M 2014
I
E
I
C

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

OPTIMIZED MOBILE SYSTEM FOR SEAMLESS INDOOR/OUTDOOR POSITIONING

RUI FILIPE DIAS VALENTE MAIA

DISSERTAÇÃO DE MESTRADO APRESENTADA
À FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO EM
MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Optimized Mobile System for Seamless Indoor/Outdoor Positioning

Rui Filipe Dias Valente Maia

Master in Informatics and Computing Engineering

Supervisor: Prof. Rosaldo José Fernandes Rossetti

Co-Supervisor: Dr. Marcelo Petry

20th January, 2014

Abstract

There are several approaches that attempt to minimize the Distance Error that is obtained by a mobile device when a position is inferred in an indoor environment, although none of them reaches full accuracy. Due to this fact, sundry different kinds of techniques have been developed in order to achieve this goal. In this document, 3 Machine Learning algorithms were chosen, developed and tested with the same final objective: an Artificial Neural Network, a Support Vector Machines algorithm and a k-Means Clustering approach.

Tests were structured using 2 scenarios with different configurations and sizes, using 2 different kinds of training – Static or Continuous -, using different numbers of Training Examples and using different numbers of Measured Positions for each one of the algorithms implemented. Due to the huge amount of tests to perform and the high necessary time taken to do it, some tests' exclusion criteria was established up to some experiments.

Results were compared and conclusions were taken relatively to the several hypothesis formulated, that included the study of the influence of the number of Training Examples from the same Measured Positions in the Mean Distance Error value, the influence of the number of Measured Positions in the Mean Distance Error value, different correlations between the statistics produced by the algorithms' results, comparisons between the implemented algorithms and the approach that does not use any Artificial Intelligence, some comparisons between algorithms and comparisons with the results found in the Literature Review phase.

The best result achieved was through the ANN approach in the first testing scenario (approximately 19m² of area) with a Mean Distance Error value of 1.23m, using 150 Training Examples and the Static Training configuration, representing an improvement of above 78 % relatively to the approach that does not use any Artificial Intelligence methodology to infer positions; in the second testing scenario (approximately 239m²) the best result achieved were also the ones of the ANN algorithm, but this time using 3 Measured Positions, the Static Training configuration and 150 Training Entries, with a Mean Distance Error value of 1.44m, representing an improvement of more than 139% of the positioning accuracy in relationship to the approach that does not implement any Artificial Intelligence.

Resumo

Várias são as abordagens que tentam minimizar a Distância Média de Erro que é obtida aquando da inferência da posição de um dispositivo móvel num ambiente *indoor*, apesar de nenhuma delas atingir total exatidão. Devido a este facto, diversas e diferentes tipos de técnicas foram estudadas e desenvolvidas com a finalidade de atingir esse objetivo. Neste documento, 3 algoritmos de *Machine Learning* foram escolhidos, desenvolvidos e testados com a mesma meta: uma Rede Neuronal Artificial, um algoritmo de Support Vector Machines e uma abordagem utilizando k-Means Clustering.

Os testes foram estruturados usando 2 cenários com configurações e tamanhos diferentes, utilizando 2 tipos diferences de treino – Estático e Contínuo -, com recurso a valores variados de Exemplos de Treino e usando diferentes quantidades de Posições Medidas para cada um dos algoritmos implementados. Devido à quantidade enorme de testes a fazer e ao elevado tempo necessário para os fazer, alguns critérios de exclusão de testes foram estabelecidos a partir de um determinado número de experiências.

Os resultados foram comparados e foram retiradas conclusões relativas às várias hipóteses formuladas, que incluíram o estudo da influência do número de Exemplos de Treino usando as mesmas Posições Medidas no valor da Distância Média de Erro, a influência do número de Posições Medidas na Distância Média de Erro, diferentes correlações entre as estatísticas obtidas dos resultados dos algoritmos, comparações entre os algoritmos implementados e a abordagem que não utiliza Inteligência Artificial, algumas comparações entre os algoritmos e comparações com os resultados utilizados na fase de Revisão Bibliográfica.

O melhor resultado obtido no primeiro cenário de teste (de aproximadamente 19m²) foi através da abordagem de Redes Neuronais Artificiais, com uma Distância Média de Erro de 1.23m, utilizando 150 Exemplos de Treino e configurada em modo Treino Estático, representando uma melhoria de mais de 78% relativamente à abordagem que não utiliza Inteligência Artificial para inferir posições; no caso do segundo cenário de teste (de aproximadamente 239m²), o melhor resultado obtido foi também através do algoritmo de Redes Neuronais Artificiais, desta vez utilizando 3 Posições Medidas, a configuração de Treino Estático e 150 Entradas de Treino, obtendo uma Distância Média de Erro de 1.44m, uma melhoria de mais de 139% na exatidão do posicionamento relativamente à abordagem que não implementa nenhum método de Inteligência Artificial.

Acknowledgements

I'll start to thank Prof. Rosaldo Rossetti, for all the orientation given, all the motivation offered and all the effort put towards the quality of my work in the last 10 months of my life; I wanna thank also Dr. Marcelo Petry, for guiding me towards smarter decisions when the time was running low. Special thanks to Bruno Fernandes, CEO of LatitudeN and Luís Ramos, CTO of LatitudeN, for the daily accompaniment, feedback, advice and specially for making me feel motivated in the most lazy hours. I want to thank also the remaining employees of LatitudeN for the example they set in my daily activity in this company: Cláudia Delgado and Vítor Ribeiro, more than a pleasure to be at work with you everyday, it's been a pleasure to be your friend. You're both awesome. Thank you, Prof. Jaime Villate, for the good influence and interest in the well-being of his students and for the continuous support given to all of us in the last years. Thank you, Prof. António Augusto Sousa, for all the effort put in the high quality of our education. One of the wisest men I knew through my whole life once told me: “Those who have friends will never die in jail”. Hence, I'd like to thank my hometown friends, all those guys and girls that make out of Café Juventude their second house in our idle hours. I want to thank also all my college friends for the constant inspiration you always passed to me with your opinions, your conversations and your assertiveness in all our discussions. You guys make me feel proud that I'm becoming an engineer. I also want to thank all the good friends I met in my Erasmus in Vienna and in my internship here in Darmstadt for allowing me to believe and understand that international borders are just lines in a paper (specially to Anja, Leoncini, Mohamed, Giorgio, Marta, Martina, Davinia, Laura, Rob, Riccardo, Francisco, Rui, José Paulo, Pedro, André, Mehdi, Somaya, Bruno Nanni, João Mura, Estevão, Kazhal, Malte, Maxime, Nzo, Mateus and Rafa). I want to thank to Ilinca Fage for all the friendship, support and good influence given in the last years and Helena Lima for the longest friendship I maintain since the old days of basic school. Special thanks to Anupma Raj, for all the advice given, all the support in the most difficult hours and all the good laughs and conversations in the happiest ones. To my family members that with their constant and unconditional love all the time made me the person that I am nowadays: Manuela Maia, my mother; Emídio Maia, my father; Conceição de Magalhães, my grandmother: Fernando Rodrigues Valente, my grandfather; Gil Valente Maia, my brother, the most intelligent young man I know and one of the strongest examples I follow in life; Cacilda de Magalhães, my great-grandmother. Special thanks to a person that, although he's not a member of my family I feel like he is: José Fernando Moreira da Silva, thank you for throughout all these years keeping on redefining and improving the old and complicated concept of friendship and dedication. Couldn't have finished this without your knowledge, advice and support. Lastly, I wanna thank and dedicate this thesis to the wisest man I have met in my life, the same one that told me the saying that this paragraph starts with: Henrique Alves de Magalhães, my beloved great-grandfather, you would be so happy to see an engineer in your family if you were here. This one's for you.

Contents

List of Figures

List of Tables

Abbreviations

ANN	Artificial Neural Network
AoA	Angle-of-Arrival
AP	Access Point
APos	Admissible Position
BN	Bayesian Network
CT	Continuous Training
EPErr	Estimated Position Error
EPos	Estimated Position
GNSS	Global Navigation Satellite System
kNN	k-Nearest-Neighbor
LErr	Literature Error
LMC	Large Margin Classifier
LSVM	Location-based Support Vector Machine
MDE	Mean Distance Error
MErr	Maximum Error
ML	Machine Learning
MLP	Multi-layer Perceptron
MP	Measured Position
PCA	Principal Components Analysis
PM	Probabilistic Model
RSSI	Received Signal Strength Indicator
RTLS	Real-Time Location System
ST	Static Training
SVM	Support Vector Machines

1 Introduction

1.1 Context/Scope

This document is in the scope of the final thesis of the Integrated Master in Informatics and Computation Engineering at Faculdade de Engenharia da Universidade do Porto (FEUP). The presented project was proposed by LatitudeN GmbH, a start-up company based in Darmstadt, Germany, which the main focus is the development of mobile applications for tourism. The company has been developing products targeting outdoor environments, specially cities' urban areas, but they also aim to develop applications focused on the indoor positioning inference and that's the reason why this thesis proposal came up.

This proposal appeared as part of an international project that is currently being developed by LatitudeN GmbH and is the continuation of a previous thesis proposal done by the company and held also in the scope of the final thesis of the Integrated Master in Informations and Computation Engineering. This previous work was mainly focused on implementing a mobile application that gathered data related with indoor positioning, differentiating between the real mobile device position and the inferred one. Several tests were done in different environments, which contributed for the data set to have a big diversity of data that needs to be treated, analyzed and classified in order to minimize the distance error of the current mobile indoor positioning systems.

The need to obtain a generalized solution for the indoor mobile positioning constraints has enabled several researchers to study, implement and test different approaches to the problem. Still, up to the date of this thesis, there is no optimal solution that solves the distance error constraint between the real position where the mobile device holder is and the inferred position by the same device. This work does not intend to provide a general optimal solution for the problem, but to follow approaches suggested by experienced researchers in the field and implement, test and analyze them in order to understand which could be the add-ons that will

contribute to a future improvement of the current solution and to conclude about several aspects related with the positioning techniques and the application scenarios.

To approach the error minimization problem, the main focus will be on the application of Machine Learning (ML) techniques to the existing data set so there's a smarter observation of the existing patterns, the establishment of comparisons between several different approaches to this constraint in order to ease future work and to infer conclusions related with the algorithm's performance, the best training characteristics, the statistics generated and the application scenarios.

1.2 Motivation and Goals

In the past few years, the automatic location of people or devices has been an increasing target of research and the rising number of established methods and developed applications related with this theme have ignited more interest in this research field.

Although outdoor localization has been achieved several years ago using a Global Navigation Satellite System (GNSS), the suggested methods for indoor localization are not yet satisfactory due to high dependency on the mobile devices' sensors, high dependency on the technology application scenario's obstacles (walls, people moving and change of obstacles' location contribute for the signal impoverishment) and the lack of accuracy in the positioning inference.

The current thesis is part of a big international project where LatitudeN GmbH is an active entity. Due to the fact that one of the goals of this work is to improve the accuracy of the existing positioning methods, the main motivations to finish with success this thesis is to contribute with real data, tests and comparisons for the research related with the minimization of error between the real mobile device position and the position inferred by the device.

Henceforth, the purpose of this project is to study the best solution to solve the error constraint in these systems, through the implementation, test and evaluation of different ML algorithms, in order to produce a solution that can be applied in real scenarios.

Thus, the specific goals for this project are:

- Make a State-of-the-art analysis in error minimization techniques oriented to indoor mobile devices' positioning, focusing on the capabilities of *smartphones* and *tablets* to infer it.
- Evaluate possible solutions for the error minimization problem and choose the approaches that show the best results as the ones to implement and test.
- Implement the set of those algorithms that prove more accuracy in the positioning inference.
- Test those implementations, produce new results, correlate them and conclude about the generated hypothesis.
- Test this set of methods in real scenarios, gather results and make comparisons between them in order to ease future work.

1.3 Dissertation Structure

Beyond this introductory chapter, this document contains 3 more chapters. In chapter 2 it is explained the ML algorithms and methods that will be taken into account for the purposes of this study and reviewed the State-of-the-art of the ML techniques that minimize the distance error of the mobile device positioning. In chapter 3 there's a detailed explanation of the hypothesis to prove or disprove, the statistical metrics that will be used to evaluate the implemented algorithms, the description of the project's requirements, of the implemented algorithms and of the functioning of the whole software. Chapter 4 is where the tests done are described and correlated in 2 different scenarios using the 3 implemented algorithms with different training configurations. Chapter 5 is where the findings about those correlations between algorithms, training configurations and performance are detailed.

2 Literature Review

In this chapter, it will be presented each one of the state-of-art approaches found throughout the study of the solutions for the problem. The first sub-chapter reviews the existent positioning methods and the second sub-chapter considers the existent distance error minimization approaches using ML techniques, whereas each of them are divided in two more sub-sections: one explaining the technique itself and one revising the different applications of the algorithm applied towards this thesis' solution evaluation.

2.1 Introduction

In the beginning of the thesis proposal, some important decisions were done in order to bounder the signal transmission technologies, the Positioning Techniques to be applied, the Error Correction approaches and the Scenarios where a possible final application would be used. Hence, it was defined that the most important signal transmission technologies were Wi-Fi and Bluetooth – due to the low cost of the technologies and mass usage worldwide -, the Positioning Technique to be studied was the trilateration technique – due to the fact that it was the technique analyzed and chosen in [C12] -, that the Error Correction approaches that were most valuable for this context were ML techniques and that the testing scenarios would be either places with big areas, wide spaces with several walls and columns and a lot of people moving from place to place and small areas where the obstacles attenuate the signal more intensively. These definitions influenced the main focus of the whole Literature Review process.

During this process, already taking into account the above specified theme constraints, in order to select the approaches who fit into the kind of problem this thesis intends to solve, each article was classified respecting three different metrics: Positioning Technique, Error Correction Technique and Testing Scenario.

The classification for each one of these three metrics was done using three different symbols: √, ~ and x. Each one of these symbols represents the importance for this study of each one of these three criteria in each of the analyzed articles. For instance, an article that was classified with Positioning Technique = x, Error Correction = x, Testing Scenario = x is considered an article with less probability to be selected for intensive study (and consequently, less probable for the approach explained in the article to be applied in the context of this thesis) than an article that was classified with Positioning Technique = √, Error Correction = √ and Testing Scenario = √. This was done exclusively to select relevant information in this problem's context, due to time constraints both in the Literature Review as in the Implementation, Testing and Analysis phases.

Summarizing, x means “not important”, ~ means “may be important”, √ means “important” for each of the above explained metrics for each of the analyzed bibliography items. Some other criteria may be added in some comparisons done throughout the document, but the classification symbols are maintained nevertheless.

2.2 Positioning Methods

Knowing which is the position of a mobile device is a task that can be achieved using several different positioning methods, such as Lateration, Angulation, Scene Analysis and Proximity methods [LDBL07]. The Lateration and Angulation are the ones explained in the next sub-sections, because of the following reasons: first, Lateration was the method chosen by [C12] and due to this thesis time constraints it will be the method used for the purposes of positioning inference in the work done ahead; second, because Lateration and Angulation are the two main localization algorithms [WBLP09] and are computationally cheaper than Scene Analysis and the Proximity methods [PC11].

The combination between Lateration and Angulation is feasible and although it would bring better performance to the system, it would add a lot more complexity and would require higher processing power [C12]. Hence, no combination between them was implemented during the period in which this dissertation has been developed. Still, both methods will be explained in this sub-chapter, creating enough documentation for future improvements.

2.2.1 Lateration

The Lateration technique is a derivation of Triangulation. The first goal of the Lateration technique is to estimate the distance from the mobile device to the Access Point. The high sensitivity of the RSSI-based location tracking methodologies is due to environmental changes. Signal strength varies over time, even if the mobile device doesn't change its position, due to its dynamic nature. Hence, the implemented Lateration technique applies a low complexity

smoothing algorithm, where the basic assumption of this algorithm is that the constant velocity motion will result on constant data change rate [LC07].

Due to the fact that in the current project the Lateration technique needs at least 3 Access Points to infer position, the kind of implemented Lateration technique is called trilateration [FRM13]. Trilateration is a conjunction of three distances between transmitters and one receiver that provide position. There are diverse possible scenarios for trilateration to infer position: the best case scenario is when the three circumferences intercept at one point and there's a scenario where the interception of the three circumferences is an area [YYC11].

According to [C12], one of the most difficult decisions taken during his thesis development was the one of which propagation model to use in order to calculate distances based on the RSSI metric. The propagation model chosen was the *Hata-Okumura* model, expressed by the Equation 2.1:

$$\log d = \frac{1}{10\mathrm{n}}\left(P_{TX} - P_{RX} + G_{TX} + G_{RX} - X_{\alpha} + 20\log\lambda - 20\log(4\pi)\right) \tag{2.1}$$

Because the signal measurements have a really high variance, a signal variance attenuation formula was introduced, based on a weighted Median method:

$$M_{final} = M_1 * 0.02 + M_2 * 0.04 + M_3 * 0.06 + M_4 * 0.08 + M_5 * 0.1 + M_6 * 0.14 + M_7 * 0.14 + M_8 * 0.14 + M_9 * 0.14 + M_{10} * 0.14 \tag{2.2}$$

Three types of lateration algorithms were tested by [C12] and the one chosen was the Linear Least Squares (LLS) algorithm. It starts with the equation 2.3, that is the equation of circle:

$$(x - x_i)^2 + (y - y_i)^2 = r_i^2 \tag{2.3}$$

where $i = 1,2,3,\ldots,n$ is the *ith* Access Point for the calculations and r_i is the estimated distance from that Access Point to the mobile unit. In the case of the implemented algorithm, it's actually prepared for the introduction of any amounts of Access Points, although in the current dissertation 4 APs where used and the minimum of 3 are needed. Then, the LLS algorithm obtains a linearized system of the form $Ax^r = s^r$ of (n-1) equations where:

$$A = \begin{matrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ x_n - x_1 & y_n - y_1 \end{matrix} \tag{2.4}$$

$$x^r = [x - x_1 \; y - y_1]^{-1} \tag{2.5}$$

$$s^r = \frac{1}{2}[r_j^2 - r_i^2 + d_{ij}^2] \tag{2.6}$$

$$d_{ij} = \sqrt{((x_i - x_j)^2 + (y_i + y_j)^2)} \quad (2.7)$$

Still, instead of solving directly the linear system, it's used a normalized QR decomposition of A. This method combines the values obtained in x^r and s^r and A = QR where Q is an orthogonal matrix and R is an upper triangular matrix. The equation 2.8 determines the solution:

$$Rx^r = Q^T s^r \quad (2.8)$$

2.2.2 Angulation

To use Angulation it's only required to get two reference points and measure their Angle-of-Arrival (AoA), with the need to know at least one length measure such as the distance between the two Access Points [Pai11].

