem”: too many layers cause the loosing of the inputs’ features and the precision of the network reaches a plateau or gets lower. Switching to another structure like Residual Networks [21] solves this problem but increases drastically the size and the complexity of the system.

15.2.2 Neurons activation function

The chosen activation function is the Selu function for all the inputs and hidden layers. Various other choices are possible but Selu gives the best precision/speed trade-off.

Other possibilities are Relu plus Batch normalization or one of the classical Sigmoid functions.

For the output layer the choice is more complex:

- Use Selu and train the network to never give outputs outside [0,1]
- Use a logistic function which can give only outputs between [0,1]
- Use a Softmax layer which gives the outputs so that their sum is near 1

The outputs should be interpreted as a probability of the inputs belonging to a specific category.

Using a logistic function we have an independent probability of the inputs belonging to each of the chosen classes. As an example if the outputs are:

1. 0.65
2. 0.9
3. 0.01

the first one (0.65) is the probability that the inputs are of the first class, 0.9 of the second class and 0.01 of the third class. The values are clearly independent since the does not sum up to 1.

The Softmax layer computes the input values for the last layer making a sort of weighted average of all the inputs.

The Softmax function is

$$f_i(x) = \frac{e^{x_i}}{\sum_j^J e^{x_j}}$$

where i is the position of the output neuron and j is the position of the previous layer neuron and J is the total number of neurons in the previous layer. The function needs to be applied to each output neuron.

After trying the various possibilities, the final choice is:

- Use Selu for all layers but the outputs
- Use the logistic function for the outputs

Using Selu for the outputs usually give values grater than 1 with the need of saturation and the Softmax layer, while giving more convenient outputs since each input should belong to only one class, reduces the precision of up to 10 %.

15.2.3 Choice of outputs

The network needs to be trained go give the requested outputs. The outputs need to be chosen before starting the training and are not easily changeable without re-training.

If using the network to approximate a function (regression) the outputs should be the required one or a scaled/normalized version of the required output. Using the AI as a classifier, there are more choices. As mentioned in the previous section I have chosen to get outputs between 0 and 1, easily interpretable as probabilities. There is an output for each class which gives the probability of the inputs belonging to that specific class. So, at the end, the network compute n probabilities for n classes and the response class is the one with the highest probability.

If the maximum probability is below a certain threshold, the response can be considered as “uncertain” since the network does not have enough confidence.

Other possibilities here are:

- A different output range instead of [0,1]. A possibility here is [−1,1] which allows to use different error functions at the outputs like **Cross-Entropy** functions (Kullback-Leibler divergence [32] and its derivatives, Bernoulli log-likelihood)

- Using regression and assigning a specific value called *score* to each class. Each class should be "far enough" (delta) from the others but at the same time the output range should not be too large. In this case the network has only one output. This is the approach used by multiclass SVMs. Figure 15.5

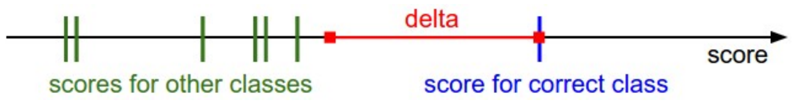


Figure 15.5: Example of division into classes using scores.
delta is the margin that separates one class from another. [13]

15.3 Training settings

The training phase has some hyper-parameters that influence the final performances of the trained network:

- The learning factor is the step size and it is used to scale the corrections to avoid too big adjustments
- The number of training iterations per mini-set
- The maximum allowed error per mini-set
- The normalization type and its values (strength) used to avoid over-fitting
- The number of mini-sets
- The iterations over the whole set

Some of those values can be identified doing some empirical tests and can be fixed while doing a more fine grained research on the others. Some general advice is present in the literature for some parameters:

- The learning factor should be searched in a logarithmic space (0.1, 0.01 ...) and if possible using a random layout instead of a grid layout
- The normalization factors should be very small values like $10^{-6} \sim 10^{-8}$
- The maximum allowed error per mini-set should not be too low to avoid over-fitting

The other parameters are usually application-dependent and need some tries to be identified.

For the following tests the network characteristics have been fixed using the data obtained in the previous section and the following settings:

- The training set of 462 elements has been randomly divided in 10 mini-set

- 5 iterations over the whole training set
- Maximum allowed MSE (Mean Squared Error) of 0.005 per mini-set
- The normalization is fixed to 10^{-8} for both L1 and L2

Each training settings have been tested 5 times in order to analyze the reproducibility of the results as in the previous section.

The results showed that the best error rate is achieved with 300 iterations over each mini-set with a learning rate of 10^{-3} Figure 15.6

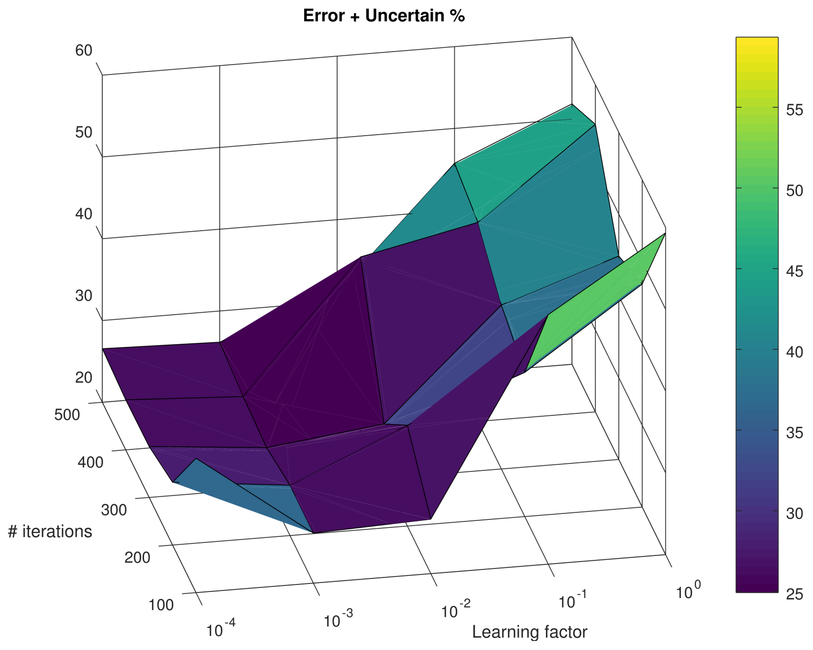


Figure 15.6: Errors and uncertainties considered as errors.
Learning factor (log scale) and # of iterations (linear scale) as parameters

A similar result is obtained when considering the standard deviation (which is the square root of the sum of the two variances) of the sum of errors and uncertainties for each training setting Figure 15.7: settings that gives the better precision allows better predictability of the results which usually results in smaller training times.

Greater learning rates perform worse than the smaller ones due to the greater weights adjustments that make the network “jump” around the minimum of its error function. This oscillation avoids further learning and the final result depends more on the initial random weights than when doing smaller steps toward the minimum of the error function.

Further refinement in the choice of the learning factor shows no particular improvement in accuracy and the variations are inside the standard deviation margin Figure 15.8.

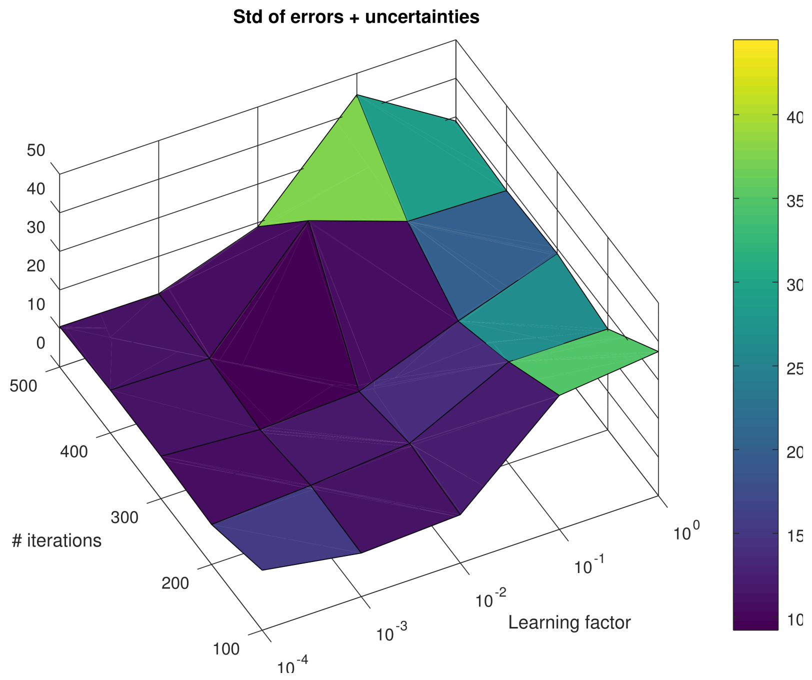


Figure 15.7: Standard deviation of errors+uncertainties for different learning parameters Learning factor (log scale) and # of iterations (linear scale) as parameters