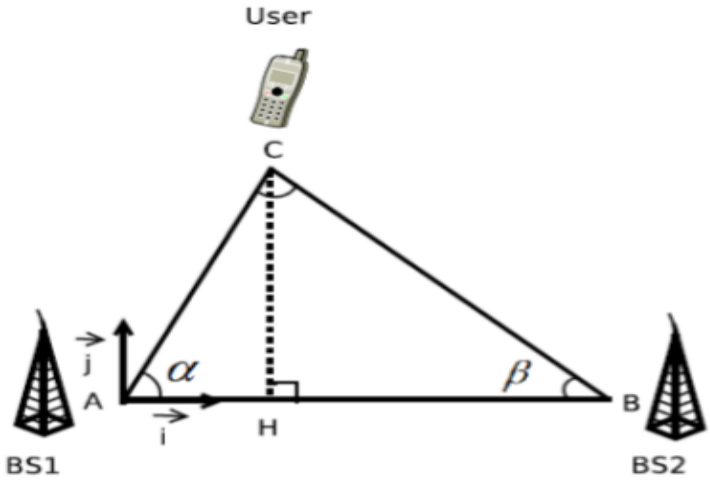

Figure 2.1: Angulation method representation. Source: [Pai11].

Ubisense is an example of an AoA-based location sensing system [WOK13]. The increased complexity and the hardware requirements are the main hindrances for the wide success of such systems [PaC11].

2.3 Machine Learning Algorithms

Carbonell, Michalski and Mitchell [CMM83], in 1983, defined Machine Learning as *"a many-faceted phenomenon that includes the acquisition of new declarative knowledge, the development of motor and cognition skills, the organization of new knowledge into general, effective representations, and the discovery of new facts and theories through observation and experimentation"*.

In fact, it's through the organization of new knowledge into general and through the discovery of new facts and theories, through observation and experimentation, that the main problem that this thesis is approaching can be solved, or at least get to an improved solution in relationship to the existing ones. Therefore, there is no better way to approach the constraint of error minimization in indoor positioning using mobile devices than using ML techniques.

The current sub-chapter explains the most used ML approaches that try to bring accuracy to indoor positioning using mobile devices and explains several experiences and results obtained by authors who have done work in this research field. This will be done detailing the most common techniques and algorithms with the goal of bringing scientific sustainability to the methodology that will be elaborated in order to try to improve the existing solutions for the current thesis' constraint.

2.3.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational methodologies inspired by networks of biological neurons that perform multi-factorial analysis. They contain layers of simple computing nodes that operate as nonlinear summing devices and rich connections between the nodes. The connection lines are weighted and these weights are adjusted when data is presented to the network, during a “training” process” that, if it ends up being successful, can result in the neural network to perform tasks such as predicting an output value, classifying an object, approximating a function, recognizing a pattern in multi-factorial data and completing a known pattern [DLD01].

ANNs are organized into layers of processing units, where the units of a layer are similar in the sense that they all have the same activation dynamics and output function. Connections between these units can be intra-layer – between units of the same layer –, inter-layer – between units of different layers – or both intra-layer and inter-layer (see Figure 2.2). There are methods to implement the learning feature of an ANN, leading to several learning laws that use local information for adjusting the weight of the connection between two units [Yeg06].

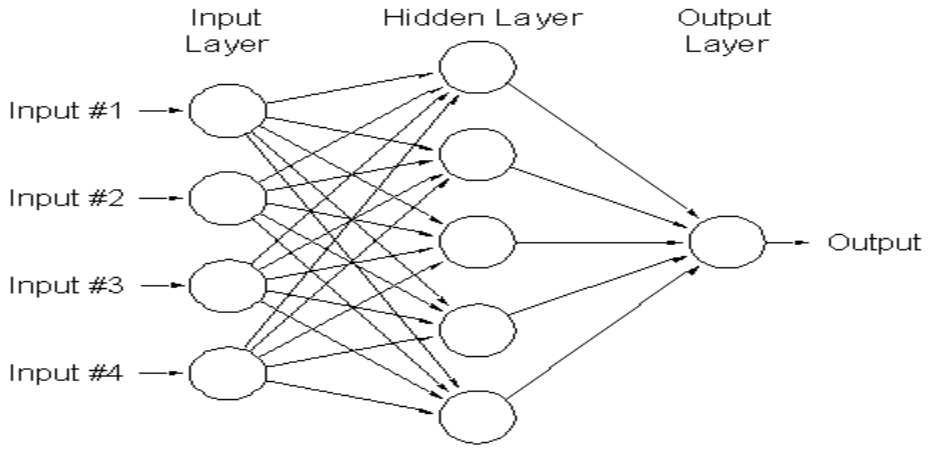

Figure 2.2: Artificial Neural Network representation.

Generally, obtaining indoor positioning using ANNs is done by adopting the RSSI values as inputs and the positioning coordinates as targets for the training purposes. After the training phase, appropriate weights are obtained. Usually, a MLP with one hidden layer is the kind of ANN chosen for a neural-networks-based positioning system and its output is a two-element vector or a three-element vector representing the inferred or estimated position [LDBL07]. However, in the case of this dissertation, the structure of the ANN will possess a different configuration from the ones studied, depending on the features used and on the results of the processing of the best configuration parameters.

1 Using Wi-Fi

It's possible to use an ANN as algorithm for the inference of indoor positions, instead of using Lateration, Angulation or any other mathematical models that allow the determination of the position from the received signals. On the other hand, it's possible also to use a combination of Lateration and/or Angulation with an ANN approach [TBPA11].

Tapia, Bajo et al. [TBPA11], in 2008, developed a *multi-layer perceptron* with the goal of enhancing the performance of Real-Time Location Systems (RTLS), using ANNs to mitigate the ground reflection effect and calculate the position of a mobile device. Basically, *Tapia, Bajo et al.* structured and developed an ANN with RSSI readers as input nodes in the input layer, and each of the *perceptrons* used one output node that indicated the distance. The performance of the ANN was tested in a 19m x 19m monitored area with 3 rooms and they compared the results obtained with this multi-layer *perceptron* model with a linear regression model, a logarithmic model and a Support Vector Regression model. They concluded that the ANN model is the one that contained the lower Mean Distance Error (MDE).

Lin and Lin [LL05], in 2005, wrote a comparison article based on their experiences where they focused on comparing 3 different algorithms: a Bayesian Network (Probabilistic Model), a *k-Nearest-Neighbour* algorithm and an ANN. They benchmarked the performance of each one of these algorithms using accuracy, precision, complexity, robustness and scalability as the chosen criteria. The experiments done show that the algorithm who performs better in an overall evaluation is the *k-Nearest-Neighbour*, followed by the ANN and the Bayesian Network, respectively.

Mehmood, Tripathi and Tipdecho [MTT10], in 2010, experimented an indoor positioning system using an ANN and the Back-Propagation algorithm, in a 9mx8m room full of obstacles and compared the Distance Error (DE) results with the DE of a probabilistic algorithm. They concluded that the ANN using Back-Propagation is the methodology with the smaller DE between the estimated position and the real mobile device position (the MDE for the ANN is of 1.43m and the interval of [*minimumDistanceError*, *maximumDistanceError*] for the ANN is of [0.7, 2.06], while for the probabilistic model the average distance error is of 2.54 and the interval of [*minimumDistanceError*, *maximumDistanceError*] is of [1.04, 4.37]) .

Gholami, Cai and Brennan [MCB12], in 2012, presented a document where the approach was to infer positioning for a mobile device in a 3.5mx1.4m environment, using *trilateration* to estimate positioning and compared results with an ANN fed exactly with the same data points. They concluded that an ANN technique can infer the position of a static object with higher accuracy than *trilateration* alone, despite all the noise sources in a real-world environment. They also concluded that the minimum DE using *trilateration* is 0.058802m and the average error using an ANN is of 0.0098m, corroborating the previous conclusions.

Table 2.1: ANN references using Wi-Fi comparison.

Reference	Positioning Technique	Error Correction Technique (s)	Scenario (dimension, rooms, obstacles)
[TBPA11]	√	√	~
[LL05]	√	√	x
[MTT10]	√	√	~
[MCB12]	√	√	x

In the first and third articles in the Table 2.2, the scenarios have acceptable dimensions and they possess several obstacles that attenuate the signal received by the mobile devices in each of them. Taking into account that the studied results show that the smaller the area of a testing scenario, the lower the MDE, the main goal of this project is to test intensively in a big-sized scenario. That's why the ~ classification for the first and third articles; the second article does not disclaim in which kind of scenario the tests to the proposed solution were done, therefore the x classification was applied; in the fourth article the tests were done in a small dimension scenario with few obstacles, so x was the classification attributed to the Scenario criteria.

Henceforth, the implementation of ANN using the Wi-Fi technology may use the suggested approaches in the first and third articles, although some changes in the configurations of the specified ANN structures may be introduced and/or added some algorithm in order to improve its functioning.

2 Using Bluetooth

Bluetooth technology is widely implemented in mobile devices, therefore they're cheap and widely used and it allows distances to be estimated by link quality within a Bluetooth covered area around 30~40 meters, approximately. However, there are restrictions related with

compatibility problems or service restrictions in terms of the implementation of a Bluetooth service. Still, due to this technology's high commercial success, it's expected that in a near future there will be a standardized Bluetooth service [Gen05].

Fonseca, Neves and Ralha [FNR11], in 2011, wrote a case-study that compared results between the Wi-Fi, Bluetooth and ZigBee approaches using the accuracy criteria and trying to minimize the distance error through an ANN. The experiments were done in one floor of the SG11 building in the University of Brasília Campus and they concluded that the technology that allows better accuracy is the Wi-Fi technology, followed by ZigBee and Bluetooth respectively. The improvements done in the accuracy percentage using the ANN were of 17% with the Wi-Fi protocol, 11% with the Bluetooh protocol and 21% with the ZigBee protocol.

Altini et al [ABFB10], in 2012, proposed a low-cost *Bluetooth*-based localization system based on multiple ANNs and achieved 90% precision and 0.5m of accuracy during a walk in an indoor environment. The experiments done showed that depending on the user's orientation, the RSSI values change completely and that's the main factor for the necessity of implementing more than one ANN. They also made comparison tests with only one ANN and showed that a multiple ANN approach has a higher accuracy than using only one ANN, due to the fact stated above. The MDE of the walk in the indoor corridor is of 0.6m.

Li, Liu et al [LLCL12], in 2012, developed a mathematical model using two ANNs, one to obtain a feasible solution and one to improve the previous solution. The experiments done were in two large environments: the first was done in a tunnel – wide environment without too many obstacles – and the second done in a 60mx60m supermarket – wide environment with the presence of several obstacles. They defined Beacon phones as mobile devices with GPS devices and Blind phones as mobile devices without GPS devices, but both equipped with Bluetooth devices. Although the experiences done are fully detailed, it wasn't done any results comparison relatively to accuracy in these environments.

Table 2.2: ANN references using Bluetooth comparison.

Reference	Error Correction Technique(s)	Scenario (dimension, rooms, obstacles)
[FNR11]	√	~
[ABFB10]	√	~
[LLCL12]	√	√

Although several references that use ANN and the Bluetooth technology were studied and classified according to the reference classification criteria established, no implementation using Bluetooth was done.

2.3.2 Support Vector Machines

Support Vector Machines (SVMs) are usually applied to perform binary classification and regression estimation and it's currently the most popular approach for "off-the-shelf" supervised learning. This means that for people who don't have any specialized prior knowledge about a specific domain, then SVM is an excellent method to try first due to its cleanness, easiness to be understood and implementation and it is more powerful than the majority of the other ML techniques. Often, SVMs have the flexibility to represent complex functions, but they are resistant to *overfitting*, so they combine the advantages of nonparametric and parametric models [RN10]. The cost function that is usually applied in the SVM algorithm is:

$$J(\theta) = -y_i * \log\left(\frac{1}{1+e^{(-\theta_i^{Tx})}}\right) - (1-y_i)\log\left(1 - \frac{1}{1+e^{(-\theta_i^{Tx})}}\right); \quad (2.9)$$

where θ is the matrix that contains the values of the input features. Let $z = \theta^T x$ and $cost_1(z) = max(0, k(1-z))$, where k is an arbitrary constant defining the magnitude of the slope of the line. To regularize the cost function, we can use a factor C, where $C = \frac{1}{\lambda}$ and λ represents a regularization factor, like so:

$$J(\theta) = C\sum_{i=1}^{m} - y_i * cost_1(\theta^T x_i) - (1-y_i)\, cost_0(\theta^T x_i) + \frac{1}{2}\sum_{j=1}^{n} \Theta_j^2; \quad (2.10)$$

In the Equation 2.10, *m* is the number of Training Examples (TE) and *n* is the number of features in the dataset. When the goal is to regularize more in order to reduce *overfitting*, it's decreased C and when the goal is to regularize less – to reduce *underfitting* - it's increased C. The hypothesis function of the SVM is not interpreted as the probability of y being 1 or 0, but a discriminant function that outputs 1 or 0 [CV95]:

$$h_\theta(x) = \begin{matrix} 1 \; if \; \Theta^T x \geq 0 \\ 0 \; if \; \Theta^T x < 0 \end{matrix}$$

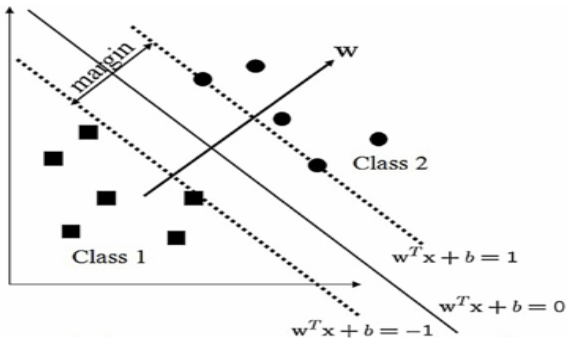

Figure 2.3: Maximum Margin Separator is the area between dashed lines.

The distance of the decision boundary to the nearest example is called *margin*. Since SVMs maximize this *margin*, they are often called Large Margin Classifiers (LMC). For better understanding, the definition of *margin* is twice the distance between the separator and the

nearest example point or, in other words, the width of the area bounded on both sides of the separator between the separator and the nearest example, like it can be seen in Figure 2.3.

However, there are two important notes to be made about how to achieve the large margin: first, the large margin is only achieved when C is very large; second, the large margin is only achieved when the data in the dataset is linearly separable and data is only linearly separable when a straight line can separate the positive and negative examples. If, for some reason, there are discrepant examples that can affect the decision boundary and they're not intended to do that, then reducing C is the solution for this problem. Thus, summarizing, increasing and decreasing C can simplify our decision boundary [FS99].

It's useful to think about SVMs as LMCs. For instance, if y = 1, then there's a need of $\Theta^T x \geq 1$ - not just $\Theta^T x \geq 0$ - and, on the other hand, if y = 0 then there's a need of $\Theta^T x \leq -1$ - not just $\Theta^T x < 0$. Now when the constant C is a very large value (for instance, 100000), the optimizing function will constrain Θ such that the equation that sums the cost of each example equals 0. Therefore, we impose constraints on the function that sums the cost of each example, such as $\Theta^T x \geq 1$ if y=1 and $\Theta^T x \leq -1$ if y = 0.

Henceforth, if C is very large there's a need of choosing Θ parameters such that:

$$\sum_{i=1}^{m} - y_i * cost_1(\theta^T x_i) - (1 - y_i) cost_0(\theta^T x_i) = 0\,; \tag{2.11}$$

which reduces the cost function to

$$J(\theta) = \frac{1}{2} \sum_{j=1}^{n} \Theta_j^2; \tag{2.12}$$

In the majority of the cases using SVMs, data that is not linearly separable in the original input space are easily separable in a higher-dimensional space. In order to solve this constraint, SVMs have the ability to embed the data into a higher-dimensional space, using a method called kernel.

The kernel method can be applied with any learning algorithms that can be reformulated to work with dot products of pairs of data points and not only with learning algorithms that find optimal linear separators. Once this is done, the dot product is replaced by a kernel function and we have a kernelized version of the algorithm [Her02].

Kernels allow the making of complex non-linear classifiers through the calculation of a function called “similarity” function [SBWA04]. The “similarity” function basically computes a new feature x_1 depending on proximity to landmarks l_1, l_2, l_3 . The similarity function can be expressed as follows:

$$f_i = similarity(x, l^{(i)}) = \exp\left(\frac{-\left(\sum_{j=1}^{n} (x_j - l_j^{(i)})^2\right)}{(2\sigma^2)}\right) \tag{2.13}$$

where σ^2 is a parameter of the *Gaussian kernel* and it can be modified to increase or decrease the drop-off of the feature f_i . The "similarity" function is also called a *Gaussian Kernel* and it represents a specific example of a kernel. There are a couple of properties for the similarity function, such as:

- if $x \approx 1$, then $f_1 = \exp\left(-\left(\frac{0^2}{(2\sigma^2)}\right)\right) \approx 1$; (2.14)
- if x is far from $l^{(1)}$, then $f1 \approx \exp\left(-\left(\frac{(large\ number)^2}{(2\sigma^2)}\right)\right) \approx 0$. (2.15)

In other words, if x and the landmark are close, then the similarity will be close to 1, otherwise the similarity will be close to 0. Each landmarks represents now the features in the hypothesis function:

- $l^{(1)} \rightarrow f_1$
- $l^{(2)} \rightarrow f_2$
- $l^{(3)} \rightarrow f_3$
- …

Hence, $h_\theta(x) = \Theta_1 f_1 + \Theta_2 f_2 + \Theta_3 f_3 + \ldots$ (2.16)

Combined with looking at the values inside Θ , we can choose these landmarks to get the general shape of the decision boundary. The easiest way to get the landmarks is to put them exactly in the same position as all the training examples. This gives us *m* landmarks, with one landmark per training example [CGGRC09]. Give an example x:

$$f_1 = similarity(x, l^{(1)}), f_2 = similarity(x, l^{(2)}), f_3 = similarity(x, l^{(3)}) \quad (2.17)$$

This allows the representation of a feature vector, for example $x^{(i)}$:

$$x^{(i)} \rightarrow \begin{matrix} f_1^i = similarity(x^{(i)}, l^{(1)}) \\ f_2^i = similarity(x^{(i)}, l^{(2)}) \\ f_3^i = similarity(x^{(i)}, l^{(3)}) \end{matrix} \quad (2.18)$$

To get the parameters Θ it can be used the SVM minimization algorithm but with $f^{(i)}$ substituted in $x^{(i)}$:

$$min_\Theta C \sum_{i=1}^{m} -y_i * cost_1(\theta^T f^{(i)}) - (1 - y_i) cost_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^{n} \Theta_j^2; \quad (2.19)$$

Lastly, because of the fact that there are implemented and tested SVM libraries, the choices that are needed to put the SVM algorithm running are mainly about the *C* factor value and the choice of the kernel: if a standard linear classifier is needed (when the *n* – number of features – is large and *m* – number of training examples – is small), then the right choice is no

kernel at all; if there's a need to choose a *Gaussian Kernel* (when *n* – number of features – is small and *m* – number of training examples – is large) it'll be necessary to choose also the σ^2 parameter.

1 Using Wi-Fi

Brunatto, Battiti and Villani [BBV05], in 2005, developed a discovery technique based on SVMs that estimated the location of a mobile user through a classification engine to decide the area that the user is currently in. They experimented their engine in a 35x25m scenario, with several different rooms, and concluded that the SVM approach they used bested all the other approaches they compared with, due to the fact that the SVM had an average of 3.04m error distance from the real position.

Pan, Kwok and Yan [PKY06], in 2006, developed a method using a *Gaussian Kernel* (based on SVMs technique) for a signal space to adapt to the noisy characteristics of radio-propagation channels and a different type of kernel called *Matérn kernel* for the physical location space. The methodology proposed by the combination of these two kernels showed results of improvement in comparison with the typical SVM approach. The experiments were done in a 30x37.5m environment with different rooms of different sizes and the accuracy of the best combination of kernels is of 81.7 percent when the acceptable error distance is of 1.5m.

Sun, Chen et al. [SCQL08], in 2008, proposed an algorithm that combines *Laplacian graphs*, *dimensionality reduction* and SVMs to train out the data set. The experiments were done in an area of 30x15m covering a hallway and five rooms and the results obtained clearly show that the algorithm proposed, in comparison with other approaches that use only SVMs, has a better performance over time.

Figuera, Rojo-Alvarez et al [FAWJC12], in 2012, published a paper that combined a-priori information with a supervised learning technique based on SVMs. The tests were done in a scenario with 43x13m of area with several different rooms and several obstacles and the comparisons were done between SVM approaches with different kinds of *kernelization* algorithms and it was proven that a-priori information can enhance the performance of positioning systems. The results are not shown using the accuracy percentage, but using intervals of MDE standard deviation, uncertainty and bias as comparison metrics.

There are several publications that address the Wi-Fi based indoor positioning with the SVM algorithm to train data and help in the inference of a more accurate positioning, each one of those with their own *kernelization* methods and optimization techniques. The best of them all show results that will serve as comparison between the approach that will be implemented in the context of this thesis and tests in different kinds of scenarios.

Table 2.3: SVM references using Wi-Fi comparisons.

Reference	Error Correction Technique(s)	Scenario (dimensions, rooms,	MDE (m)

		obstacles)	
[BBV05]	√	√	~
[PKY06]	√	√	√
[SCQL08]	~	√	x
[FAWJC12]	√	√	√

The first article in the table is classified with √ in Error Correction Technique(s) and Scenario because it fills the expected criteria for the current thesis and as ~ in the MDE criteria because 3.04 is still a high error value.

The second and fourth articles are the only studied articles that satisfy all the criteria, due to the fact that the Error Correction Technique(s) are easy to implement, the scenarios are big enough and the MDE results are quite good to solve the problem this thesis is aiming to.

The third article has a ~ classification in the Error Correction Technique(s) criteria, due to the fact that the implemented algorithm looks complex and time would be required to do it properly, therefore, due to the time constraints of the development of this thesis, using the proposed approach may not be the best option. It also does not show MDE results.

2 Using Bluetooth

Tran and Nguyen [TN08], in 2008, proposed an algorithm called *Location based on Support Vector Machines* (LSVM) that offers fast localization in a distributed manner based on mere connectivity information and uses the SVM approach for classification. LSVM assumes the existence of a number of beacons and uses them as training data to the learning process. The experiments were done in a 50x50m scenario and the results show that the MDE decreases with the increase of the number of beacons' percentage .

Park, Patel et al. [PPCTL12], in 2012, proposed an approach for device pose classification and walking speed estimation based on SVMs, with the goal to learn the walking speed of a mobile device user and where the device was located in the body of the user. Since the problem's context is related with the user's movement and not exactly with positioning, the results presented are in m/s, representing the user's walking speed. It was obtained a [0.22, 0.3] mean error interval between the two tested approaches and the testing environment was a large office area with several rooms.

Table 2.4: SVM references using Bluetooth comparisons.

Reference	Error Correction	Scenario (dimensions,	Mean Distance Error

	Technique(s)	rooms, obstacles)	(m)
[TN08]	√	√	~
[PPCTL12]	√	√	~

In the first article in Table 2.4, the Error Correction Technique(s) and Scenario were classified as √ for this thesis' context, since the Error Correction Technique(s) is through the usage of an SVM to minimize the error between the real mobile device's position and the inferred position and the Scenario is a wide scenario with several obstacles. The Mean Distance Error metric though, was classified as ~ due to the fact that the article does not present numerical results regarding the DE and it concludes only that the error decreases as the percentage of beacons being used increases.

In the second article, both the Error Correction Technique(s) and the Scenario were classified as √ since it uses SVMs to minimize the error and the scenario is a wide office with several obstacles and several rooms. The Mean Distance Error has been classified as ~, only because the article does not talk about positioning, but about the movement of a user carrying an handheld mobile device, so the information about this criteria is not applicable to this study's context.

Even though it was found references that related the SVM algorithm applied using Bluetooth technologies, no implementation was done using it.

2.3.3 Clustering Algorithms

Clustering is an unsupervised learning technique, which the main goal is the discernment of multiple categories in a collection of objects. Clustering is an unsupervised problem due to the fact that category labels are not given, thus the algorithm works with raw data and with the need of understanding what kind of probability distribution generated that data. The big difference between this method and the already studied supervised learning methods is that, while the other techniques had a labeled training set with a vector y of expected results, clustering techniques compute only a dataset of features where the goal is to find structure. The main applications of clustering are in market segmentation analysis, social network analysis, astronomical data analysis and in organizing computer clusters [WF05].

There are several kinds of these techniques, each one of those containing different algorithms that can be applied in order to fulfill its goal. There's not a general rule of which algorithm to apply to which problem, therefore the algorithm choice depends greatly on subjective criteria by who applies it. Still, there are algorithms that perform better than others when solving specific problems and that are more widely used than others [Ber06].

Nevertheless, in this section it will be detailed the *k-Means* algorithm, one of the most commonly used and well-known of the huge set of clustering algorithms and that performs well in the plurality of its applications. Despite the fact that it has been designed and published in 1955 and thousands of other clustering algorithms have been proposed since then, k-Means is still the choice of most of the data scientists all over the world every time they need to use this technique, due to its comprehension and implementation simplicity and to its capability of adaptation to a more specific application algorithm [Jai10]. Henceforth, in the context of this thesis, the application of the clustering technique will initially be based exclusively in the *k-Means* algorithm.

In order to start the *k-Means* algorithm, it's necessary to initialize 2 points in the dataset that will be called, in the algorithm context, as *cluster centroids* (step 1). There are divers articles discussing how to choose the initial points, some using heuristics to determine which number of points should be set as cluster centroids [SZQ02], others initializing a certain number of points randomly [ARS97]. For the purposes of the explanation of the algorithm in this section and for the initial implementation of the algorithm in the context of this thesis, the 2 points will be initialized randomly. However, due to the fact that *k-Means* can get stuck in local optima with the randomly generated initial 2 points, there's a method to reduce the possibility of this to happen. The method consists on verifying that there are less clusters than training examples, to pick randomly K training examples and set $\mu_1, \ldots, \mu_k$ equal to those training examples. Hence, cyclically, N times (N may be defined subjectively by the person who is implementing the method) it is done an initialization of k-Means, it is computed c and m and it is computed the distortion of $J(c,m)$ – the cost of centroid c for the m entry of the dataset. In the end, the clustering that had the lowest cost is the one chosen where the so far computed best initialization points are placed.

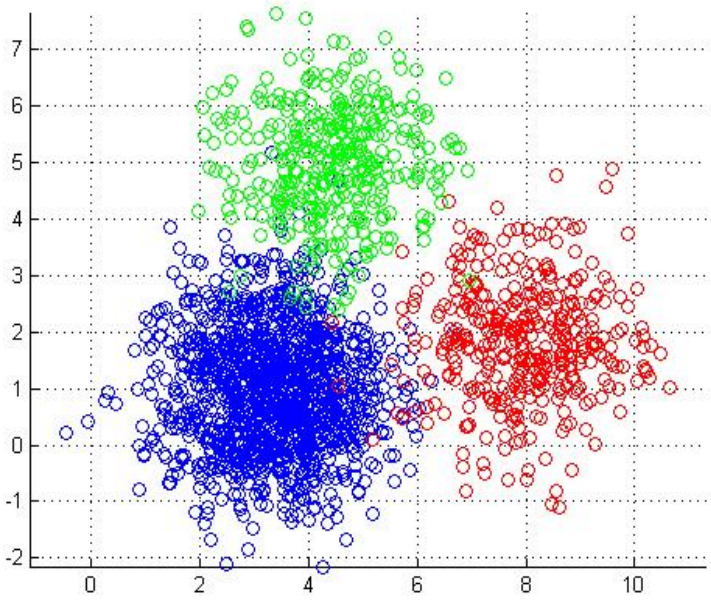

Figure 2.4: Example of data set with categorized data.

The step after the initialization is the assignment of the training examples into one of at least 2 groups, based on which cluster centroid each example is closest to (step 2). The next step is to move the centroid, computing the averages for all points inside the 2 groups and then move the cluster centroid points to those averages (step 3). The 4th step is to keep doing step 2 and step 3 until the algorithm converges and where new iterations do not affect the clusters.

Detailing the algorithm, the main variables are K, that represent the number of clusters, the training set $x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}$, where $x^{(i)} \in \mathbb{R}^n$. In step 2, it's created a vector c, where $c^{(i)}$ represents the centroid assigned to $x^{(i)}$, Then, it's computed each $c^{(i)}$ that contains the kth element that has minimal distance to $x^{(i)}$. By convention, the right-hand side of the operation is squared, which makes the minimization function to converge more sharply. Mathematically, this operation can be written as follows:

$$c^{(i)} = min_k ||x^{(i)} - \mu_k||^2 \tag{2.20}$$

Step 3 is the 'Move Centroid' step and it's done by moving each centroid to the average of its group of points. Formally,

$$\mu_k = \frac{1}{n}[x^{k_1} + x^{k_2} + \ldots + x^{k_n}], \mu_k \in \mathbb{R}^n \tag{2.21}$$

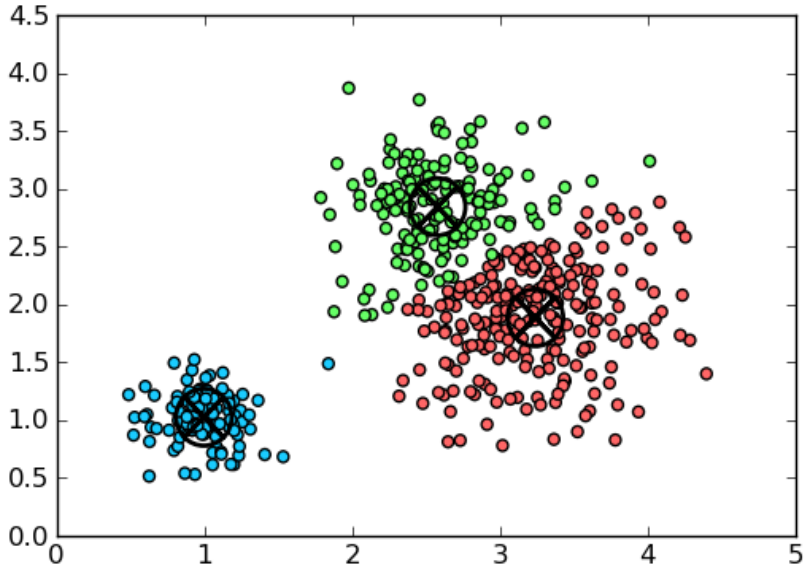

Figure 2.5: Example of data set with the cluster centroid calculated.

If there's a *cluster centroid* with zero points assigned to it, it can randomly be re-initialized the centroid to a new point and eliminated that cluster group. Some of the datasets possess no real separation or natural structure, but *k-Means* can still evenly segment the data into *K* subsets, so the algorithm can still be useful in cases like this.

The optimization objective of *k-Means* algorithm is to minimize all the parameters through the usage of a cost function, that is, to find all the values in sets c, representing all the clusters, and μ, representing all the centroids, that will minimize the average of the distances of every Training Example to its corresponding *cluster centroid*. This cost function is called the *distortion* of the training examples:

$$min_{c,\mu} J\left(c^{(i)},\dots,c^{(m)},\mu_1,\dots,\mu_k\right)=\frac{1}{m}\sum_{i=0}^{m-1}\left(\|x^{(i)}-\mu_{c^{(i)}}\|\right)^2 \quad (2.22)$$

It is clear that with *k-Means* it is not possible for the cost function to increase anytime, so it will always descend. It needs to be noted that, in step 2, or the cluster assignment step, the goal is to minimize *J(...)* with $c^{(i)},\dots,c^{(m)}$, but holding fixed $\mu_1,\dots,\mu_k$ and in step 3, or the 'Move Centroid' step, the goal is to minimize *J(...)* with $\mu_1,\dots,\mu_k$.

Choosing how many clusters to use (*K*) can be quite ambiguous and arbitrary. There are some different proposed methods to solve this constraint such as [PM00] or [TWH01], but there are others simple less time-consuming methods that can help too, like the *elbow method* [SCS06]. However, in a big number of cases, it is not possible to use this method to determine what the *K* value should be, since the curves are very gradual.

The *k-Means* algorithm has a computational cost of *O(T*k*m)*, where *T* is the number of iterations and *m* the number of objects in the input data set, so, with large data sets and when

applying a lot of iterations, the algorithm may be heavy to compute. Furthermore, there was a necessity to reduce this computational cost and make *k-Means* a faster algorithm [Hua98].

In order to do this, there are several approaches that apply the Principal Component Analysis (PCA) algorithm, with the motivation of data compression – reducing the dimension of the data set features, if there's a lot of redundant data, rather than the number of TE and consequently reducing the total data stored in computer memory – and of easiness of visualization – allowing data of more than 3 dimensions to be reduced to 3 dimensions, summarizing some features and finding some others, making it possible for the human eye to understand better the plotted data.

The PCA algorithm reads *n* features and maps them to a $\mathbb{R}^k$ space, where k is the final dimension of our feature reduction. For instance, given two features, x_1 and x_2, the goal is to find a line that effectively describes both features at once and then these old features are mapped onto this new line in order to get a new single feature. This can be done with any number of features. The goal of PCA is to reduce the mean of all the distances of every feature to the projection line, calculating the *projection error*.

Before applying PCA, feature scaling is needed, so we process the data such as:

$$\mu_j = \frac{1}{m} \sum_{i=0}^{m-1} \left(x_j^i \right) \tag{2.23}$$

and to the original feature it is subtracted its mean and then scaling is applied:

$$x_j^{(i)} = \frac{\left(x_j^{(i)} - \mu_j \right)}{s_j} \tag{2.24}$$

where s_j is the standard deviation of the element j.

The next step is to compute the covariance matrix, in order to, in the step right after, compute the eigenvectors of this covariance matrix. The covariance matrix can be computed with:

$$\Sigma = \frac{1}{m} \sum_{i=0}^{m-1} \left(\left(x^{(i)} \right) \left(x^{(i)} \right)^T \right) \tag{2.25}$$

Next, it is computed the eigenvectors of this covariance matrix and after, it's computed the vector *Z*, through the assignment of the first *k* columns of the eigenvectors matrix (*U*) to a variable called *Ureduce*. So, mathematically,

$$z^{(i)} = Ureduce^T \cdot x^{(i)} \tag{2.26}$$

where $z^{(i)}$ is the vector of the projected data points and the compressed representation is here finished. To uncompress this representation back to the original feature space, it's not possible to obtain back the original data but only approximations to the original data:

$$x_{approximation}^{(i)} = Ureduce \cdot z^{(i)} \tag{2.27}$$

Summarizing, the most common use of PCA is to speed up supervised learning. However, it's recommended to only apply PCA to a ML algorithm after it has been fully tested and shown that its performance is slow [DH04].