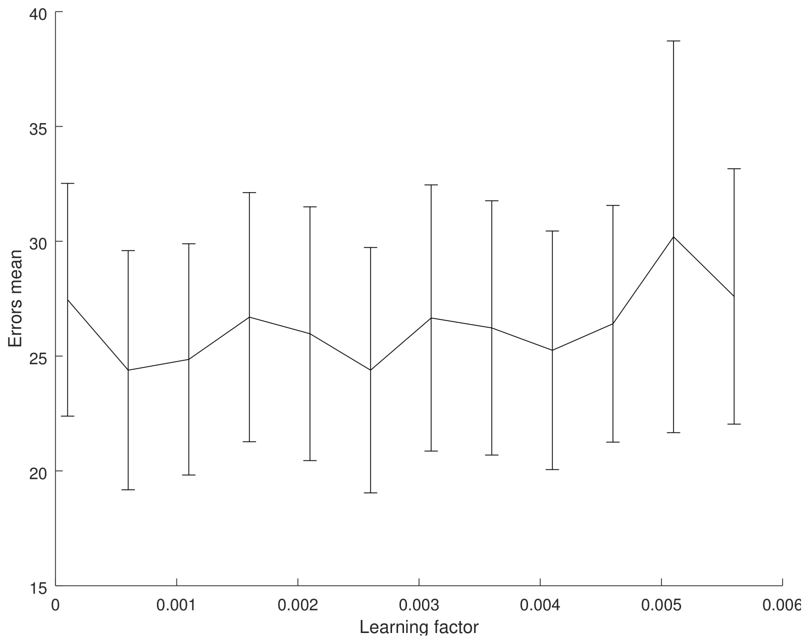


Figure 15.8: Error mean using various learning factors between 0.0001 and 0.0060

15.3.1 Training set

In order to train the network a lots of samples are needed. It is common belief that the higher number of samples the better the precision of the final network. The total number of available samples is 462. For the following tests the whole set has been divided in 66 mini-set of 7 values each. The network has been trained 5 times with an increasing number of mini-sets starting from 7 values used for training up to the whole 462. The network has then been validated using the whole training set of 462 values Figure 15.9.

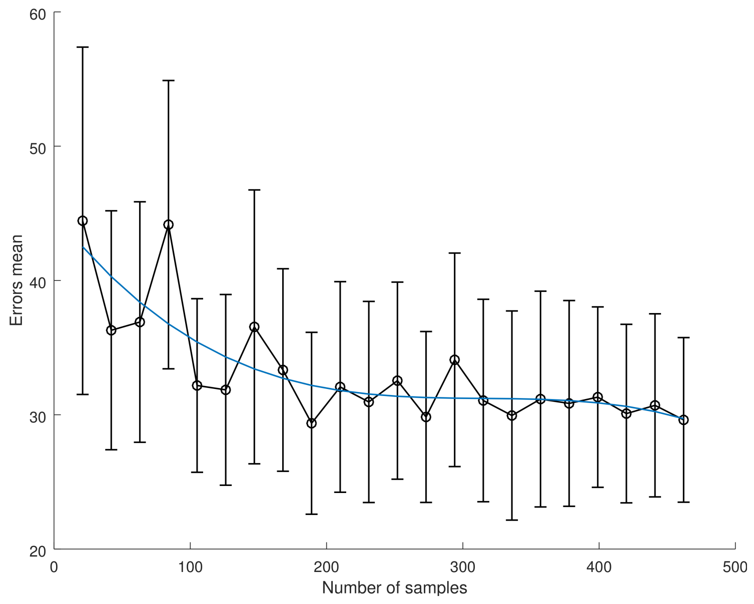


Figure 15.9: Error mean using a different number of samples.
The whole training set is used as validation set

The same experiment has been done using a different validation set while keeping the same training set: the network precision is tested using samples that the network has never seen during training (Figure 15.10). The result is quite similar to the previous one.

Increasing the number of samples has less impact as the total samples increases or can be said that the higher the number of training samples the higher the number required to further refine the network (Figure 15.11).

The gain goes negative between around 280 and 340 samples meaning a loss of precision of 0.04% in that zone. Further increasing the samples shows an increase of precision of 1.05% when going from 340 to 462 samples.

A feed-forward network is considered error-resilient even in the presence of wrong values in the training set: the loss of precision in the previous mentioned range probably is due to some wrong samples combined with a little over-fitting. Adding some more hopefully correct samples helps the network to fix its response.

To conclude: adding more samples is generally a good idea because the networks can refine better its weights and helps in discarding the wrongly taken tuples which presence is highly probable in a real scenario environment.

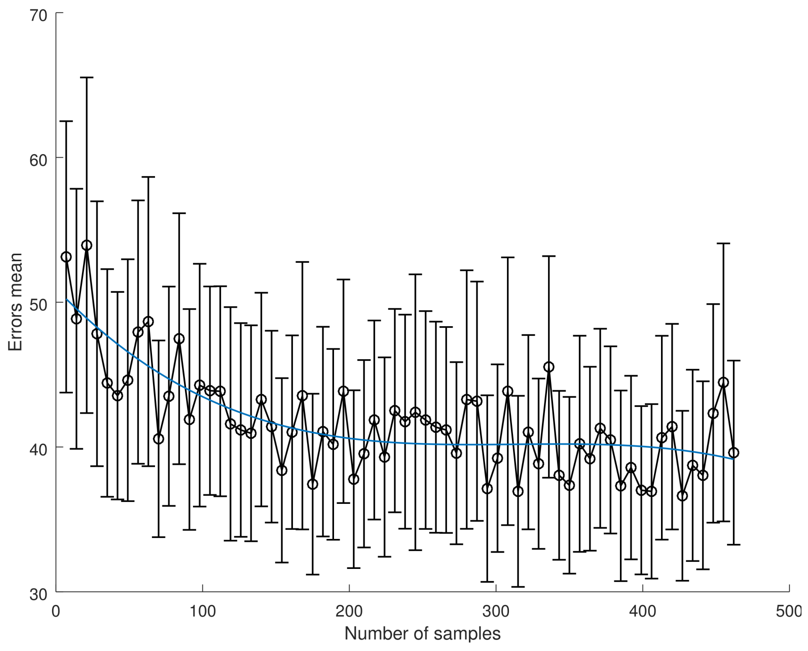


Figure 15.10: Error mean using a different number of samples.
A different set of 238 samples is used as validation set

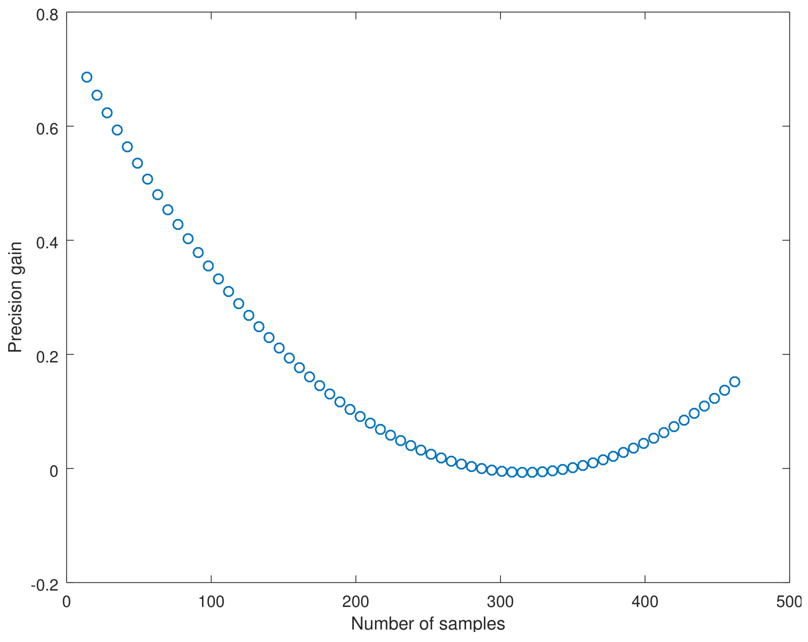


Figure 15.11: Error decrease in % when adding more samples to the training set.
A different set of 238 samples is used as validation set

15.4 Response time of the system

The bottleneck that slow down the response of the system is the time required to accumulate the samples from the RFID reader. As reported in Chapter 5 the reader requires more time to read all the tags when their number is high. The window should vary its size depending on the read rate; keeping a fixed size window can lead to no data available in some particular moments.