Clustering algorithms are an effective way of categorizing raw data. They are useful in problems such as the one that will be dealt with in the context of this thesis, since the gathered data set from previous work does not possess any classification labels. Through *k-Means* it can be obtained a simple, reliable and easily improvable solution (through extensions of *k-Means*) for the data classification problem and through this solution some initial questions may be immediately answered.

1 Using Wi-Fi

Youssef, Agrawala et al [YASN02]., in 2002, published an article where they presented two different approaches to the indoor positioning through Wi-Fi using probabilistic methods combined with a clustering technique: one called *Joint Clustering* technique, the other called *Incremental Triangulation* technique, where both techniques depend on probability distributions to handle the noisy characteristics of the wireless channel and on clustering to manage the computational cost. Experiences were done in a scenario with dimensions of 68x26m, with several rooms and several obstacles and the results show 90% of accuracy until 2.1m radius distance from any of the Access Points (APs).

Ji, Biaz et al. [JBPA06], in 2006, proposed a system that through the capture of the characteristics of a floor plan could generate a 3-D model necessary for ray tracing and through a clustering-based search algorithm was able to locate a mobile device. The experiments were done in a 43x19m environment with several rooms and several obstacles. The presented results show that the average distance error was of 2.8m using the proposed clustering technique.

Ma, Li et al. [MLTL08], in 2008, proposed an approach where clustering is used to partition in different clusters the *k-Nearest-Neighbor (kNN)* algorithm, choosing one cluster as delegate. They proved that this new approach outperforms the typical *kNN* approach. The experiments were done in an environment with several rooms and obstacles (the dimensions of the scenario were not revealed in the paper) and they show that in fifty-percent of the cases the distance error using the proposed technique varies in the interval [0m, 1.2m] and in the other fifty-percent it varies in the interval [1.2m, 2.2m].

Mengual, Marbán and Eibe [MME10], in 2010, proposed a methodology to locate mobile stations in an indoor environment using a clustering approach. Although in this particular article, the authors are not locating mobile devices, it was decided that it should be included in this section because it's being used the Wi-Fi technology to infer the position of wireless cards using clustering techniques. The experiments were done in a 300 to 500 m^2, scenario, with several different rooms and several obstacles and the presented results show 90.09 % accuracy in the position estimation after the application of the proposed clustering technique.

Altintas and Serif [AS11], in 2011, proposed a methodology similar to the one of [MLTL08], where clustering is used to partition the *kNN* algorithm in different clusters. The experiment was done in a scenario with an area of around 500 m², with several rooms and several obstacles. The results presented in the article show that the clustered *kNN* approach had a mean error of 4.11m when q = 5, 2.7m when q = 7 and of 2.68 when q = 9, where *q* represents the number of nearest neighbors computed.

Wang, Sen et al. [WSE12], in 2012, proposed an approach based on the fact that specific locations in indoor environments, such as elevators, corridor-corners and any other building's characteristic points, present identifiable signatures on one or more sensing dimensions. Three different experiments were done, all in scenarios with more than 200 m², several rooms and several obstacles and the mean location error was of 1.69m.

The above references propose some different approaches to solve the error minimization between the real position and the mobile device's inferred position constraint using Wi-Fi as positioning technology and clustering techniques as ML algorithm, although some are more important to the context of this study than others. That differentiation is done by Table 2.5.

Table 2.5: Clustering references using Wi-Fi comparisons.

Reference	**Positioning Technique(s)**	**Scenario (dimensions, rooms, obstacles)**	**Mean Distance Error (m)**
[YASN02]	~	√	~
[JBPA06]	~	√	~
[MLTL08]	√	√	√
[MME10]	~	√	~
[AS11]	√	√	~
[WSE12]	√	√	√

All the articles summarized in this section of the study have a √ classification in the scenario criteria, because all of the experiences done were in wide areas with several rooms and obstacles. The first, second and fourth articles were classified as **~** for the Positioning Technique(s) metric, due to the fact that the combination of algorithms proposed in the articles is computationally costly and of lengthy implementation or do not achieve the expected results and **~** for the Average Distance Error criteria, because the evaluation results are based on the percentage of accuracy between distances or simply because the distance error is still high and considerable for the problem. The fifth article was classified as √ for the Positioning Technique(s) and that's related with the fact that it uses the *kNN* algorithm combined with a clustering technique, which improves the efficiency of the *kNN* algorithm. It was classified as **~** for the Mean Distance Error criteria, due to the fact that the presented results show considerably

high errors that do not allow the proposed methodology to he considered a reliable one. The third and sixth articles were classified as √ for all the classification metrics, because the Positioning Techniques of both of them are simple to implement and the performances, as the MDE criteria shows, are good enough allowing a small DE for both of them, closer to the optimal indoor positioning inference than any of the other studied and referenced approaches.

2 Using Bluetooth

Throughout the phase of Literature Review, no publications were found that related the Bluetooth technology applied in indoor positioning systems using mobile devices and clustering techniques to minimize the error distance between the real position and the inferred position of the mobile device. Hence, the Bluetooth approach was also not implemented during this dissertation.

2.3.4 Summary

In this Literature Review chapter, the main goal was to detail and explain all the techniques, algorithms and methodologies that may be useful to approach the seamless indoor positioning using mobile devices error minimization problem using ML techniques. The chapter starts by making an introduction to the classification criteria for each one of the analyzed articles, in order to help to understand their value for the solution and to establish importance comparisons between each others, respecting the decisions done in the beginning of this thesis' Literature Review study. After that, it was done a review about the positioning techniques and explaining why some of them were going to be detailed and others not, thus the chosen techniques were detailed and explained – in this case, specifically, the Lateration and Angulation techniques.

The sub-chapter right after is the main core of the whole Literature Review process, because it was where important ML techniques were explained. The choosing of those techniques was related with the amount of references found about each of them. Each sub-chapter was divided in two different parts: the first detailing how each the generic algorithm works and the second enumerating and comparing – according to the initially defined metrics - all the publications read throughout this Literature Review process. This sub-chapter starts by referring the ANNs algorithms and techniques, then going through the SVM technologies and variants and posteriorly explaining and enumerating the different Clustering approaches, with specific focus on the *k-Means* algorithm.

The purposes of this chapter was to help the decision of which approaches to follow and which are the techniques or algorithms that show the best performances in terms of error minimization. The next chapter of this document focuses strictly on the establishment of the methodology to be used in this thesis in order to solve the current constraints, using the information gathered and explained throughout this Literature Review chapter.

3 The Proposed Approach

In this chapter, the different methodologies used to approach this dissertation problem will be detailed and a comparison between the approaches studied in Chapter 2 will be done with the goal of choosing which ones to implement, test and apply in order to obtain a smaller average distance error than the existing techniques of mobile indoor positioning. The chapter is divided into five sub-chapters, starting by making an introduction to the proposed approaches, followed by an explanation of how the problem is formalized, how the implementation process will be done and how the results will be compared. Next, it will be detailed how the algorithms were implemented, immediately followed by an overview of how the whole Android application is structured. The chapter finishes by making a summary of what was established and explained previously, with the goal of grouping together the major details of the implementation process.

3.1 Introduction

As it was discussed throughout Chapter 2, there are several different ML algorithms that approach the indoor positioning theme with the goal of minimizing the distance error between the real position where the mobile device is located and the inferred position by it.

Hence, the total number of techniques designed, developed and tested is quite huge, which implies that choosing a specific one or a small group of approaches to test has to be a properly constrained process with the goal of developing and testing the most efficient and accurate ones. Finally it's parametrized the evaluation metrics and concluded which approach is actually the best, depending on each metric and applied to different application scenarios.

This chapter is where those metrics are specified, where those techniques are chosen, where it's explained how each of those techniques were developed – if changes were introduced or not and properly explained which were the reasons to do it – and detailed the whole application functioning process, going from the most intrinsic details of ML techniques to the

whole overview of how each interaction and information exchange in the Android application is done.

3.2 Methods and Materials

3.2.1 Problem Formalization

This dissertation's purpose is to minimize the distance error between the real position and the estimated position of a mobile device in indoor environments, as it was stated some times throughout this document. Due to the fact that error minimization is a vague statement, the first part of this section will detail what that means in the context of this study.

The magnitude of the error is dependent on each problem. For the small testing scenario, achieving MDE lower than 1.0m is considered good and achieving MDE above 1.0m is considered less good. For the big testing scenario, achieving MDE below 2.0m is considered good and achieving MDE above 2m is considered bad.

The goal of this thesis is not to achieve an optimal solution for the above stated intention. According to the Literature Review studied documents, it is still not possible to achieve a full accurate position in indoor environments in all the measurements, because of the high dependency the proposed techniques possess in relationship to the mobile device's sensors, signal loss and the application environments' configurations. Still, some of the experiments done, have achieved better accuracy than others using different computational techniques. So, the goal of this thesis is to study and test the approaches to be chosen and to conclude about multiple factors related with the techniques themselves and the scenarios configurations.

In order to do this, it will be reminded the positioning technology that is being used in this project, the implementation of the different algorithms and experiments done. It will also be taken in account the total remaining time to complete this thesis as a problem constraint, due to the fact that there is a short limit that implementation, tests and experiments have to be restrained to.

Making a connection between linear programming mathematics and the stated problem, the referred goal and the explained constraints, it was decided to summarize the whole problem as a linear programming approach. Henceforth, let:

- t_i - time necessary (in months) to complete the task i;
- N - the total number of tasks;
- A_x - area of the small scenario (in square meters);
- A_y - area of the big scenario (in square meters);
- ε_{max}^{x} - maximum imposed MDE for the small scenario;
- ε_{max}^{y} - maximum imposed MDE for the big scenario;
- $\varepsilon_{obtained}^{x}$ - error obtained using the best ML technique or algorithm for the small scenario;

- $\varepsilon^{y}_{obtained}$ - error obtained using the best ML technique or algorithm for the big scenario;

For the obtained solution to be optimal in the context of this thesis, the the objective function is:

$min\ Z = \varepsilon^{x}_{obtained} + \varepsilon^{y}_{obtained}$, subject to:

- R1: $\varepsilon^{x}_{obtained} \leq \varepsilon^{x}_{max}$;
- R2: $\varepsilon^{y}_{obtained} \leq \varepsilon^{y}_{max}$
- R3: $A_x \geq 15$;
- R4: $A_y \geq 200$;
- R5: $\sum_{i=1}^{N} (t_i) \leq 4.5$;

where $t_i, N, A_x, A_y \varepsilon^{x}_{max}, \varepsilon^{y}_{max}, \varepsilon^{x}_{obtained}, \varepsilon^{y}_{obtained} \geq 0$.

3.2.2 Implementation Process

The implementation process will be define as a step-by-step approach in order to achieve the final goal of this thesis. The steps defined to do it are:

1. Gather data in the selected testing environment, store it in a database and use them as training entries for the algorithms to learn.
2. Select the three most promising technique to implement according to Table 3.1 – for Wi-Fi positioning.
3. Implement the 3 most promising techniques.
4. Experiment the implemented approaches in 2 scenarios, one with more than 15 m² and other with more than 200 square meters of area, gather results and compare them.

Table 3.1: Classification of approaches using the Wi-Fi technology.

Reference	ML Technique	Positioning	Scenario	Average Distance

		Technique		Error vs Maximum Defined Error
[TBPA11]	ANN	√	~	~
[LL05]	ANN, BN, kNN	√	x	~
[MTT10]	ANN, PM	√	~	√
[MCB12]	ANN	√	x	~
[BBV05]	SVM	~	√	x
[PKY06]	SVM + Gaussian kernel	√	√	~
[SCQL08]	Dimensionality Reduction and SVM	x	√	x
[FAWJC12]	A-priori information + SVM	√	√	√
[YASN02]	Joint-Clustering	~	√	~
[JBPA06]	Clustering-based	~	√	~
[MLTL08]	kNN + k-Means	√	√	√
[MME10]	k-Means	~	√	√
[AS11]	kNN + k-Means	√	√	x
[WSE12]	Clustering-based	√	√	x

Table 3.2: Classification of approaches using the Bluetooth technologies.

Reference	ML Technique	Scenario	Average Distance Error vs Maximum Defined Error
[FNR11]	ANN	~	~
[ABFB10]	ANN	~	~
[LLCL12]	2 ANNs	√	x
[TN08]	LSVM	√	~
[PPCTL12]	SVM	√	~

As it was explained before, although the Literature Review chapter allowed for the study the combination of indoor positioning ML techniques and the Bluetooth technology, that

technology was not implemented throughout the development phase, mainly due to time constraints and changes in the initial dissertation requirements. Hence, Table 3.2 serves as a study of those studied algorithms using the Bluetooth technology eventually oriented for future improvements that include the implementation of these technologies.

3.2.3 Results evaluation

From the moment it was understood how the implemented algorithms behave in the current context and which were the approaches each one was using to minimize the Distance Error, it was specified which were the hypothesis that needed to be tested. Those hypothesis are mainly related with the behavior of the implemented algorithms, the characteristics of the testing scenarios and the configuration and amount of the scenarios' positions used as training set for the algorithms. The hypothesis that the tests intend to give answers to are:

1. **Hypothesis 1:** All of the implemented ML algorithms obtain a lower Mean Distance Error than the approach that does not use any Artificial Intelligence, in the same measured positions and in positions that were not used for training of the implemented approaches.
2. **Hypothesis 2:** One of the implemented ML algorithms obtains a lower Mean Distance Error than the other implemented ML algorithms.
3. **Hypothesis 3**: One of the implemented ML algorithms learns at a faster rate as the number of different measured positions for training grows.
4. **Hypothesis 4:** Measured Positions from which more samples were gathered have a lower Mean Distance Error than the ones from which less samples were gathered.
5. **Hypothesis 5:** Until a certain limit of Measured Positions, the number of Measured Positions used for training decreases the Mean Distance Error of each one of the ML algorithms.
6. **Hypothesis 6:** One of the implemented ML algorithms learns at a faster rate than the other implemented ML techniques, as the number of samples from the same Measured Positions grows .
7. **Hypothesis 7:** The Mean Distance Error in smaller-sized scenarios is lower than the one in bigger-sized scenarios.
8. **Hypothesis 8**: Using a kind of Training that per each classification entry added to the database uses that same entry as a valid Training Example for the next position inference has a lower Mean Distance Error than a Training with a static number of entries.

The set of tests planned was thought taking into account the goal of answering positively or negatively the hypothesis above. Some of them can't be tested by a single test or evaluated individually, since they are global enough in terms of the evaluation of all the

algorithms, the testing scenarios and the amount of samples necessary to obtain better results for each one of them.

3.3 Implemented Algorithms

3.3.1 Supervised Learning Algorithms

In the beginning of the development phase, some constraints related with the chosen algorithms were evaluated according to the following premises: first, the fact that the author never developed a ML algorithm before, although he had worked with some classification algorithms – specifically, ID3 and C4.5; second, the development of the current dissertation time constraints – namely, four months and a half, where part of this time was dedicated to testing, results evaluation and writing. These two facts contributed for making the choice of choosing a library for each one of the algorithms in order to ease the process and reduce the time spent developing each one of them, instead of implementing the algorithms from scratch.

From the moment this decision was final, the step that followed was to choose the library to be used according to a number of parameters that were important for the whole project. Those parameters are different for each of the algorithms, due to the fact that they all have totally different kinds of functioning. In the case of the Supervised Learning algorithms, namely the Artificial Neural Networks and the Support Vector Machine algorithms, there are common parameters like the Efficiency of the library, the Results Reliability and the Parameter Configuration possibility. Specifically for the SVM chosen approach, there is an extra parameter: Kernel Specification possibility.

To each one of these chosen parameters was attributed a classification number, from 1 to 5, where 1 is the worst case and 5 is the best case. It was also specified that, in the case of the choice for the ANN, the parameters Efficiency, Results Reliability and Parameter Configuration have 20%, 50% and 30% of weights – chosen according to the developer's requirements - to make a final decision of which one to use, respectively; in the case of the choice for the SVM library, the parameters Kernel Specification, Efficiency, Results Reliability and Parameter Configuration have 30%, 10%, 40%, 20% as weights for the final decision, respectively. Therefore, the following tables were built in order to make a final decision, either for the ANN library and the SVM library:

Table 3.3: Comparison between libraries that implement ANN.

Library Name	**Reference**	**Efficiency (1-5)**	**Results Reliability (1-5)**	**Parameter Configuration**

Encog v2.4	[Hea10]	5	5	5
Neuroph v2.4	[JTR10]	2	4	5
JOONE v2 RC1	[Hea02]	3	4	5

Table 3.4: Comparison between libraries that implement SVM.

Library Name	**Reference**	**Kernel Specification (Yes/No)**	**Efficiency (1-5)**	**Results Reliability (1-5)**	**Parameter Configuration (1-5)**
libSVM	[CL11]	Yes	4	4	5
JNI SVM Light	[Tho99]	Yes	5	3	5

The scores for each parameter were given according to the existent information in the References field and comparing to each one of the other libraries information.

As it can be seen in Table 3.4, Encog v2.4 has the best scores for each one of the evaluated parameters for the implementation of the ANN algorithm and according to Table 3.5, libSVM ends up being the right choice, mainly due to the fact that the Results Reliability parameter weights more than the Efficiency parameter. Hence, the chosen libraries for the ANN and SVM algorithm were Encog v2.4 and libSVM, respectively.

1 Artificial Neural Networks

The Artificial Neural Network algorithm has 6 input nodes and 2 output nodes. The 6 input nodes correspond to the features necessary to train the network and 2 output nodes are the x and y values of the position inferred by the network after training. The features of the current approach were defined as: the distance from the mobile device to the Access Point 1, the distance from the mobile device to the Access Point 2, the distance from the mobile device to the Access Point 3, the distance from the mobile device to the Access Point 4, the x inferred value and the y inferred value. Each one of the distances to each of the Access Points – in the our case, they are four – are calculated using a formula developed by [C12] and explained in Section 2.2.1. The inferred values, the x and the y, are calculated using the Lateration technique, also developed by [C12]. The 2 output nodes are the x and y values, obtained after training the network with the entry data available.

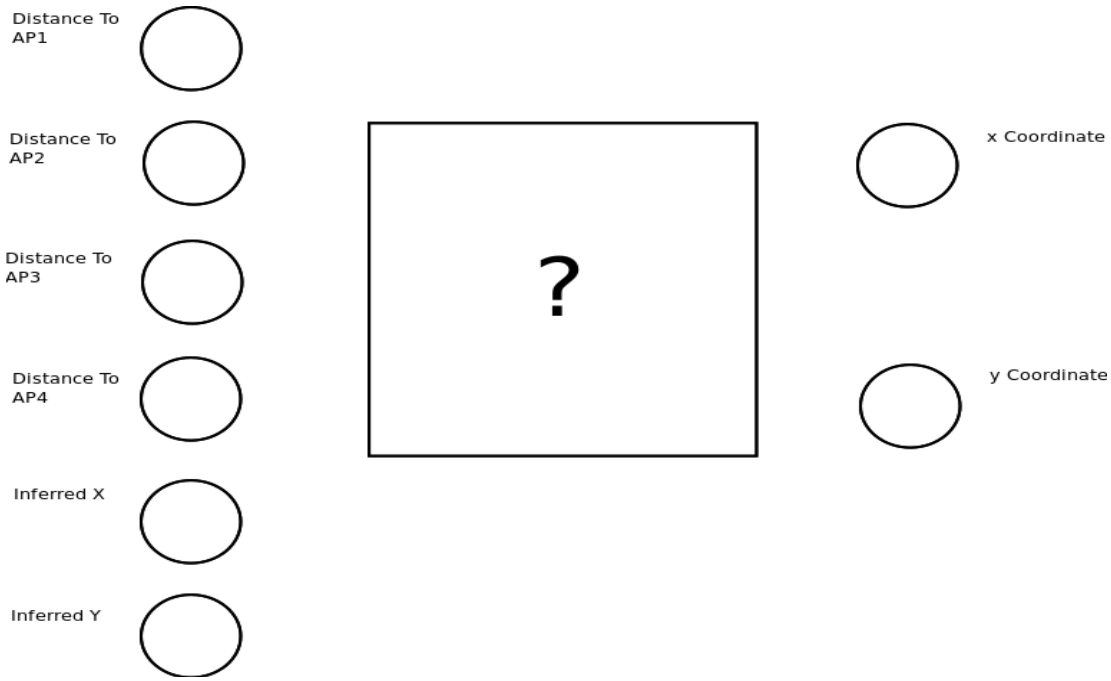

Figure 3.1: Artificial Neural Network without Hidden Layers or connections.

As it can be seen in Figure 3.1, at this point of explanation, the ANN does not have any Hidden Layers or weighted connections between nodes either inter-layers. To determine how many Hidden Layers and how many nodes those layers need to achieve the best solutions possible in the current problem using an ANN, there is a need to perform a search in order to accomplish the best configuration possible,

With that goal in mind, it was developed a function that calculates the accuracy of different Artificial Neural Networks, using different numbers of Hidden Layers and different numbers of Hidden Nodes inside them. All nodes possess inter-layers connections to each one of the nodes of the layers ahead – including the from Input Layer to the first Hidden Layer, from each Hidden Layer to the next, and from the last Hidden Layer to the Output Layer – and the Encog v2.4 API was configured to use as Activation Function the Sigmoid function 3.1, and Resilient Propagation as learning algorithm [RB93]:

$$f(x)=\frac{1}{1+e^{-x}} \tag{3.1}$$

Each created ANN was trained while the Mean Square Error was higher than 0.1% and the number of iterations was smaller or equal than 1500 iterations. These criteria were chosen taking into account that the computational time to train each network would increase considerably above the 1500 iterations of training if the goal was to achieve a Mean Square Error smaller than 0.1 %. The Mean Square Error, in Encog v2.4, is calculated by

$$MSE=\frac{1}{n}\sum_{i=1}^{n}(y_i-y'_i)^2 \tag{3.2}$$

where y_i is the ideal value and y'_i is the actual value. After the training process using the current configuration is finished, the function will go back to the first Training Entry and it will use its data as input for classification in the current weighted ANN. It will keep doing it, for each Training Entry in the data set, until the last one of them. The last step of the function is the calculation of the Accuracy of the current ANN configuration and the saving of that data in a data structured that is printed to a file. The Accuracy function is calculated with

$$Acc(x) = \frac{correctGuesses}{totalGuesses} * 100 \tag{3.3}$$

where the correctGuesses is the number of entries that were correctly classified and the totalGuesses variable is the number of entries in the data set. The output generated by the function through the whole process is available in Appendix A: Calculation of Number of Hidden Layers and Hidden Nodes in each layer for ANN.

As it can be seen there, the maximum Accuracy value obtained was of 93.548387 % -with 7 Hidden Layers and 10 Nodes in each HL -, although the computational time to train such a network was of 7.266924 seconds in a computer with more processing power than the standard mobile devices in the market nowadays. In a mobile device, the processing time would be much higher which brought the necessity of choosing the second best one in terms of Accuracy. The second highest Accuracy value was of 87.09677 %, achieved by several different configurations, so the final choice became dependent of the computational time as a tiebreaker. The configuration that had the lowest Computational Time for training an ANN is the one with 3 Hidden Layers and 9 Nodes in each HL, so the choice fell on that one.

Therefore, the final structural configuration of the implemented ANN had 6 Input Nodes in the Input Layer, 3 Hidden Layers with 9 Nodes in each HL and 2 Output Nodes in the Output Layer, as it can be seen in Figure 3.2:

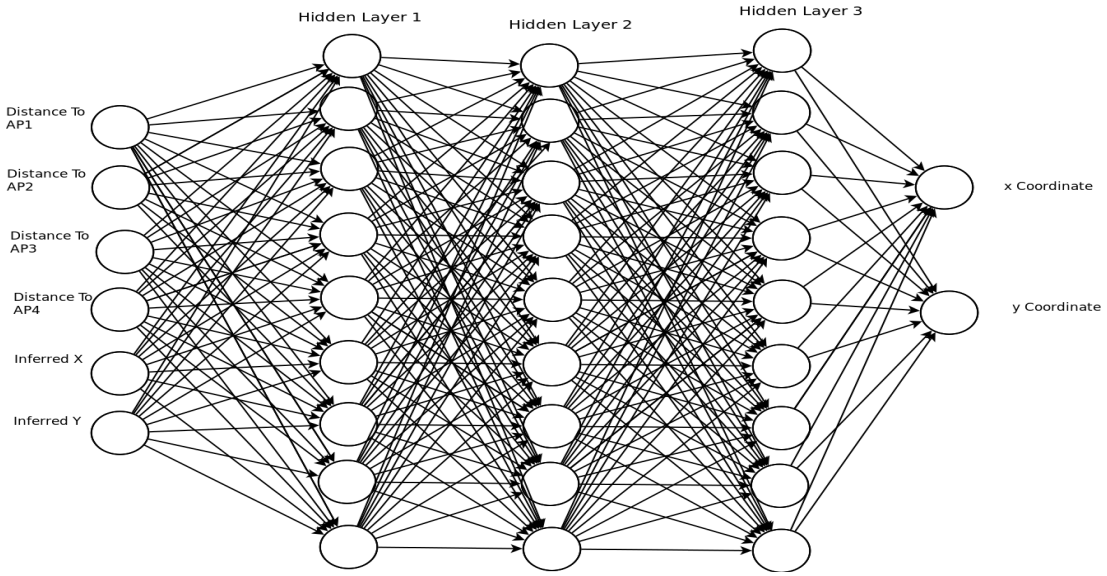

Figure 3.2: Final structure of the implemented ANN.s

The classification process using an ANN in the Android application, has several steps being the training just one of them. It starts by loading the data available in the database to data structures, then it normalizes the data by multiplying each one of the features by 0.1, in order that the ANN does not need to do difficult memory and time-consuming calculations. It's also recommended by the documentation of the Encog v2.4 API to normalize all the input data before they are fed in the Input Layer and denormalize it after it comes out of the Output Layer.

Next, the ANN is trained using exactly the same training parameters as the ones when searching for the best ANN structure – it trains while the MSE is higher than 0.1% and while the number of iterations done is lower than 1500. After the training process is over, the new entry data is normalized and fed into the trained network, producing results in the Output Layer. Those results are denormalized and passed for the last step of the process, the one that it was decided that it should be called Approximation.

This process consists of approximating the values output in the Output Layer to Real Position values of the closest entry in the data set to the input values of this new entry that have been fed to the Input Layer. Going into detail, it's calculated the Euclidean Distance between the new entry input distance features and the distance features from each one of the entries in the data set, being assigned as closest entry the entry that possesses the minimum Euclidean Distance to this new entry. Let TE^i represent the i^{th} Training Entry in the data set and NE the New Entry; let $TE^i_{d1}, TE^i_{d2}, TE^i_{d3}, TE^i_{d4}, TE^i_{infX}, TE^i_{infY}$ represent the TE distance to AP1, distance to AP2, distance to AP3, distance to AP4, inferred x and inferred y features, respectively; $NE_{d1}, NE_{d2}, NE_{d3}, NE_{d4}, NE_{infX}, NE_{infY}$ represent the same features for the New Entry. The Euclidean Distance – d -, in this case is calculated by the expression:

$$d(TE^i, NE) = \sqrt{(TE^i_{d1} - NE_{d1})^2 + (TE^i_{d2} - NE_{d1})^2 + (TE^i_{d3} - NE_{d3})^2 + (TE^i_{d4} - NE_{d4})^2} \quad (3.4)$$

The expression 3.4 is calculated for all the entries in the data set and the one that has the minimum value is assigned as the closest TE to the current NE. Let n be the number of Training Entries in the data set. Let z be the minimum distance value. Hence,

$$z = min(d(TE^i, NE), d(TE^{(i+1)}, NE), d(TE^{(i+2)}, NE), ..., d(TE^n, NE)) \quad (3.5)$$

and let m be the index of the obtained z value in the data set.

After the minimum value and its respective index are found, it's calculated the error between this entry's Real Position and the Inferred Position:

$$err(TE^m_{RP}, TE^m_{IP}) = \sqrt{(|TE^m_{RPx} - TE^m_{InfX}| + |TE^m_{RPy} - TE^m_{InfY}|)} \quad (3.6)$$

If this error is lower or equal than a predefined value of 0.5 – the value was chosen with the goal of having close-to-optimal solutions with less than 0.5 meters of Distance Error – the final value that will be attributed to x and y would be the values of TE^m_{RPx} and TE^m_{RPy}, respectively, due to the fact that the distances to the APs calculated by the Lateration algorithm were pretty similar with each others, which may mean it's the same position or a very similar one. If the error is higher than the predefined value of 0.5, then the following conditions are verified:

$$C1: TE^m_{InfX} < TE^m_{RPx}, \|TE^m_{InfX} = TE^m_{InfX} + |TE^m_{RPx} - TE^m_{InfX}\| \quad (3.7)$$

$$C2: TE^m_{InfX} > TE^m_{RPx}, \|TE^m_{InfX} = TE^m_{InfX} - |TE^m_{RPx} - TE^m_{InfX}\| \quad (3.8)$$

$$C3: TE^m_{InfX} = TE^m_{RPx}, TE^m_{InfX} = TE^m_{InfX} \quad (3.9)$$

$$C4: TE^m_{InfY} < TE^m_{RPy}, \|TE^m_{InfY} = TE^m_{InfY} + |TE^m_{RPY} - TE^m_{InfY}\| \quad (3.10)$$

$$C5: TE^m_{InfY} > TE^m_{RPy}, \|TE^m_{InfY} = TE^m_{InfY} - |TE^m_{RPy} - TE^m_{InfY}\| \quad (3.11)$$

$$C6: TE^m_{InfY} = TE^m_{RPy}, TE^m_{InfY} = TE^m_{InfY} \quad (3.12)$$

The conditions C1, C2 and C3 are related with the final value of x. C1 verifies if the inferred x of the TE m is lower than the real x of the same TE. If this condition is confirmed, it adds the module of the difference between real x and the inferred x of the *mth* TE to the current inferred value of x; if not, it goes to C2. C2 verifies if the inferred x of the TE m is higher than the real x of the same TE. If this is true, it subtracts the module of the difference between x and the inferred x of the *mth* TE from the current inferred value of x; if not, it goes to C3 that keeps the same value for the inferred x. The conditions C3, C4 and C5 verify exactly the same conditions as C1, C2 and C3, respectively, but now using y and always operating on the final value of the inferred y.

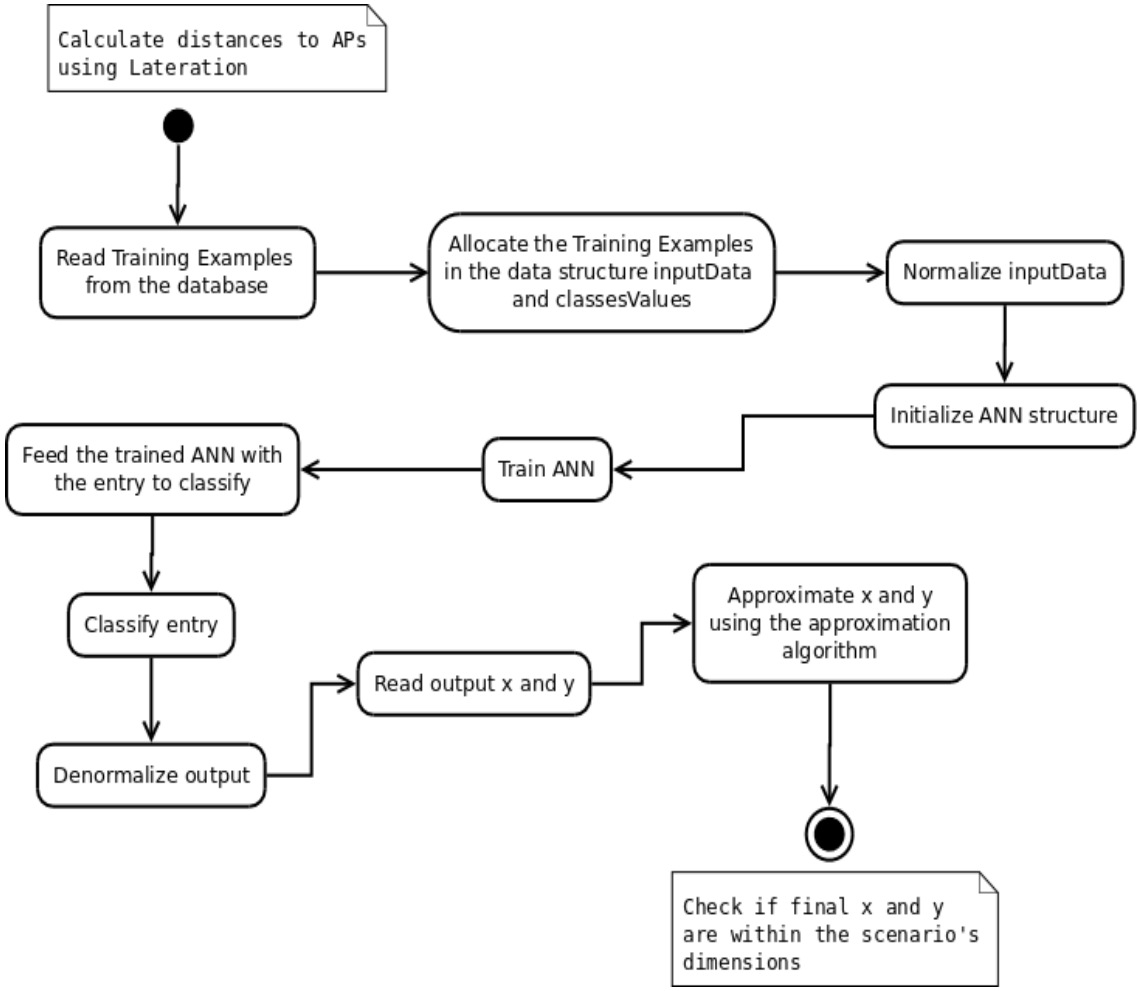

Figure 3.3: The states of the ANN whole classification process.

When these operations are over, it's obtained the final values for the current inference of x and y, therefore we got our final position and the algorithm is over when this state is achieved, as it can be seen in Figure 3.3.

2 Support Vector Machines

Despite the fact that the choice of a library to implement the Support Vector Machines algorithm fell on the usage of libSVM, as it was explained in Section 3.3, there are some restraints associated with it.

First of all, although through libSVM it is possible to implement multi-label classification solutions for such kind of problems, the library just allows solutions with one numerical output. In the case of the indoor positioning inference using mobile devices, there's the need of a data structure to output the position or of two numerical outputs that can represent either x and y. Because of this fact, it was necessary to make a decision about how this constraint should be

approached and it was decided to basically repeat two times the processing of the SVM algorithm: the first time for x and the second time for y, using exactly the same features but different labels, these last ones corresponding to the values of x and y, respectively in each of the times.

The second restraint is related with class labeling. Because SVM is a classification algorithm, the solutions that the algorithm produces are always the exact same known labels that were used for training. This means that the SVM algorithm that libSVM structures and implements will never produce any new positions other than the ones known in the training process, which is not minimally useful in this problem's case. In order to solve this, it was developed an approximation algorithm – the same developed for approximating solutions in the ANN approach – in order to enable the possibility of the generation of new positions.