My proposal is to use "sliding windows" of samples which are updated as soon as a new sample is read. A list of the last n samples, where n is the size of the window, is kept for each tag (EPC). The proposed structure can be seen in Figure 15.1. A possible implementation can be:

```
 1: function TagRead(TagInfo)                                    ▷ Called when a tag is read
 2:                                            ▷ TagInfo contains all the data about the read Tag
 3:                                         ▷ IdentifiedTags contains the last n values (window)
 4:     if (not IdentifiedTags.contains(TagInfo.EPC)) then
 5:         IdentifiedTags.addTag(TagInfo.EPC)                  ▷ Tag read for the first time
 6:     end if
 7:     IdentifiedTags[TagInfo.EPC].addValues(TagInfo)
 8:                                        ▷ Compute the inputs for the network using the window
 9:     IdentifiedTags[TagInfo.EPC].computeInputs()
10:                                               ▷ Run the network with the computed values
11:     RunFeedForward(inputs = IdentifiedTags[TagInfo.EPC].getInputs())
12:                                                          ▷ Read the response and process
13: end function
```

With this approach the values of each tag are updated and ready as soon as the tag is read. Since Octane SDK return a list of tags instead of a single tag, the process can be further optimized using multiple threads.

Computing the input values in this way it usually quite fast:

- Computing cost for RSSI standard deviation is $O(n)$
- Computing cost for CFPR related values is $O(n \cdot f/2)$
- Computing cost for VFPR related values is $O(n \cdot (f-1))$

where n is the window size and f is the number of frequencies (4 in Europe).

The total computing cost per tag is linear with respect to the window size and it is usually negligible.

The window size used for the experiments is 10 samples which gives a good compromise between the time required to take a window for training purposes and responsivity, time between the interaction and the response on screen.

The time required to the neural network to compute its response is negligible too: on a mid-range laptop with the samples pre-stored on the hard-disk, the time consumed to compute 462 different readings, which corresponds to 462 windows with size 10, is around 0.013 ~0.021 seconds.

This time has been obtained using the *time* utility on Linux and includes the time to load the network weights from the disk, initialize the network, parse the text file which contains the 462 samples, compute the error and uncertain percentages of the outputs, print the statistics on the terminal.

15.5 Antenna position

The previous tests, as already mentioned, has been conducted using non ideal antenna positions. In [34] they advice to put the antenna on the ceiling to enhance the precision. The major interference is the human presence: no human should be present in the path between the reader antenna and the tag.

While putting the antenna on the ceiling is doable for some shops without shelves, like a clothes shop, it is not ideal for other shops which have most of the products in metallic shelves. In these cases, the performances of the system is not highly impacted but the range is drastically reduced. Using multiple antennas around the shelves should assure the coverage. Depending on the shelf type different locations are possible:

- The top, if not occupied by products
- The bottom or on the floor below the shelf
- To the side of each shelf
- Below each shelf, if the space is sufficient

In this situation it is inevitable to have the same tag read by multiple antennas: the window in this case should be distinct not only by tag EPC but by antenna.

15.6 Dividing the system in multiple parts

The complete system is made of different software elements with specific functions:

P.1 SDK (interface with the reader)

P.2 Tag info (EPC, RSSI, phase) grouping

P.3 Window management

P.4 Numerical computations (RSSI, CFPR, VFPR related values)

P.5 Neural network implementation and management

P.6 Outputs interpretation

P.7 Response dispatch (to the user, to a DB)

Normal operation and training behave in different ways so the two phases has been kept separated. For the "proof of concept" the following choices have been done. During normal operation **P.1-P.4** are in the same Java program, while **P.5-P.6** are a C++ program and **P.7** is just a message on the terminal in which the program has been run.

The training phase has been divided in storing raw data and elaborating data.

The first step was done with the SDK that get the tags data and write it to a file in the **\tmp** folder **P.1**. A Python script is in charge of checking if the file in **\tmp** has changed, parses the required data, manages the window and writes the window to a separate file **P.2**. The script is in charge of asking the user the type of interaction done with the tag during the window.

The raw data stored can be later processed without the need of re-taking the sample if the data processing needs to be changed.

A second Python script is in charge of the processing: reads the windows stored in multiple files inside a folder, processes the data and computes the inputs for the network **P.7**.

A modified C++ program parses the generated file, instantiate a network, performs the training and stores the values required to reconstruct the network in two files containing the list of neurons and synapses, their characteristics and how they are linked **P.5-P.6**.

The two saved files can be used during normal operation to load the trained network and start the working phase of the system.

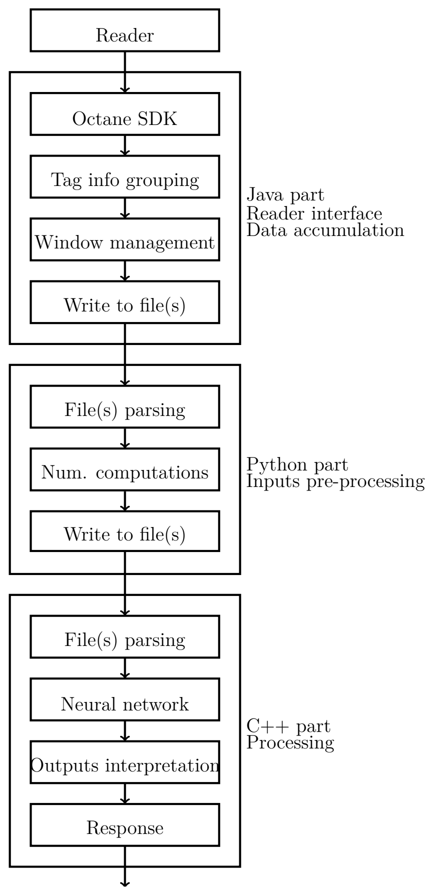
The training phase is only related to **P.5** so all the other parts, except **P.7** which is related to the interface with the user/other systems, have to be equivalent for the training to be effective with the final system.

The choice to divide the two phases has been done to have maximum flexibility when choosing the parameters of the system, as analyzed in the previous sections (processing raw data, network topology, ...). In the final system, where the parameters are fixed, the training and normal operations can be merged in a single system like the one used for the "normal operation": Java part **P.1-P.4** and C++ part **P.5-P.6** plus an additional interface like a database in a real scenario as in Figure 15.12b.

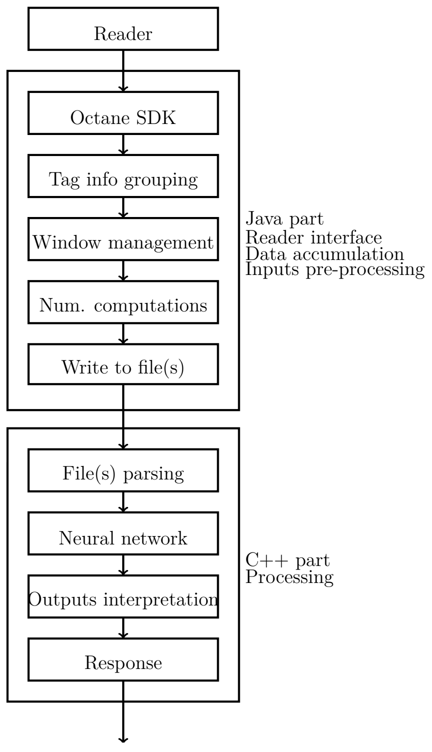
The data pre-processing has been assigned to the Java parts due to its limited computation cost; moving it to the C++ part implies passing all the tag raw values (EPC, RSSI, phase for each frequency, frequency), which are not usually available all together but at different times, after each tag read instead of only passing the already processed inputs for the network.

It is advised to update **P.2-P.3** as fast as possible, after each reading, to always have the latest data available and keep the window always updated. The computation **P.4** can be done only when the "user" requires it or at specific intervals. For the tests the interval has been set to 200 milliseconds but values up to one second are doable in order to reduce the system load and the output data.

Until now only a single tag has been processed: the reader is set to consider only a specific EPC. The approach in Figure 15.12 is scalable to multiple files: for each tag a file with the name corresponding to the EPC is created and inside it is written the data.



(a) Initial implementation



(b) Improved implementation

Figure 15.12: Comparison of implementations

15.7 Integrating the system, data visualization and final results

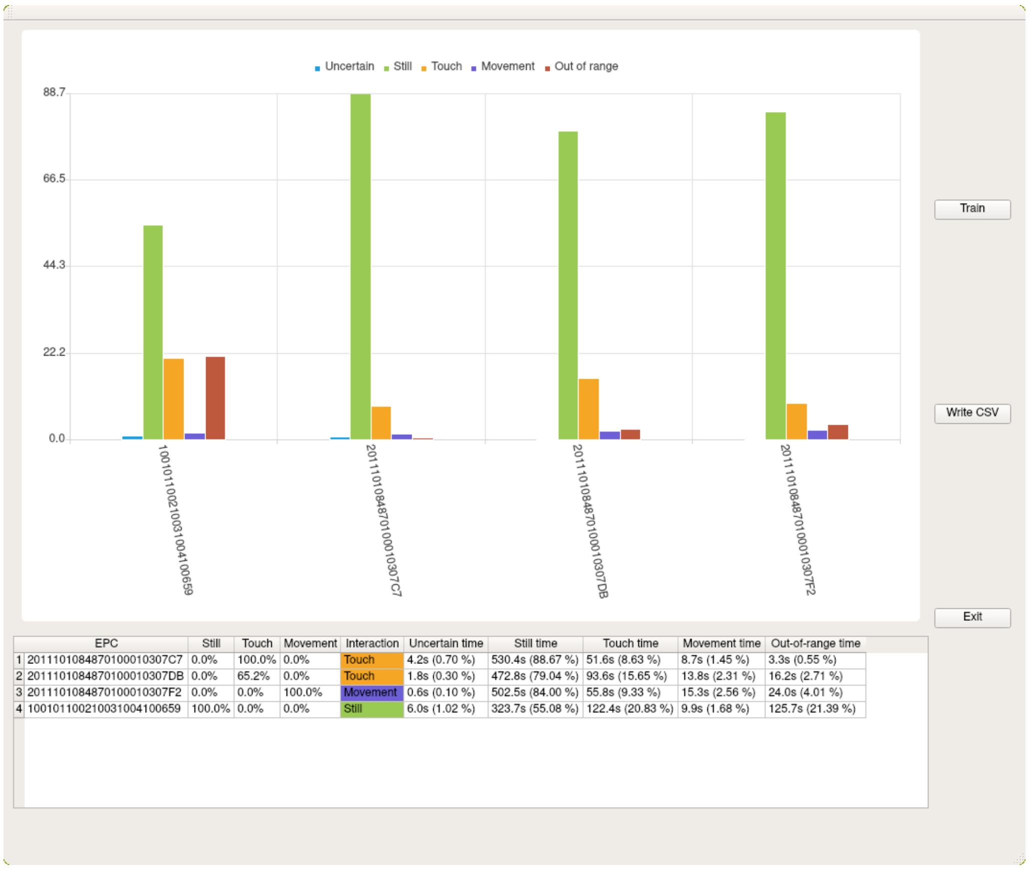
In this section are reported the last steps done in the implementation of a fully usable system for a demonstration. A GUI (Graphical User Interface) in Figure 15.13 has been written to show the data realtime and offers some practical functionalities like storing the responses on a **CSV** file for later analysis.

15.7.1 Integrating the system

The structure in Figure 15.12b has been used plus the GUI who offer a easy usable interface. The interface is in charge of:

1. Ask the user the address of the reader
2. Start the Java part and start the capture

3. Monitor the output of the Java part and read the data given
4. Send the data to the neural network
5. Fetch the responses and organize them
6. Create a table with the list of the read tags and for each of them:
    - Show the EPC
    - The network response about the current probabilities for each interaction
    - Show the most probable interaction
    - Show for how much time and in percentage with respect to the total time that interaction has been present
    - Create a plot using the previous computed percentages



	EPC	Still	Touch	Movement	Interaction	Uncertain time	Still time	Touch time	Movement time	Out-of-range time
1	2011101084870100010307C7	0.0%	100.0%	0.0%	Touch	4.2s (0.70 %)	530.4s (88.67 %)	51.6s (8.63 %)	8.7s (1.45 %)	3.3s (0.55 %)
2	2011101084870100010307DB	0.0%	65.2%	0.0%	Touch	1.8s (0.30 %)	472.8s (79.04 %)	93.6s (15.65 %)	13.8s (2.31 %)	16.2s (2.71 %)
3	2011101084870100010307F2	0.0%	0.0%	100.0%	Movement	0.6s (0.10 %)	502.5s (84.00 %)	55.8s (9.33 %)	15.3s (2.56 %)	24.0s (4.01 %)
4	100101100210031004100659	100.0%	0.0%	0.0%	Still	6.0s (1.02 %)	323.7s (55.08 %)	122.4s (20.83 %)	9.9s (1.68 %)	125.7s (21.39 %)

Figure 15.13: The user interface. **blue** is the “uncertain” response, **green** is “still”, **orange** is “touch” and **violet** is “movement” and **brown** is “out of range”

15.7.2 Data visualization

Each time a new tag is read, it is added both in the table and in the plot and its status is constantly updated. In case it is not read anymore goes into the state “Out of range”. The total time (in seconds) upon which the percentages for each interaction are computed starts from zero when a tag is first read. The system needs to be left working for a couple of minutes before the percentages stabilize. For the tests the values are updated each 0.3 seconds and the system has been left running for around 30 minutes. Some lag of around one seconds is present between the interaction and the result printed on screen. Different tag models perform in different ways: some react faster, others are more sensible, other are almost “insensible” to a specific interaction resulting in different plot shapes.

15.7.3 Final results

For the next steps, some carefully chosen samples have been used to train the network. Around 90 samples has been used as a training set and 190 as validation set. The error rate achieved settles at around 4% and an uncertain rate of less than 1%. Different tags has been used at the same time to compare their responses. The antenna has been placed below a table at a few centimeters below the top made of a layer of metal and wood and the tags are on the table. This is similar situation in which the antenna is below a shelf and the objects are on top.

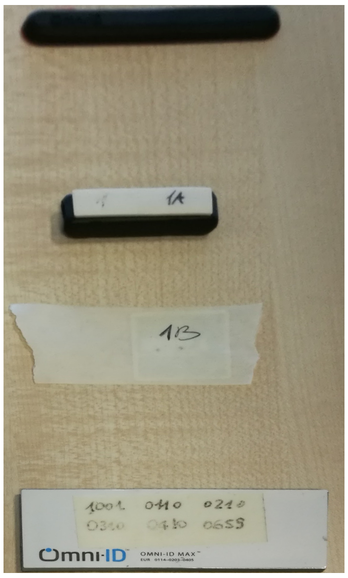


Figure 15.14: Tags used for the final tests. The last 4 numbers of the EPC are from top to bottom: 0019, 081A, 501B, 0659

"Still" status and noise

In this part the capability to distinguish between the "still" status and the "touch" status have been tested. Noise can induce a still tag to be registered as touched. The tags have been placed as in Figure 15.14 and never touched. The environment is a typical open-space office with a lot of people moving and lots of electronics devices switched on as possible noise sources.

The results given by the system are in Figure 15.16 and reported in the following table. As can be seen the results slightly differ depending on the tag.

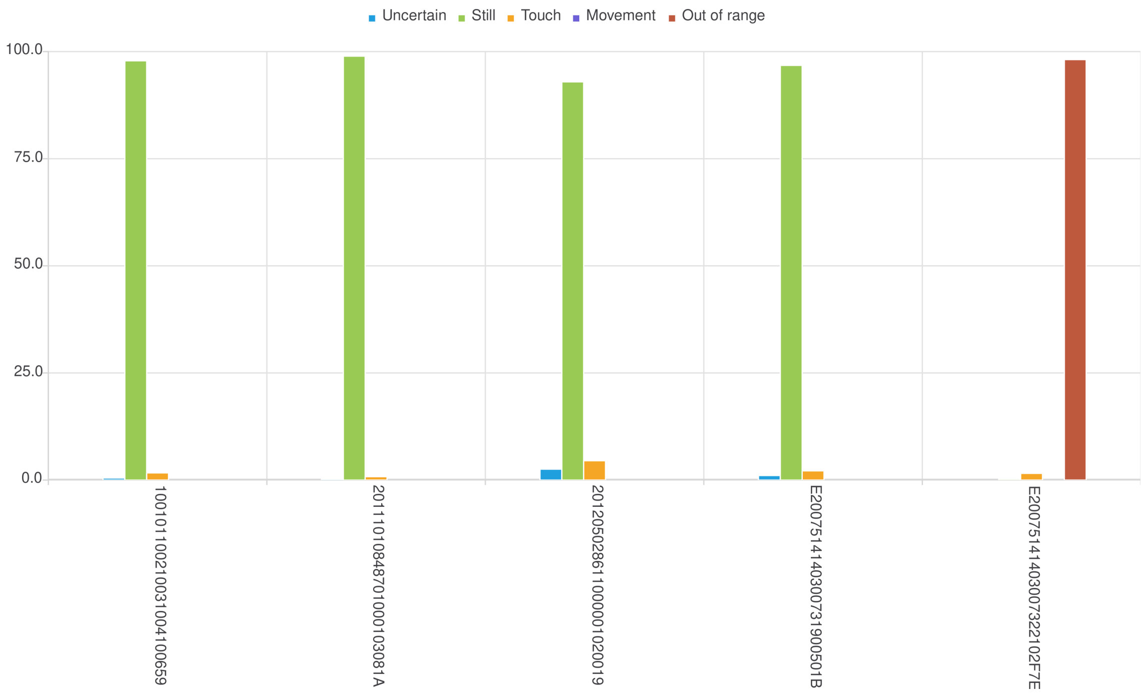


Figure 15.15: Results after around 25 minutes (plot). The last tag has been spuriously read in the distance and it is not part of the test the other tags are from left to right: 0659, 081A, 0019, 501B