The libSVM has some spescific objects than can be configured and manipulated in order to treat and transform the whole data needed for the algorithm to work properly. Those set of objects include:

1. the object *nodes*, that is a container of Training Entries – including the classes' labels and each one of the features defined for the problem;
2. the object *svm_problem* that has three attributes (the length of the dataset, the array of Training Entries and the expected classes labels);
3. the object *svm_parameter* that is used to configure the data related with the *kernel* of the SVM and the variables related with it;
4. the object *svm_model* that uses the object *svm_problem* and the data from the *svm_parameter* object to build a model from which the SVM algorithm will train with the Training Entries and test using the testing data defined.

Due to this problem's context and to the data that the previously developed Android application by [C12] is able to receive and transform, the features chosen to approach the problem using the SVM algorithm were:

1. the calculated distance to AP1 from the Lateration algorithm;
2. the calculated distance to AP2 from the Lateration algorithm;
3. the calculated distance to AP3 from the Lateration algorithm;
4. the calculated distance to AP4 from the Lateration algorithm;
5. inferred x or y, because of the fact that it was decided to create two SVM iterations due to the restraints of the libSVM library, the first for the calculation of x and the second for the inference of y, as it was stated before.

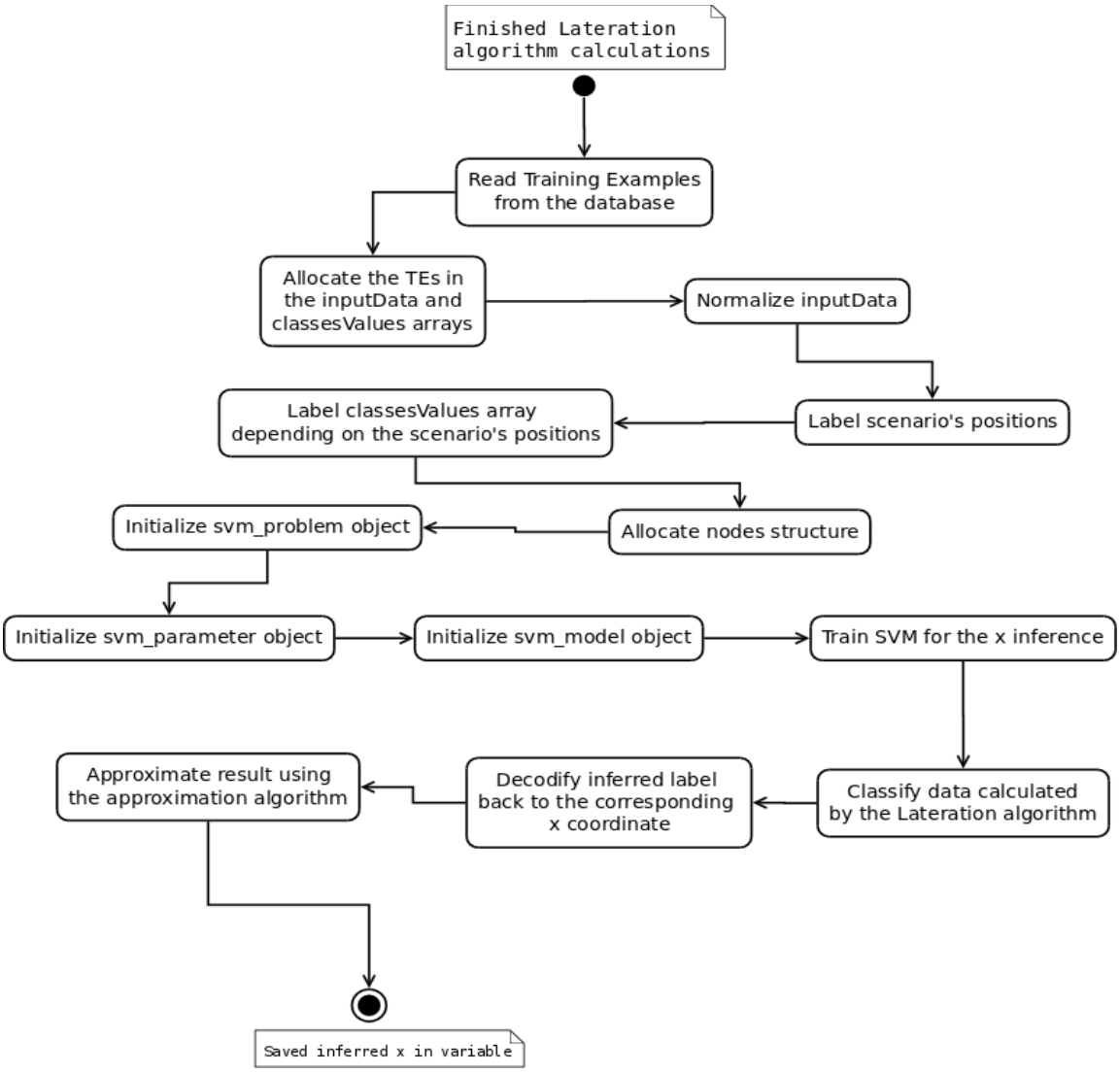

Figure 3.4: The states of the SVM algorithm classification for the x coordinate.

The content of these features is stored in the software database and are read into a two-dimensional array called *dataMatrix*. There is a one-dimensional array in the application, called *classesValues*, that contains all the information from Real Positions of each one of the Training Entries stored in the database. Through the SVM algorithm, this *classesValues* array is filled twice, once with the corresponding label of the x value and the second time with the corresponding label of the y value.

Before these two arrays are assigned in the *svm_problem* object along with the size of the data set to train the SVM, the *dataMatrix* suffers an operation of data scaling. This consists in the transformation of every single value of each feature in a new value that ranges from 0 to 1 and the main goal of this operation is to keep the sparsity of the data so that the chosen kernel works properly. Right after this process is finished, the process of labeling starts. Labeling, in this case, consists in converting the Real Position values allocated in the *classesValues* array

into numbered integers corresponding to each one of the positions of the map. This is a necessary process because libSVM only accepts classes values as being labels represented by an integer and, in this case, each integer represents a position far away 0.1m from the previous position. Hence, the bigger the scenario, the more labels will be calculated for it. The next step is to allocate all the *inputData* and *classesValues* information into the *nodes* structure and then to the *svm_problem* object. Posteriorly, the problem is initialized by filling in the *svm_parameter* object attributes, that are mainly related with the kernel specifications (such as *kernel_type*, *C* and *gamma*) and with the definition if there is a will of attributing probabilities to each of the known classes values during the training process. In our case, more than necessary, that's helpful since it's a multi-label classification problem (and therefore uses a one-vs-all approach for each one of the known labels). The probability generation for each one of the classes may be of interest for future improvements in the SVM algorithm too, if that's the case at some point.

The main choices that need to be done when using a SVM algorithm for a practical application is the choosing of the kernel and the C attributes to use. The choice of the kernel was obvious, in this case, since we want to compare the results the algorithm will obtain with the ones from the best approach in the literature ([FAWJC12]), furthermore the same kernel type was used – Gaussian kernel. When the choice is a Gaussian kernel, there's the need of choosing γ^2 . The “squared gamma” parameter has strong influence in the way the algorithm classifies entries, making strong variations in the bias and variance of a whole distribution. It was chosen to have a low gamma parameter, of 0.5 and then $\gamma^2 = 0.25$, so the bias is small also (which means the algorithm will produce different positions as output), attempting to minimize the average distance error. Due to the fact that when gamma is low, the bias is low and the variance is high and because it's necessary to balance the bias/variance relationship from the beginning, it was chosen a low C parameter (valued 2), in order to attempt to decrease variance and increase the bias levels all over the distribution of classified entries. Lastly, the probability parameter was set to 1 because like that it would generate probabilities for all the known labels to be the correct one, due to the reasons stated before. Also, it helped to control the development of the algorithm and check how far the classification of a certain entry was from the correct label at each one of the tests done during development.

The next step is to create the *svm_model* object. It receives as attributes the *svm_param* object, the array of labels and the number of class labels. Then it trains the SVM using those parameters.. Supposedly, after training, the SVM should do the cross-validation step, that consists in testing part of the training set and check if the accuracy of the algorithm is as good as what the programmer defines as good. If it's not, the parameters are changed and it's tested again, repeatedly, until it achieves the goal the programmer defined. This is an important part of the SVM algorithm and libSVM allows an easy implementation of it, but it's a time-consuming process and that's the main reason why it was decided that our approach of the SVM algorithm would not do it. Instead, the parameters were chosen for the reasons explained previously.

When the model is created and the SVM is trained, the algorithm evaluates the new entry and classifies it. During this process, the algorithm decodifies the label output to the

position it corresponds, so it can actually be shown on the map of the mobile device where the user is located. Then it performs exactly the same approximation algorithm over that decodified value as the one that ANNs use, attempting either to minimize the error between the real position where the mobile device is located and the inferred position by the SVM and adding a new entry to the data set that will be used for Training posteriorly, which means that new known labels will be added too, since the approximation algorithm will output a new position.

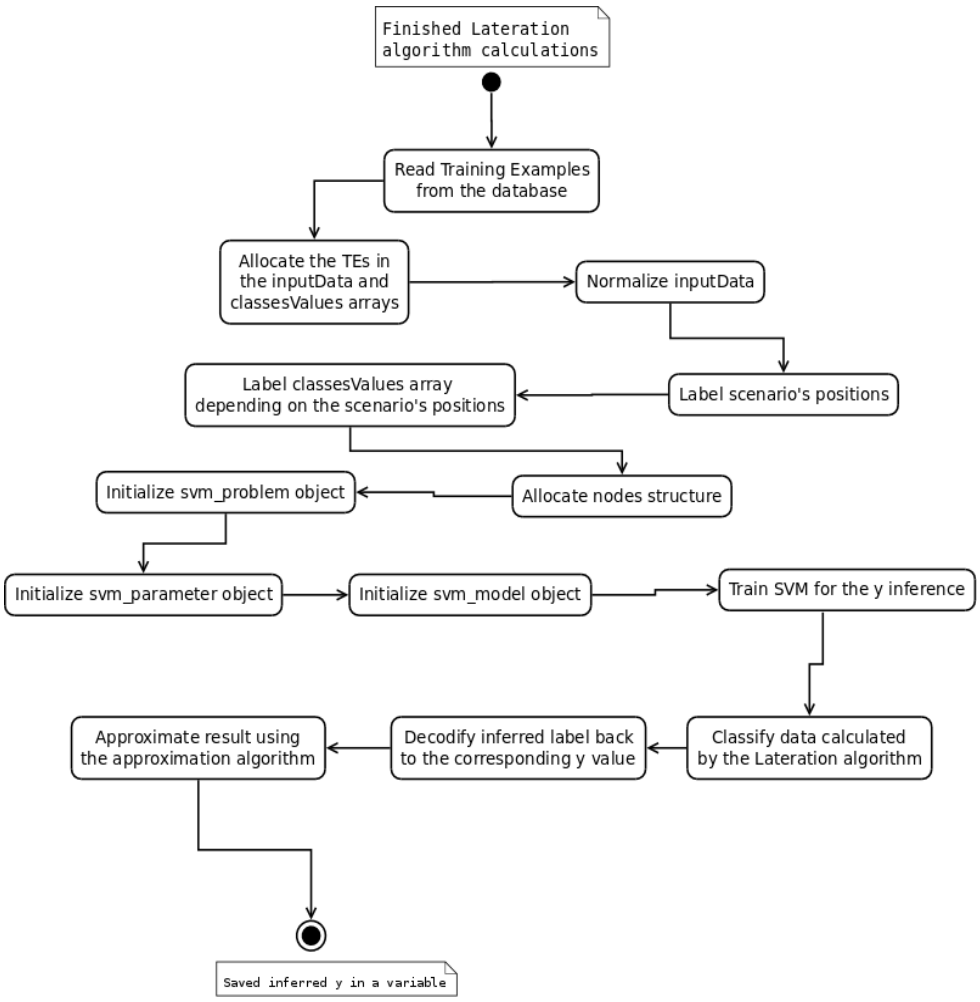

Figure 3.5: The states of the SVM algorithm for the y coordinate.

This whole algorithm is repeated twice, as it was said before, once for x the other time for y.

3.3.2 Unsupervised Learning Algorithms

Unsupervised Learning algorithms differ, conceptually, from Supervised Learning algorithms because they use unlabeled training sets instead of labeled ones. Applied to the problem being solved by this Dissertation, this means that the Training Entries available in

Unsupervised Learning algorithms do not have any Real Position classes values, but they be used as Training features instead.

Similarly with what happened when planning the Supervised Learning algorithms, in the beginning of the development phase it was decided that it would be preferable that a library would be used for the implementation of the Clustering approach, due to the same reasons presented in Section 3.3.1, rather than implementing everything from scratch. Due to the fact that there was some experience already with the functioning of Encog v2.4, since this library offers support for Clustering algorithms' development and since all the other libraries found possess the same exact possibilities as Encog v2.4, it was decided that the implementation of the K-Means Clustering approach would be done using it.

Henceforth, of the three selected and implemented algorithms, the K-Means Clustering was the one that took less time being developed, mainly because there was some previous experience with the Encog v2.4 library.

1 Clustering

The main goal of any approach using the k-Means algorithm is to group data into coherent subsets. One of the first decisions to be done when initializing a k-Means Clustering algorithm is how many of those subsets are necessary to group data the most accurate way possible. There are several different methods that solve this constraint, from which the Elbow Method is one of the most well-known, of the most straightforward in terms of implementation and one of the less time-consuming techniques. But in order to compute how many clusters this application of the algorithm would need, it was necessary to know, at each point of the algorithm, how many Training Entries are available for the Clustering process. Due to this fact, implementing an Elbow Method technique to know how many clusters the software needed at each time a positioning inference required too many calculations. Instead, it was decided to implement a technique called Rule of Thumb, that basically computes the number of entries divided by 2 and calculates its square root, converting the result to an integer, as it can be seen in Equation 3.13.

$$k \approx \sqrt{\left(\frac{n_{entries}}{2}\right)} \tag{3.13}$$

The variable $n_{entries}$ represents the number of entries to be grouped by the algorithm and k is the number of clusters calculated by the formula. Using this simple operation, it's possible to calculate, at every position inference instance, how many clusters are necessary.

Quite similarly to the algorithms explained in Section 3.3.1, it was decided that the features of the k-Means algorithm should be:

1. The distance value from the mobile device to the AP1;
2. The distance value from the mobile device to the AP2;
3. The distance value from the mobile device to the AP3;

4. The distance value from the mobile device to the AP4;
5. The inferred value for coordinate x;
6. The inferred value for coordinate y;
7. The real value of coordinate x;
8. The real value of coordinate y.

Unlike what happened with the Supervised Learning algorithms list of features, that did not include the real values as features, but as of classes values or labels for the presented algorithms, in the k-Means Clustering approach, since it's an Unsupervised Learning algorithm, the values of the real coordinates are used as features for grouping data. The remaining features are exactly the same used either in the ANN and the SVM algorithms and are all calculated by the software developed by [C12].

As you can see in Figure 3.6, the algorithm starts by reading each one of the entries to group from the system's database and allocates them in a two-dimensional array called

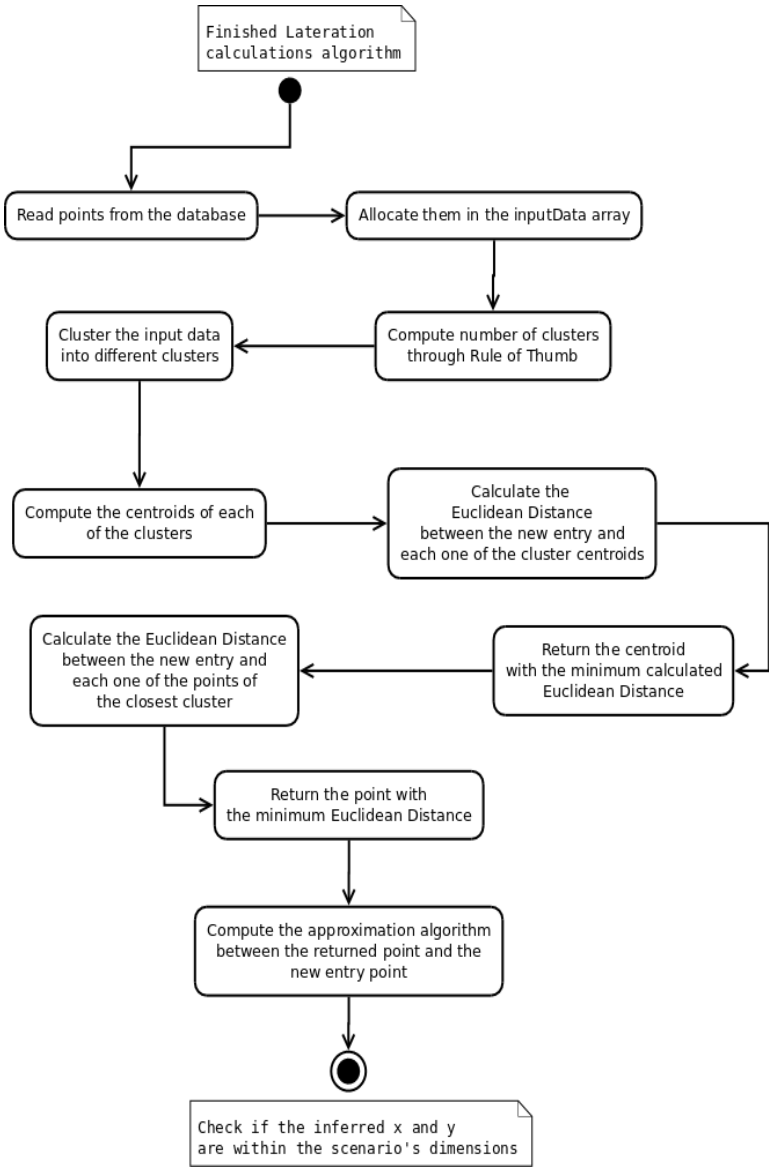

Figure 3.6: The states of the Clustering whole grouping and classification process.

inputData. Next, it computes the number of clusters necessary using the Rule of Thumb, according to the number of input entries (Equation 3.13). Right after, it executes the Clustering process, dividing the input data in different clusters depending on the similarities between the feature values of each one of them (Chapter 2.3.3). Posteriorly, the entry to classify is passed to the algorithm. The entry data has all the features from the grouped entries, unless the real positions' ones, since they are not known at this point of execution.