EPC	Total time [ms]	Uncert. %	Still %	Touch %	Mov. %	Out of range %
100101100210031004100659	1422600.0	0.49	97.83	1.69	0.00	0.00
20111010848701000103081A	1422600.0	0.27	98.92	0.80	0.00	0.00
201205028611000001020019	1422600.0	2.57	92.87	4.56	0.00	0.00
E2007514140300731900501B	1422600.0	1.08	96.75	2.17	0.00	0.00

Table 15.1: Results after around 25 minutes in numerical form

Tag **0019** is the one that performs worse with a false positive "touch" response rate of 4.56 % and an uncertainty rate of 2.57 %. All the other tags performs significantly better with **081A** which has a precision of almost 99 %; the other tags are in between the two. The smallest tag **501B** performs quite well with an error rate of 2 % and uncertainty rate of 1.08 %: the tag has a thickness around that of a normal printer sheet of paper and a surface of 2x2 centimeters.

"Touch" status

The "touch" status has been tested waving the hand at around 5 centimeters on top of the tags. For around half of the time the tags has been "touched" for the other half they have been left on the table. This is not a precise test as the previous one: different tags performs in a different way depending on how near the proximity of the hand is and how fast the interaction is.

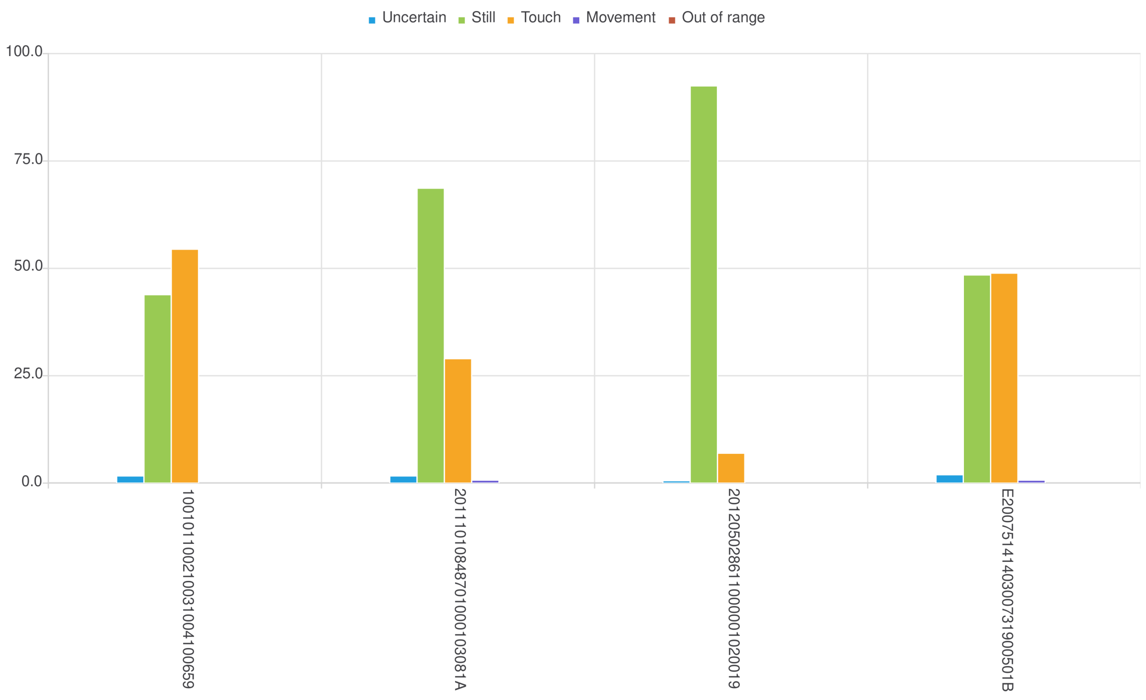


Figure 15.16: Touch interactions plot. The tags are from left to right: 0659, 081A, 0019, 501B

The tag that fails most to identify the touch interaction is **0019**: the only touch interactions detected by it where those in which the hand rested on the tag (physical contact) for more than one second. **501B** and **0659** are those who performs better while **081A** fails to detect more than 20 % of the interactions. The tags with an higher surface seems to be more sensitive to the touch interaction.

EPC	Total time [ms]	Uncert. %	Still %	Touch %	Mov. %	Out of range %
100101100210031004100659	184800.0	1.46	41.56	56.98	0.00	0.00
20111010848701000103081A	184800.0	1.79	65.75	31.66	0.81	0.00
201205028611000001020019	184800.0	0.65	92.37	6.98	0.00	0.00
E2007514140300731900501B	184800.0	2.11	45.29	51.79	0.81	0.00

Table 15.2: Touch interaction results

“Movement” status

The movement is the most difficult interaction to test due to the multiple factors involved like speed, distance, starting position equal or not to arrival position. Some approximate tries have been done: as before some tags are more sensible than others. The entity of the movement depends on the tag: **0659** is the most sensible and is able to recognize movements of a few centimeters while **0019** often reports movement events as touch events.

Final results thoughts

As seen with the previous experiments the precision of the system is deeply affected by the tags type. Different tags type can be used together but the may respond in slightly different ways. Different tags of the same model may have marginally different responses but the impact is not so important as using tag models not suitable for the general scenario. For more specific applications it is possible that a more “lazy” tag, needs a stronger interaction, is more suitable but using a general point of view, it is better to have a tag which is enough sensible even if this causes more noise, especially if the final data will be treated in a statistical way (as in the previous experiments).

Part IV

Conclusion and possible improvements

Chapter 16

Conclusion of the "proof of concept" implementation

The aim of this thesis was to create a "proof of concept" system. The system, even if fully functional, has still some rough edges to be removed. The main difficulties encountered are porting the current state of the art to the European frequencies and implementing a fully functional artificial intelligence so to be able to tune and identify the best settings for it.

Some "quick" implementations have been used in order to obtain a very flexible and easily modifiable system (training part in section 15.6), but a complete reference system has been developed as a starting point for the final product (normal operation in section 15.6).

The reference system proves the feasibility of the methodology explained in the previous parts but keeps some minor issues that need a more specific attention in order to be solved. The suggestions in this part are based on of those issues found while implementing the "proof of concept" and are an attempt to further refine the system.

This suggestions can be categorized as:

1. Software related
    - SDK $\rightarrow$ AI communication
    - AI $\rightarrow$ User/Database communication
    - Using a cloud service
2. Hardware related
    - Multiple readers and multiple antennas extensions
    - Choice of the reader
    - Tags model and placing on the object
3. General
    - Increasing precision
    - Real scenario data (related to the previous point)
    - Extending to more specific human-object interactions

Chapter 17

Software improvements

Octane SDK is available in both Java or C# versions. I have chosen Java because it is historically more "cross-platform" than C# and in some scenario could be necessary to run it on a smaller PC or board which is not X86 like an ARM processor. C# has became cross platform initially thanks to the Mono project [14] and later on with the open-sourcing of Microsoft .Net framework [45] but I have not tested if the C# Octane works on other platforms.

The performances of a final implementation in both languages should be equivalent [8, 51] and enough for the purpose. Some tests should be done with a very high number of tags to check if the system keeps an acceptable level of responsiveness.

17.1 SDK → AI communication

As said in section 15.6 a file in **\tmp** folder has been used. In *nix system the **\tmp** folder is usually a *tmpfs* filesystem: it is considered has a normal hard-disk partition but it is stored in RAM, so read/write operations are faster and do not require disk access [35]. This insert a bit of overhead which is unnoticeable for a small number of tags but can lead to excessive memory usage if not managed correctly. Other interfaces are generally faster like the pipes in a *nix environment.