The next step is to compute which cluster centroid is closer to the entry data, by calculating the Euclidean Distance between the features related with distances to the APs of those two points. Let $ED_{(i,c)}$ represent the Euclidean Distance value from the entry to be classified i to the centroid c and let $d1_i$, $d2_i$, $d3_i$ and $d4_i$ be the distances from the entry to be classified to AP1, AP2, AP3 and AP4, respectively and $d1_c$, $d2_c$, $d3_c$ and $d4_c$ be the distances from the centroid of the current cluster to AP1, AP2, AP3 and AP4, also respectively. Hence, the Euclidean Distance from the entry i to the centroid c is calculated by the equation 3.14.

$$ED_{(i,c)}=\sqrt{((d1_i-d1_c)^2+(d2_i-d2_c)^2+(d3_i-d3_c)^2+(d4_i-d4_c)^2)} \tag{3.14}$$

If the $ED_{(i,c)}$, where $c\in[1,..,k]$ and where k represents the number of clusters is lower than the minimum distance up to the current iteration of the current operation, then:

$$min(ED)_{(i,c)}=ED_{(i,c)} \tag{3.15}$$

If not, the same calculations are done with the next cluster centroid, until all centroids have been checked. It is returned the centroid with the minimum distance. Let that centroid be represented by $min_{(i,c)}$.

Then, it's computed which is the closest point from the points that are part of the closest cluster. Let CP represent the closest point and the set S be the set that contains all the points of the cluster where the centroid is $min_{(i,c)}$ and let p be the number of elements in the set S . This calculation operates cyclically, from the first point assigned in the set S until all points are iterated. In each iteration, the operations applied are:

$$min((ED_{(i,min_{(i,c)})})_1,(ED_{(i,min_{(i,c)})})_2,...,(ED_{(i,min_{(i,c)})})_p),S\in[1,...,p] \tag{3.16}$$

which represents the calculation of the minimum of the distances between the entry to be classified i to each one of the points belonging to the cluster S where the centroid is $min_{(i,c)}$.

The element from the cluster S which possesses the minimum distance to the entry point i will be the one used in the approximation phase, during the next step, in order to serve as comparison for the new entry, since it is the closest point from the closest cluster centroid,

which means that their features are the most similar between all points of the same cluster, so most probably they should have a numerically close real position.

Finally, the approximation algorithm is done, exactly like the one implemented either by the ANN and the SVM approaches, where it's calculated an error C_{error} - meaning Current Error - between the real position of the closest point CP and the position inferred by the Lateration algorithm for the entry data i and it's defined a maximum error allowed – that in this case, was set to 0.5m – because it was defined that a very good positioning inference was one which had an error lower or equal to that value. Let the inferred position be represented by i, the maximum error constant by M_{error} and the final inferred positions by $FPos_x$ and $FPos_y$, being $FPos$ correspondent to the point with coordinates $(FPos_x, FPos_y)$. Let also CP_I^x and CP_I^y represent the coordinates x and y of the inferred position of CP, CP_R^x and CP_R^y represent the coordinates x and y of the real position of CP and i_x and i_y represent coordinates x and y of the inferred position of the entry data i:

$$A1: C_{error}^x = \left| CP_R^x - i_x \right| \tag{3.17}$$

$$C1: if\ C_{error}^x < M_{error} \tag{3.18}$$

$$C2: if\ i_x > CP_R^x \tag{3.19}$$

$$C3: if\ i_x < CP_R^x \tag{3.20}$$

$$A2: FPos_x = i_x \tag{3.21}$$

$$A3: FPos_x = \left| i_x - \left| CP_R^x - CP_I^x \right| \right| \tag{3.22}$$

$$A4: FPos_x = \left| i_x + \left| CP_R^x - CP_I^x \right| \right| \tag{3.23}$$

A1 is the representation of the expression that calculates the value of C_{error}^x, the current error for the x coordinate, which is given by the module of the subtraction of the closest point's real x coordinate and the entry data's inferred x coordinate. The condition C1, compares if C_{error}^x is lower than the maximum error constant M_{error} and if true assigns the value of the entry data's inferred x coordinate to the final position's x coordinate; the condition C2 compares if the value of the entry data's inferred x coordinate is higher than the real x coordinate of the closest point and if it is true, calculates the module of the subtraction between entry data's inferred x coordinate and the module of the subtraction between the real x of the closest point and the inferred x of the closest point; C3 verifies if the entry data's inferred x coordinate is lower than the closest point's real x coordinate and if it is true, assigns to the final position's x coordinate the module of the value of the addition between the entry data's inferred x coordinate and the the module of the subtraction between closest point's real x and closest point's inferred x; A2 assigns the value of entry data's inferred x coordinate to the final position's x coordinate. The way these conditions are linked, in order to produce the value of the final position of x, is:

```
//for the inference of the x coordinate
if C1 then A1
else {
    if C2 then A3
    else if C3 then A4
    else A2
}
```

After the inference of the x coordinate for the final position, it was still necessary to infer the y coordinate for the final position. It was used exactly the same conditions as in the x coordinate position inference, now changing only the coordinates to be compared, assigned or calculated from x to y. It also has the exact same structure of grouping the conditions together, in order to obtain the final position's y coordinate.

$$A3: C^{y}_{error} = \left| CP^{y}_{R} - i_{y} \right| \tag{3.24}$$

$$C5: if\ C^{y}_{error} < M_{error} \tag{3.25}$$

$$C6: if\ i_{y} > CP^{y}_{R} \tag{3.26}$$

$$C7: if\ i_{y} < CP^{y}_{R} \tag{3.27}$$

$$A4: FPos_{y} = i_{y} \tag{3.28}$$

$$A5: FPos_{y} = \left| i_{y} - \left| CP^{y}_{R} - CP^{y}_{I} \right| \right| \tag{3.29}$$

$$A6: FPos_{y} = \left| i_{y} + \left| CP^{y}_{R} - CP^{y}_{I} \right| \right| \tag{3.30}$$

```
//for the inference of the x coordinate
if C5 then A4, A3
else {
   if C6 then A5
    else if C7 then A6
    else A4
}
```

When this process is completed, the final inferred position of the mobile device is given by the point composed with the values of the inferred x and the inferred y. That's when the algorithm stops, shows the final position to the user of the mobile device and asks him to insert the real position in order to write the results in the database.

3.4 Application Overview

This section presents the changes introduced in the developed prototype of [C12]. It starts by listing the Requirements Specification, then it details the System Physical Architecture – that did not change much from what was specified by [C12] -, going to an explanation of the application functioning steps and further it describes the database structure and it finishes by showing screen shots of the Android application interface.

3.4.1 System Requirements' Specification

This section will summarize the requirements established for this research project, although the majority of them have been described and reported throughout the previous document parts. The list of requirements will be divided in 4 parts: the functional requirements, the non-functional requirements and the assumptions. The following list is a continuation of the list written in [C12], due to the fact that all the implementation and testing done was on top of the work developed in that document.

Functional Requirements:

- The Lateration results have to be organized in data structures, in order for the algorithms implemented and/or algorithms that may be implemented in a posterior iteration of the current project to be done without too many structural changes.
- The algorithm of ANN has to work according to the definition of an Artificial Neural Network, although changes may be done throughout the process with the goal of improving the final results.
- The algorithm of SVM has to work according to the definition of a Support Vector Machine, although changes may be done throughout the process with the goal of improving the final results.
- The algorithm of k-Means Clustering has to work according to the definition of a k-Means Clustering algorithm, although changes may be done throughout the process with the goal of improving the final results.
- The results of each one of the ML algorithms implemented, using a smaller or higher number of Training Examples, have to improve the results obtained without using any Artificial Intelligence.
- The application has to be able to ask the user to insert the real position and save it to the database.
- It should be fast to select the algorithm to use and the kind of test that needs to be done at each time.

Non-functional Requirements:

- The application has to run on Android devices.
- The implemented algorithms should be processed on the Android device.
- At least three ML algorithms should be implemented for comparison.
- The application may use Wi-Fi and Bluetooth signals as combination, or Wi-Fi or Bluetooth signals alone.

Assumptions:

- The results obtained are valid only for the used hardware.
- The Access Points are fixed in a position.
- The scenario information and the Access Points coordinates in the testing environment are stored in the database.

3.4.2 System Physical Architecture

The physical architecture of the system is briefly described in this section through a diagram that represents all the active components in the position inference process. As it can be seen in Figure 3.7, the architecture is divided in two parts, being them the Device and the External Hardware.

The External Hardware is composed by a minimum of 4 Access Points that send RSSI signals to the Device in order for it to infer the current position in a determined scenario. The Access Points used were all equal in order to avoid the introduction of errors due to different signal providers.

The Device has two main components: the Application and the Database. The Application communicates several times with the Database during the position inference process, since it's in the Database that all the data is saved and from there that it's loaded. The Database contains the environment information, received from the Wi-Fi Access Points and treated with calculations done in the Device. These transactions of environment information from the Database to the Device and of results in the opposite direction composes all the interactions done between the two components. The Application contains four main components: The Wi-Fi Signal Scan, the Position Estimation using the Lateration algorithm, the Position Estimation using an ML algorithm – ANN, SVM or k-Means Clustering and the visualization of the results.

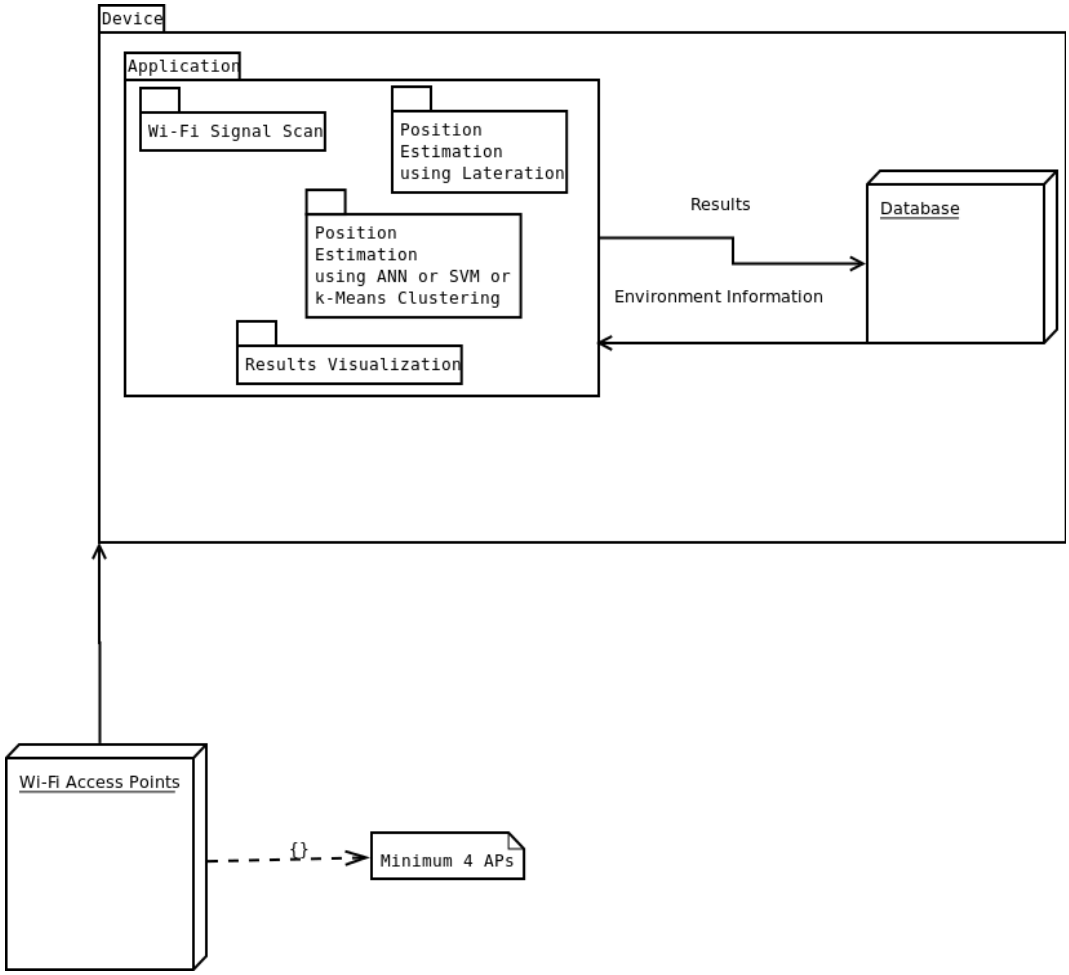

Figure 3.7: Deployment Architecture of the system.

3.4.3 Application Functioning Steps

1 Main Menu and Options

The Main Menu of the TouAReg Indoor application is composed with 5 different options that the user can choose:

1. Locate
2. Select Scenario
3. Options
4. About

5. Exit

Each one of these options has a button, where the user can click and select the action he/she wants to execute. Each one of these actions is a different Activity and as the options names explicit, the first one is to start the location process, the second is to select a different scenario where to apply the location process, the third is to choose some options related with the location and the algorithms to use to infer positioning, the fourth shows some information about the app and the fifth is to exit it.

The Options Menu contains only testing options until the end of this Dissertation. Those options include the possibility to select the algorithm to use – the possibilities are the Artificial Neural Network, Support Vector Machines and Clustering algorithms – and the possibility to choose which test it is to be done. The options related with the tests are:

1. Test with Static Training using first 30 Training Examples, then 100 TE and finally 150 TE. For each of these configurations, test first with no Artificial Intelligence, then using the ANN, followed by using the SVM and finally using the k-Means Clustering.
2. Test with Continuous Training using first 30 Training Examples, then 100 TE and finally 150 TE. For each of these configurations, test first with no Artificial Intelligence, then using the ANN, followed by using the SVM and finally using the k-Means Clustering.

These options may disappear once the application becomes commercial and may be replaced by options more related with the end user's own preferences. For the purpose of this study and all the tests done in this problem's context, these options were extremely useful due to the fact they allowed the author to select which test he wanted to do at each moment, it would save the information in different databases, organizing it better and in an easier way and it would automatically generate statistics about each one of the collections of data gathered and the algorithms' usage and performance, allowing the author to spare time organizing all the information stored and preventing the loss of time doing calculations that the software can do by itself.

2 Signal Scan

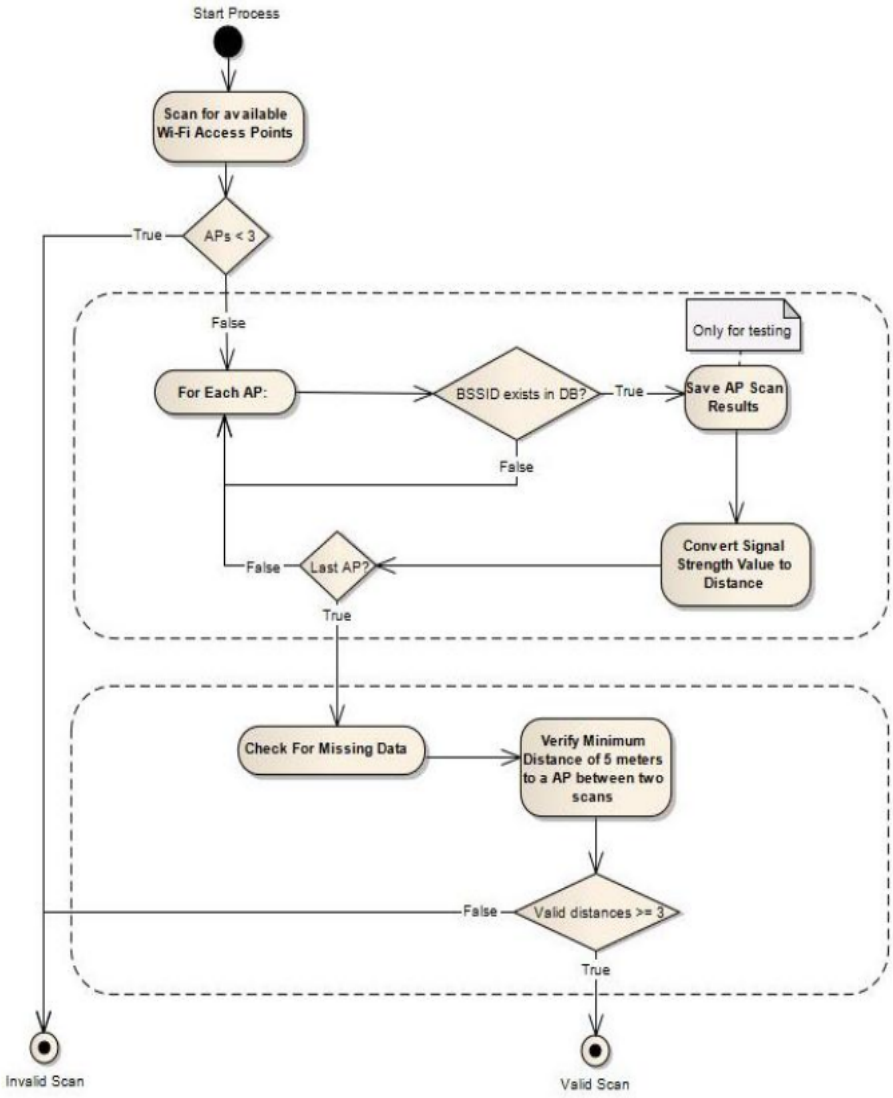

Figure 3.8: Wi-Fi scan and validation process. Source: [C12].