Artificial networks are usually written in lower level languages, usually C++, and have bindings for some higher level languages like Python. This fact implies the need of transferring data from the SDK to the artificial intelligence for processing.

17.1.1 Using pipes

Pipes or pipelines are generally used in *nix system for communication across different processes. They are one-directional so one process writes and the other reads: to have a communication in the opposite direction another pipes has to be created.

Pipes are generally faster then files and have been specifically created to link multiple processes/programs. From a programming point of view, they are like normal files but can be read or written only sequentially. This is the fastest solution to avoid using a file for the SDK → AI communication.

17.1.2 Access external libraries

Using this approach is possible to obtain a single executable that contains both the SDK and the AI or, alternatively, call a library directly without the need of executing two different programs that need to communicate.

Java

Java can call or be called by an external library/program written in another language. Various libraries exist like JNI (Java Native Interface) [26] and JNA (Java Native Access) [25] that are in charge of passing functions calls and data between different programs/libraries. This allows two possible results:

- The Java program is embedded inside a lower level language program (like C++)
- The C++ program executes the Java program and directly calls Java function to get the data from the SDK

C#

C# allows to call functions from an external library (*.dll* files) compiled from C or C++ code [9].

17.2 AI → User/Database communication

In the "proof of concept" the response of the AI has been initially printed on the terminal as a text message and then shown on a real time updated table and plotted. In a real system the data should be permanently stored for statistical/historical analysis. Due to the high amount of data a dedicated database can be used.

High level programming languages like C# and Java have an easy interface to the database while languages like C++ may need some external library (SOCI [54] or a more specific MySQL library [40]).

17.3 Offline processing

A different approach, suitable for the scenario of the system, is offline processing. The raw or pre-processed data can be directly stored in a database. In a later phase, like during the night when the retail/store is closed, the data is read from the database and processed or alternatively can be processed only when needed. The scheme could be SDK → Database1 ... Database1 → AI → Database2. This choice uses the database as interface between the SDK and the AI and separate the processing from the accumulating of the data.

17.4 Using a cloud service

All the software parts of the system can be moved entirely or partially on the cloud. The SDK can connect to a reader over the internet (public IP or better a VPN). We can assume

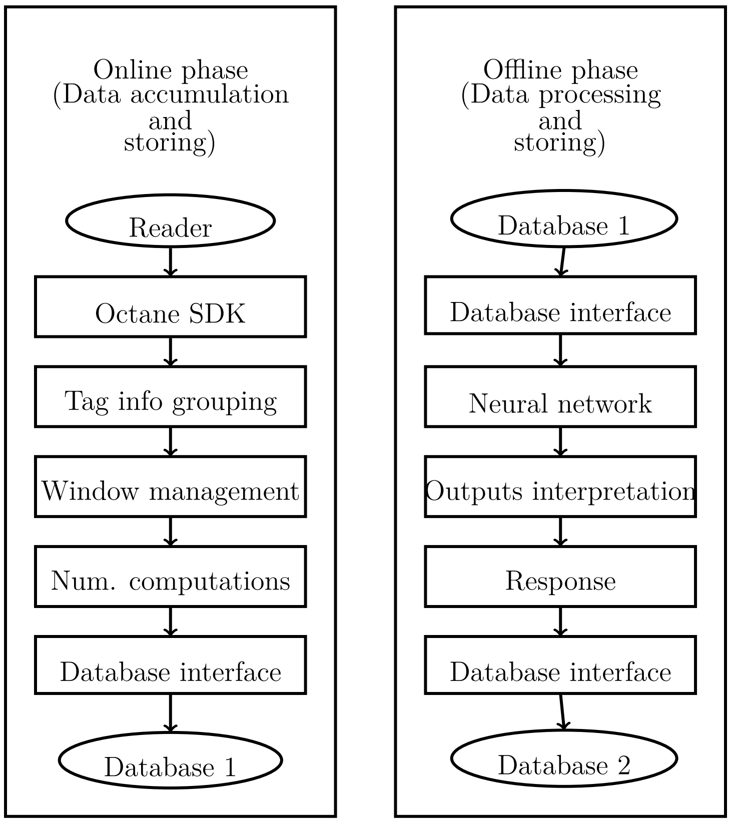


Figure 17.1: Proposed structure for offline processing.
The two phases are done at different times and are completely separated

that the system is constituted of multiple independent part that can be or not transferred on a cloud service. This results in different configuration:

- All cloud (SDK, AI, Database)
- Cloud processing (AI) and database; SDK run locally
- Cloud processing (AI) using commercially available services; SDK and database run locally

While considering a cloud solution, it is important to take into account the confidentiality of the data, if any, and the various technical problems involved in a remote connection.

The main advantage of a cloud solution in this scenario is the possibility to accumulate massive amounts of data useful to further refine the system and offer a better precision.

Chapter 18

Hardware choice and placing

This chapter is dedicated to different hardware configurations than the one used. Using a single reader, single antenna configuration is enough for a proof of concept but in a real scenario, where the space to monitor is more extended and the presence of various materials reduce the reading range, it is required the presence of multiple antennas and multiple readers.

18.1 Multiple antennas and multiple readers

Connecting multiple antennas to the same reader usually causes no problems, but inserting multiple readers in an environment results in reader-reader interference. Particular attention should be paid in this case especially on the distance between antennas connected to different readers: if the antennas are too near with each other their reading range is heavily reduced. This effects are deeply analyzed and the results presented in [27]: the numerical results may different in Europe since the paper uses the North American and Korean regulations but the general concept should subsist.

Various antennas models and brands are available: they differ not only in performances but in shape and size. The reader usually supports the connection of different antenna models at the same time but choosing a single model is still better for data congruence.

One other important fact which is related to antennas is the antenna-reader cable length: the power of the reader should be modified according to the cable loss (section 6.4) which is related to the length. Longer cables reduces the reading range at the same reader transmitting power. To uniform all the antennas connected, the reader can be placed at the center of the total range so to have cables of similar lengths.

Using multiple antennas, which is a forced choice for very extended shops, can give better precision if some precautions are taken as in section 15.5: each antenna can be considered as a different “point of view” which allows to differentiate between environment changes/interference and actual interaction.

The idea to use multiple antennas to increase precision and range can be found in various papers about localization using passive UHF RFID tags. Even if the scenario is quite different, the concept can be applied for the system designed in this thesis. Some examples are:

- «Relative Localization of RFID Tags using Spatial-Temporal Phase Profiling.» [52] which uses some fixed antennas and a moving one to compute the relative position of the tag
- «A 2D localization technique for UHF-RFID smart bookshelves» [7] similar to above but specialized on bookshelves
- «UHF RFID localization system based on a phased array antenna» [31]
- «Analysis of low range Indoor Location Tracking techniques using Passive UHF RFID tags» [11] which combines 4 antennas and an AI to compute the position of a tag
- «UHF RFID shelf solution with cascaded reader antenna and positioning capability» [66] which uses custom made antennas embedded in shelves to identify in which shelf a tag is
- «Enhancing localization accuracy with multi-antenna UHF RFID fingerprinting» [1]

18.2 Reader characteristics

A reader suitable for the task should:

- Be as much noise resilient as possible
- Be able to run for extended periods of time
- Give the minimum amount of data: EPC, RSSI, phase
- Be able to frequency hop
- Be fast enough to capture normal interaction: at least a sample per second per tag
- Be able to support the connections of more antennas at the same time and keep the data obtained from each one of them separated

Due to the problems in the previous section the best choice is a reader which support as more antennas as possible while keeping an acceptable reading time. Using less readers, which are the most expensive part, reduces the cost of the whole system.

18.3 Tags model

Different tags behave differently: this depends mostly on their type (flexible, rigid) and their size. In particular, the size of a tag is directly related with the surface of its antenna: wider tags are usually more sensible but, when a lot of tags are near with each other, it is possible that they react to interactions of their neighbors. On the other hand, smaller tags are more suitable to be kept at low distance with each other. A good compromise depends on the specific case.

An already trained network can easily adapt to different kinds of tags: the tests has been done with an AI trained with data from wide rigid tags and with around 10 samples taken with the different tags, the system has been able to adapt to the changes without losing the ability to recognize the interactions with the old ones.

If the number of the tag types increases, it is still advised to keep a separate neural network trained with each tag type than depending on the read EPC send the data to the appropriate network.

18.4 Tags placement on objects

In some cases the placement of the tag on an object can enhance its readability. This choice is relevant only for boxed objects which are usually placed on the shelves in a particular way.

In shelves which have the antennas on the sides, a good tag placement can be on one side of the boxes, while in those which have the antennas under the shelf it is better to place the tag under or on top of the box.

The rule of thumb is to reduce the tag-reader distance. Similar considerations should be done when it is possible to know in advance how the final product will be placed for selling: a “wall of glasses or sunglasses” is another example.

Chapter 19

General suggestions

This chapter groups some general thoughts and ideas to increase the precision of the system and extend the initial general scenario to more specific cases.

19.1 Increasing precision

This part extends the previous suggestions about precision (multiple antennas, more training data, ...).

19.1.1 Reference tags

As in section 18.1, the idea of placing reference fixed tags comes from some papers about localization. Attaching some tags to the shelves or in the same environment can help in filtering out the environment effects. Since the tags are very cheap and requires no maintenance after placing this idea is easily applicable. Some example of using reference tags are:

- «Exploiting phase measurements of EPC Gen2 RFID tags» [23] uses reference tags to delineate a zone in which an unknown position tag can be located
- «Real time location system using passive UHF RFID» [6] similar to the previous one
- «Indoor trace tracking algorithm with pattern-based positioning technique of UHF band RFID» [58] uses some pre-positioned tags to compute the position of the reader
- «Similarity Analysis-Based Indoor Localization Algorithm With Backscatter Information of Passive UHF RFID Tags» [67]

19.1.2 Real scenario data

During the experiment the training data has been taken by me, a single person. To further extend the system multiple people should produce different samples. The problem of different subjective "ways of interaction" has been identified in the current state of the art [34].

As a starting point, multiple people should have natural interactions with the objects which have to be stored and accumulated for learning purposes.

Another doable solution can be to use a cloud platform, as in section 17.4, and identify a suitable unsupervised learning algorithm for the AI to accumulate real usage data directly on the field.

19.2 Extending to more specific human-object interactions

For the proof of concept, the system has been trained to distinguish among three interactions: still, touch and movement which can be considered as generic. The current state of art [34], proposed a house scenario in which the system is able to recognize the specific interaction with an object like "drink milk" or "read a book". The system proposed in this thesis is affected by a loss of accuracy due to the more strict European regulations which is partially compensated by a more refined AI. This loss can be solved using multiple antennas to gain more precision; some other choices to increase precision are present in the literature like combining RFID data with cameras [64], but this is not really applicable in a shop as said in section 3.2.

Some tests should be done in more specific scenarios to check if the precision "as is" is enough to distinguish to more subtle differences among similar interactions. An example of possible interactions in a more specific case like a wall of glasses, already mentioned in section 18.4, can be:

- Still: the glasses are in their normal position
- Touch: a person touches the glasses or moves them slightly without removing them from the support
- Remove: a person takes the glasses
- Wear: a person wears the glasses

The difference between "Still" and "Wear" is definitely greater than "Still" and "Touch", so not only the general precision needs to be checked but the ability to classify different interactions which cause little changes in the raw data.

To summarize, the system can be easily extended to recognize more or more object/scenario specific interactions but particular care should be given to check if the technology allows to discern similar reads into different classes.

The system can use different AIs depending on the object. The process should be as follow:

1. Get raw data
2. Check EPC of the tag
3. The EPC is associated to a particular object category
4. Send the raw data to the AI associated to that particular category

5. Get response

Keeping a small AI, as the one proposed here, in memory is doable because of its relatively low memory footprint and allows to have a specific network for each category and use the most suitable one when needed.

Bibliography

[1] Alejandro Aguilar-Garcia et al. «Enhancing localization accuracy with multi-antenna UHF RFID fingerprinting». In: *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on*. IEEE. 2015, pp. 1–9.

[2] B. S. Ãiftler, A. Kadri, and I. Guvenc. «Experimental performance evaluation of passive UHF RFID systems under interference». In: *2015 IEEE International Conference on RFID Technology and Applications (RFID-TA)*. Sept. 2015, pp. 81–86. DOI: 10.1109/RFID-TA.2015.7379802.

[3] *Amazon Rekognition*. URL: https://aws.amazon.com/rekognition/ (visited on 08/09/2017).

[4] *Auto-vectorization in GCC*. URL: https://gcc.gnu.org/projects/tree-ssa/vectorization.html (visited on 08/13/2017).

[5] *Bluetooth Specification*. URL: https://www.bluetooth.com/specifications/bluetooth-core-specification/legacy-specifications (visited on 08/10/2017).

[6] Dean Brennan and Jan Kolaja. «Real time location system using passive UHF RFID». In: *Control Conference (ICCC), 2014 15th International Carpathian*. IEEE. 2014, pp. 58–62.

[7] Alice Buffi and Paolo Nepa. «A 2D localization technique for UHF-RFID smart bookshelves». In: *Antennas and Propagation (APSURSI), 2016 IEEE International Symposium on*. IEEE. 2016, pp. 1159–1160.

[8] *C .NET Core programs versus Java*. URL: http://benchmarksgame.alioth.debian.org/u64q/csharp.html (visited on 08/31/2017).

[9] *Calling Native Functions from Managed Code*. URL: https://msdn.microsoft.com/en-us/library/ms235282.aspx (visited on 08/31/2017).

[10] J. Canny. «A Computational Approach to Edge Detection». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (Nov. 1986), pp. 679–698. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1986.4767851.

[11] Arunabh Chattopadhyay and Ayyangar R Harish. «Analysis of low range Indoor Location Tracking techniques using Passive UHF RFID tags». In: *Radio and Wireless Symposium, 2008 IEEE*. IEEE. 2008, pp. 351–354.

[12] J. S. Choi et al. «Localization Systems Using Passive UHF RFID». In: *2009 Fifth International Joint Conference on INC, IMS and IDC*. Aug. 2009, pp. 1727–1732. DOI: 10.1109/NCM.2009.198.

[13] *Convolutional Neural Networks for Visual Recognition*. URL: http://cs231n.github.io/ (visited on 08/17/2017).

[14] *Cross platform, open source .NET framework*. URL: http://www.mono-project.com/ (visited on 08/31/2017).

[15] John Duchi, Elad Hazan, and Yoram Singer. «Adaptive subgradient methods for online learning and stochastic optimization». In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159.

[16] *EPC UHF Gen2 Air Interface Protocol*. URL: https://www.gs1.org/sites/default/files/docs/epc/Gen2_Protocol_Standard.pdf (visited on 08/11/2017).

[17] *ETSI EN 302 208*. URL: http://www.etsi.org/deliver/etsi_en/302200_302299/302208/03.01.00_20/en_302208v030100a.pdf (visited on 08/13/2017).

[18] *Feedforward network*. URL: https://www.theodysseyonline.com/amazing-world-of-neural-networks (visited on 08/17/2017).

[19] *Google Vision*. URL: https://cloud.google.com/vision/ (visited on 08/09/2017).

[20] Wataru Hattori et al. «Near-field RFID sensor-sheets to detect objects and persons with No RFID tags attached». In: *Microwave Conference (APMC), 2014 Asia-Pacific*. IEEE. 2014, pp. 486–488.

[21] Kaiming He et al. «Deep Residual Learning for Image Recognition». In: *CoRR* abs/1512.03385 (2015). URL: http://arxiv.org/abs/1512.03385.

[22] M. Honkanen, A. Lappetelainen, and K. Kivekas. «Low end extension for Bluetooth». In: *Proceedings. 2004 IEEE Radio and Wireless Conference (IEEE Cat. No.04TH8746)*. Sept. 2004, pp. 199–202. DOI: 10.1109/RAWCON.2004.1389107.

[23] Jordy Huiting et al. «Exploiting phase measurements of EPC Gen2 RFID tags». In: *RFID-Technologies and Applications (RFID-TA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 1–6.

[24] Sergey Ioffe and Christian Szegedy. «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift». In: *CoRR* abs/1502.03167 (2015). URL: http://arxiv.org/abs/1502.03167.

[25] *Java Native Access*. URL: https://github.com/java-native-access/jna (visited on 08/31/2017).

[26] *Java Native Interface*. URL: https://docs.oracle.com/javase/8/docs/technotes/guides/jni/ (visited on 08/31/2017).

[27] D. Y. Kim et al. «Effects of Reader-to-Reader Interference on the UHF RFID Interrogation Range». In: *IEEE Transactions on Industrial Electronics* 56.7 (July 2009), pp. 2337–2346. ISSN: 0278-0046. DOI: 10.1109/TIE.2009.2012451.

[28] Diederik P. Kingma and Jimmy Ba. «Adam: A Method for Stochastic Optimization». In: *CoRR* abs/1412.6980 (2014). URL: http://arxiv.org/abs/1412.6980.

[29] Günter Klambauer et al. «Self-Normalizing Neural Networks». In: *CoRR* abs/1706.02515 (2017). URL: http://arxiv.org/abs/1706.02515.

[30] Manikanta Kotaru et al. «SpotFi: Decimeter Level Localization Using WiFi». In: *SIGCOMM Comput. Commun. Rev.* 45.4 (Aug. 2015), pp. 269–282. ISSN: 0146-4833. DOI: 10.1145/2829988.2787487. URL: http://doi.acm.org/10.1145/2829988.2787487.

[31] Rainer Kronberger et al. «UHF RFID localization system based on a phased array antenna». In: *Antennas and Propagation (APSURSI), 2011 IEEE International Symposium on*. IEEE. 2011, pp. 525–528.

[32] Solomon Kullback and Richard A Leibler. «On information and sufficiency». In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[33] S. Lanzisera, D. Zats, and K. S. J. Pister. «Radio Frequency Time-of-Flight Distance Measurement for Low-Cost Wireless Sensor Localization». In: *IEEE Sensors Journal* 11.3 (Mar. 2011), pp. 837–845. ISSN: 1530-437X. DOI: 10.1109/JSEN.2010.2072496.

[34] Hanchuan Li, Can Ye, and Alanson P Sample. «IDSense: A human object interaction detection system based on passive UHF RFID». In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM. 2015, pp. 2555–2564.

[35] *Linux docs about tmpfs*. URL: https://www.kernel.org/doc/Documentation/filesystems/tmpfs.txt (visited on 08/31/2017).

[36] Andreas Loeffler and Heinz Gerhaeuser. «Localizing with passive UHF RFID tags using wideband signals». In: *Radio Frequency Identification from System to Applications*. InTech, 2013.

[37] David G Lowe. «Three-dimensional object recognition from single two-dimensional images». In: *Artificial intelligence* 31.3 (1987), pp. 355–395.

[38] Yuan-Ping Luh and Yin-Chang Liu. «Measurement of effective reading distance of UHF RFID passive tags». In: *Modern Mechanical Engineering* 3.03 (2013), p. 115.

[39] Jose Marroquin, Sanjoy Mitter, and Tomaso Poggio. «Probabilistic solution of ill-posed problems in computational vision». In: *Journal of the american statistical association* 82.397 (1987), pp. 76–89.

[40] *MySQL connector/c++*. URL: https://dev.mysql.com/doc/connector-cpp/en/ (visited on 08/31/2017).

[41] Vinod Nair and Geoffrey E Hinton. «Rectified linear units improve restricted boltzmann machines». In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[42] Mian Hammad Nazir and Nathirulla Sheriff. «Dynamic grouping frame-slotted aloha». In: ().

[43] P. V. Nikitin et al. «Phase based spatial identification of UHF RFID tags». In: *2010 IEEE International Conference on RFID (IEEE RFID 2010)*. Apr. 2010, pp. 102–109. DOI: 10.1109/RFID.2010.5467253.

[44] *Octane SDK*. URL: https://support.impinj.com/hc/en-us/articles/202755268-Octane-SDK (visited on 08/14/2017).

[45] *Official home of .NET on GitHub*. URL: https://github.com/microsoft/dotnet (visited on 08/31/2017).

[46] R. Parada et al. «Using RFID to Detect Interactions in Ambient Assisted Living Environments». In: *IEEE Intelligent Systems* 30.4 (July 2015), pp. 16–22. ISSN: 1541-1672. DOI: 10.1109/MIS.2015.43.

[47] M. Periyasamy and R. Dhanasekaran. «Evaluation of performance of UHF passive RFID system in metal and liquid environment». In: *2015 International Conference on Communications and Signal Processing (ICCSP)*. Apr. 2015, pp. 0414–0417. DOI: 10.1109/ICCSP.2015.7322921.

[48] Lanxin Qiu et al. «Multifrequency phase difference of arrival range measurement: principle, implementation, and evaluation». In: *International Journal of Distributed Sensor Networks* 11.11 (2015), p. 715307.

[49] M. ScherhÃ¤ufl, M. Pichler, and A. Stelzer. «UHF RFID Localization Based on Evaluation of Backscattered Tag Signals». In: *IEEE Transactions on Instrumentation and Measurement* 64.11 (Nov. 2015), pp. 2889–2899. ISSN: 0018-9456. DOI: 10.1109/TIM.2015.2440554.

[50] Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert. *Kernel methods in computational biology*. MIT press, 2004.

[51] Peter Sestoft. «Numeric performance in C, C# and Java». In: *IT University of CopenhagenDenmark, Version 0.9* 1 (2010).

[52] Longfei Shangguan et al. «Relative Localization of RFID Tags using Spatial-Temporal Phase Profiling.» In: *NSDI*. 2015, pp. 251–263.

[53] B. Shrestha, A. Elsherbeni, and L. Ukkonen. «UHF RFID Reader Antenna for Near-Field and Far-Field Operations». In: *IEEE Antennas and Wireless Propagation Letters* 10 (2011), pp. 1274–1277. ISSN: 1536-1225. DOI: 10.1109/LAWP.2011.2174603.

[54] *SOCI - The C++ Database Access Library*. URL: https://github.com/SOCI/soci (visited on 08/31/2017).

[55] *Speedway R420 installation manual*. URL: https://support.impinj.com/hc/article_attachments/115001207664/Impinj_SpeedwayR_installation_and_operations_guide.pdf (visited on 08/13/2017).

[56] Tijmen Tieleman and Geoffrey Hinton. «Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude». In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[57] Qiaoling Tong et al. «Bayesian estimation in dynamic framed slotted ALOHA algorithm for RFID system». In: *Computers Mathematics with Applications* 64.5 (2012). Advanced Technologies in Computer, Consumer and Control, pp. 1179–1186. ISSN: 0898-1221. DOI: http://dx.doi.org/10.1016/j.camwa.2012.03.060. URL: http://www.sciencedirect.com/science/article/pii/S0898122112002659.

[58] Toshinori Tsuboi and Hiromi Ueda. «Indoor trace tracking algorithm with pattern-based positioning technique of UHF band RFID». In: *Communications (APCC), 2013 19th Asia-Pacific Conference on*. IEEE. 2013, pp. 339–344.

[59] V. Viikari, P. Pursula, and K. Jaakkola. «Ranging of UHF RFID Tag Using Stepped Frequency Read-Out». In: *IEEE Sensors Journal* 10.9 (Sept. 2010), pp. 1535–1539. ISSN: 1530-437X. DOI: 10.1109/JSEN.2010.2045497.

[60] Hui Wang et al. «Group improved enhanced dynamic frame slotted ALOHA anti-collision algorithm». In: *The Journal of Supercomputing* 69.3 (Sept. 2014), pp. 1235–1253. ISSN: 1573-0484. DOI: 10.1007/s11227-014-1189-7. URL: https://doi.org/10.1007/s11227-014-1189-7.

[61] *Welford online algorithm*. URL: https://en.wikipedia.org/wiki/Algorithms_for_calculating_variance#Online_algorithm (visited on 08/29/2017).

[62] B. P. Welford. «Note on a Method for Calculating Corrected Sums of Squares and Products». In: *Technometrics* 4.3 (1962), pp. 419–420. DOI: 10.1080/00401706.1962.10490022. eprint: http://amstat.tandfonline.com/doi/pdf/10.1080/00401706.1962.10490022. URL: http://amstat.tandfonline.com/doi/abs/10.1080/00401706.1962.10490022.

[63] N.J. Woodland and S. Bernard. *Classifying apparatus and method*. US Patent 2,612,994. Oct. 1952. URL: http://www.google.com/patents/US2612994.

[64] J. Wu et al. «A Scalable Approach to Activity Recognition based on Object Use». In: *2007 IEEE 11th International Conference on Computer Vision*. Oct. 2007, pp. 1–8. DOI: 10.1109/ICCV.2007.4408865.

[65] J. Yang and Y. Chen. «Indoor Localization Using Improved RSS-Based Lateration Methods». In: *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*. Nov. 2009, pp. 1–6. DOI: 10.1109/GLOCOM.2009.5425237.

[66] Yong Yuan and Dan Yu. «UHF RFID shelf solution with cascaded reader antenna and positioning capability». In: *RFID (RFID), 2012 IEEE International Conference on*. IEEE. 2012, pp. 149–156.

[67] Yang Zhao et al. «Similarity Analysis-Based Indoor Localization Algorithm With Backscatter Information of Passive UHF RFID Tags». In: *IEEE Sensors Journal* 17.1 (2017), pp. 185–193.