The step of Signal Scan did not suffer changes in its functioning. The option taken from the beginning was that it shouldn't be changed, since it's functioning properly and there will be

need of comparing the results obtained by the algorithms developed in the context of this study with the ones of [C12].

As it can be seen in Figure 3.8, the scanning process starts by verifying if the Access Points positions is available in the database for the specified scenario. If false, the software doesn't start the scan, considering it invalid; if true, it received the RSSI signal and saves it in the database, using it also for conversions between Signal Strength to Distance to the AP that sent it.

The software does this for all the Access Points available and posteriorly verifies if the Maximum Distance between 2 scans is of 5 meters to a AP, in order to stabilize the results that will be obtained after. This process is the calculation of Valid Distances and only if the number of Valid Distances is higher or equal to 3 it is considered a valid scan.

3 Learning Process

The Learning Process depends on the algorithm that may have been selected to infer position. Still, there is a similar work flow before and after the algorithm's usage, whichever the option that was chosen.

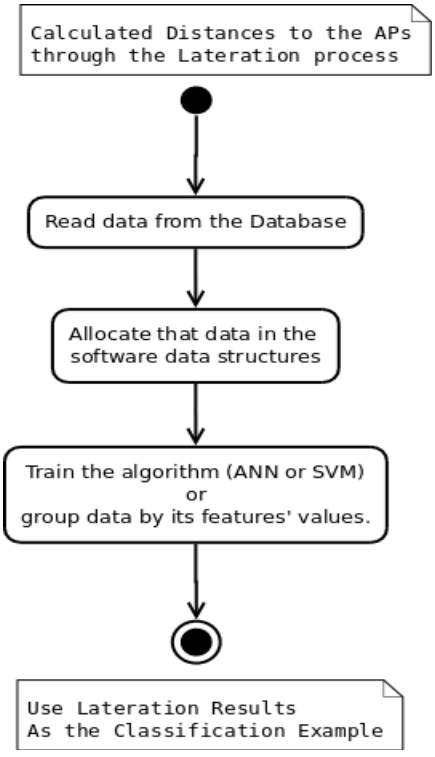

Figure 3.9: General view of the learning process.

It starts by reading the Training Examples from the database and allocates them in the data structures defined on each one of the ML algorithm's classes appropriately. Meanwhile, the training process – in the cases of the ANNs and the SVM algorithms – or the clustering process

starts and either trains the inherent structures to each one of the algorithms or groups the data into clusters according to the parameters configured for each one of them. The functioning of each one of the ML algorithms is very specific to each one of them and it was detailed already in the section 3.3.1 and 3.3.2.

Since the goal is to classify the data calculated by the Lateration algorithm and this process is not part of the Learning process anymore, Figure 3.9 as this moment of the general algorithm as end state.

4 Positioning Inference and Real Position Insertion

Figure 3.10 shows a diagram with the summarization of how all the positioning inference and real position insertion is done. The starting point of this process is when the Training or Clustering process is finished and the first step is to use the Lateration data for classification, meaning that the output of the Lateration process is allocated in a data structure that is fed in the ANN or SVM algorithm, or computed through the Clustering technique. After this is done, the output of this process is received and the approximation algorithm is started.

The approximation algorithm, as it was explained before, basically finds an existent closest point in the data set and makes some comparisons and operations between the new inferred point and this closest point found. The output of these operations goes through an optimization process, that basically checks if the final inferred positions are out of the scenario's dimensions. If x or y or both are, each one is changed for the scenario's dimension, making sure that the point can be shown on map.

The next step is the presentation of this final point on map, so the user can see where the mobile device inferred the position in relationship to the scenario where he/she is located. Then an input window appears and the user has to input his real position. When that is done, the real position and all the other information related with the inference of the position is stored in the database. Then or the user leaves the Locate Activity or the process of location starts again.

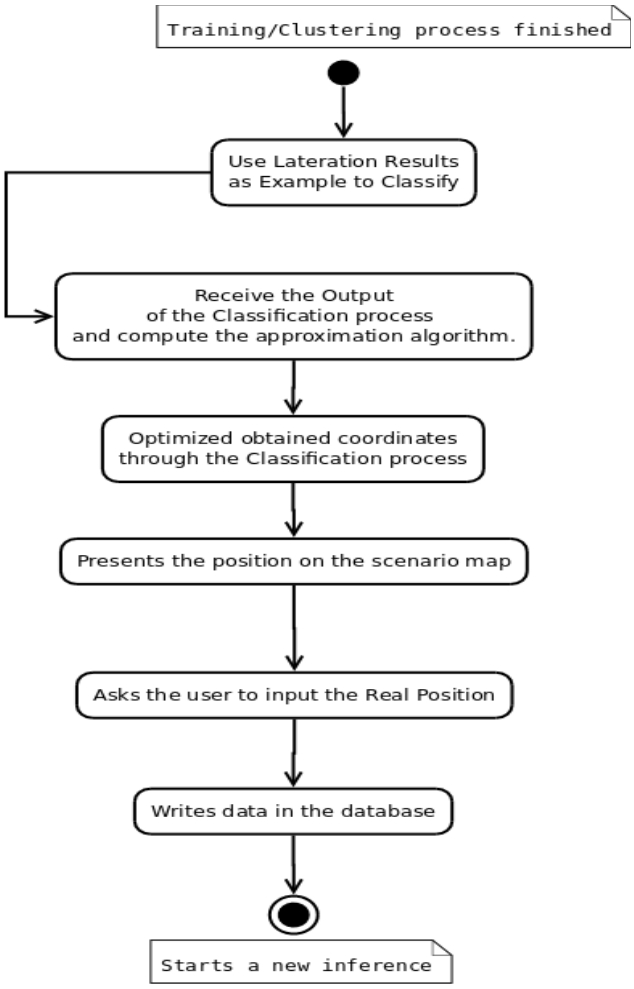

Figure 3.10: Position inference and Real Position insertion states.

3.4.4 Database

The information about the places and Access Points available is stored in the mobile device's SD card and the database is developed with SQLite, which is the standard for databases used by Android. The model of the implementation done is in the following image.

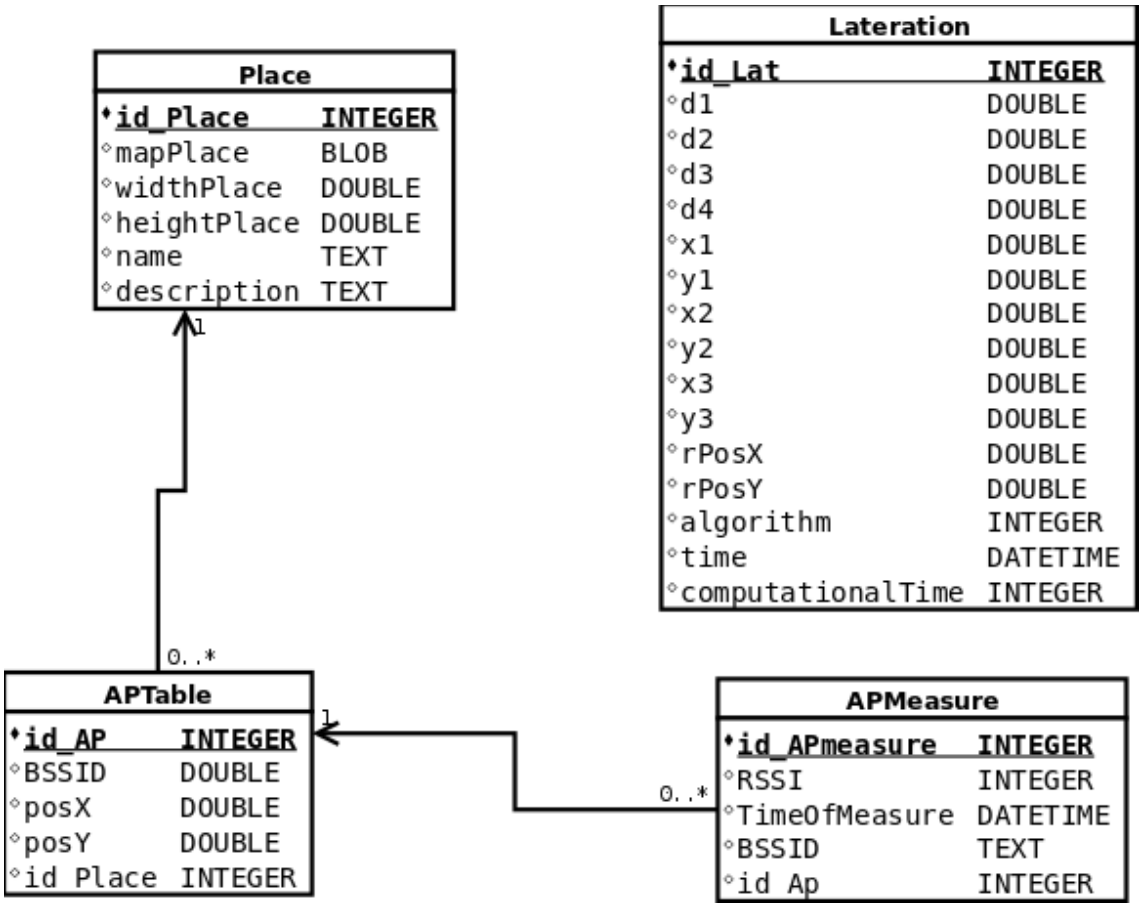

Figure 3.11: Prototype data model.

Comparing with the database of [C12], it can immediately be seen that minor changes were done in the database structure. The table Lateration was the only one that suffered changes, specifically the addition of the *rPosX*, *rPosY*, *algorithm* and *computationalTime* fields. The fields *rPosX* and *rPosY* were added because there was a need of saving in the database the real position's coordinates; the field *algorithm* was added because it was necessary to select which ML algorithm to use and hence each entry had to have the information about which technique inferred its position; the *computationalTime* was created to know how much time each entry's inferred position actually took to be inferred.

The remaining tables were not changed or modified, although it was implemented several SQL functions to retrieve information from them.

3.4.5 Interface

The goal of the current Dissertation is not the one of having a commercial application where the end-user can infer positions in each one of his/her scenarios. Hence, since the previous iteration of the current project, not too many changes were done in terms of the

interface, apart from the addition of some options that ease the testing process and the insertion of data necessary for the algorithms to work properly.

3.5 Summary

The purpose of the whole Chapter 3 was to explain which were the approaches chosen, how they were implemented, how they are supposed to be evaluated, how the whole application is structured and how the final application looks like.

As it was stated several times before, this is an ongoing project and it was already when this Dissertation started. This means that all the decisions taken were done accordingly to what has been done before. The iteration of the project has different goals than the previous one, which were to build a system that through the RSSI signals sent by Access Points and through the usage of the Lateration algorithm, could infer a position in an indoor environment. The implemented techniques achieved the main goals of the thesis, although the position inference contained a high Distance Error if compared to the real position where the mobile device was located.

Hence, this second iteration of the project intends to minimize that error using different Machine Learning techniques, testing and applying them in the same scenarios as the ones chosen for the first iteration. Because of the fact that there are too many different approaches using these ML techniques, three were chosen to be implemented and tested. The goal is to, in the end, be able to choose one that has better overall performance than others, based on several comparison criteria, either using descriptive statistical analysis and other metrics that are intrinsically purposed for algorithms of this kind.

That is what the next chapter is about. It will be evaluated, using different metrics, tools and for different application scenarios, which is the most effective algorithm in terms of each one of the most important of the measured data, which is the influence of the number of positions used for training/grouping in the accuracy of each technique and an evaluation will be done according to it in order to infer the final conclusions about the whole project.

4 Tests, Results and Discussions

This chapter is where the results achieved through the usage of the implemented algorithms in the two scenario setups are revealed, compared and discussed. An introduction to how that presentation will be done is the first part of the chapter, followed by a detailed description of the scenarios. Further, the results without and with the usage of ML techniques will be listed, analyzed and discussed. Posteriorly, comparisons between the obtained results using each one of the implemented algorithms and the position inference without any AI method will be done. Ahead, comparisons between the obtained results and the results presented in the chosen literature pieces will be established. The chapter finishes with a summary, with the goal of grouping the different conclusions obtained by the analysis done in the previous parts, evaluating all the work done.

4.1 Introduction

Structurally, it is possible to use different approaches to the way results are exposed and compared, according to the dissertations and papers studied in Chapter 2. In this chapter's case, it will be done a description of the statistical analysis metrics related with the samples of data gathered, either using no Artificial Intelligence in the position's inference and using the implemented ML techniques. Further, it will be presented the ML metrics to evaluate the performance of each one of the implemented algorithms. Each one of the tests that were done will be evaluated according to the details explained above. Lastly, this sub-chapter will explain which was the approach used to validate the data gathered for the Training process and the one for algorithm testing. The target of this last process presented in this chapter is to demonstrate that the improvements or deterioration of the Mean Distance Error obtained by each one of the algorithms is dependent on the algorithms themselves and not in the chaos of the RSSI signal reception.

In order to evaluate the results that will be presented throughout this chapter, it will be used a set of statistical variables that can be easily and automatically calculated by the software itself. Each one of those metrics intends to conclude different facts about the tests and about the way the algorithms behave in each of them. Basically, the goal of analyzing the results through these metrics is to understand if the results presented by each of the algorithms are valid and in case they're not, to be able to explain in each case why they are behaving the way they are. Also, these calculated metrics can be used in the future work in order to adapt training parameters and consequently, to build better training models that can minimize the Distance Errors of every inference.

The above referred metrics are:

- **Mean Distance Error (MDE)**: defined by the Equation 4.1, which calculates the square root of the sum of the distance errors in x and y.
- **Standard Deviation of Distance Error (σ):** defined by the Equation 4.2, the Standard Deviation metric represents the variation or dispersion that there is from the Average Distance Error.
- **Variance of Distance Error (σ^2):** defined by the Equation 4.3, it represents the average of squared differences from the mean.
- **Maximum Distance Error (m) :** defined by the Equation 4.4, it represents the maximum obtained distance error between the real position and the inferred one by the mobile device in a certain population of data.
- **Minimum Distance Error (n) :** defined by the Equation 4.5, it represents the minimum obtained distance error between the real position and the inferred one by the mobile device in a certain population of data.
- **Bias of x and y inferred positions (b_x and b_y) :** defined by the Equations 4.6 and 4.7 - respectively for x and y -, it represents how non-randomly the Measured Positions where selected. A low bias in the inference of x and/or y means the ML algorithms implemented are outputting disperse solutions for different positions and a high bias represents that they are producing similar solutions for different positions.

  Let R_x represent the real x coordinate, R_y the real y coordinate, I_x the inferred x coordinate, I_y the inferred y coordinate, N the number of samples in a population P, DE_t the Distance Error of the position t, where $t \in \{1,..,N\}$, m the Maximum Distance Error, n the Minimum Distance Error, b_x the bias of the inferences of the coordinate x and b_y the bias of the inferences of the coordinate y:

$$MDE_t = \frac{\sqrt{\left(R_x^t - I_x^t\right)^2 + \left(R_y^t - I_y^t\right)^2}}{N} \tag{4.1}$$

$$\sigma = \frac{\sqrt{\left(\left(\sum_{(i=1)}^{(N-1)} \left(DE_i - MDE_i\right)^2\right)\right)}}{N} \tag{4.2}$$

$$\sigma^2 = \frac{\sum_{(i=1)}^{(N-1)} \left(DE_i - MDE_i\right)^2}{N} \tag{4.3}$$

$$m = max\left(DE_i, DE_{(i+1)}, DE_{(i+2)}, \ldots, DE_{(N-1)}\right), i \in \{1, \ldots, N\} \tag{4.4}$$

$$n = min\left(DE_i, DE_{(i+1)}, DE_{(i+2)}, \ldots, DE_{(N-1)}\right), i \in \{1, \ldots, N\} \tag{4.5}$$

$$b_x = \frac{\left(\left|\sum_{(i=1)}^{(N-1)} R_x^i\right| - I_x^i\right)}{N} \tag{4.6}$$

$$b_y = \frac{\left(\left|\sum_{(i=1)}^{(N-1)} R_y^i\right| - I_y^i\right)}{N} \tag{4.7}$$

The equations above represent how these metrics are automatically calculated by the Android application. Intense reviewing was done either to the code that produce the results and to the results themselves in order to guarantee that they were being properly calculated. Although these metrics are necessary do to a descriptive statistical analysis of the generated populations, there is a set of metrics that is also necessary to analyze the performance of each one of the Machine Learning implemented algorithms. Hence, according to [TII10], the evaluation metrics that determine if an algorithm is performing better or worse than others are:

- **Hamming Loss (HL):** the amount of incorrectly classified labels in relationship with the total number of labels. The optimal value is 0, since this is a loss function.
- **Exact Match Ratio (EMR):** Indicates the percentage of samples which labels were all correctly classified.

Since the problem its being solved is a multi-label classification problem (the system has to infer x and y), achieving a high EMR and a low HL would be ideal. This is a difficult achievement to do, since the number of existent positions in a certain scenario is massively big. For instance, if an algorithm infers (1,5m; 3,1m) and the real position is (1,55m; 3,15m), this classification would not count as an EMR, hence it would contribute for the increase of HL. Because of this fact, it was decided to establish that:

1. A position inference is classified as correct, if the Distance Error between the real position and the position inferred by the mobile device is lower or equal to 1m in the Meeting Room scenario and lower or equal to 2m in the TiZ Entrance Hall scenario.
2. A position inference is classified as incorrect, if the Distance Error between the real position and the position inferred by the mobile device is higher than 1m in the Meeting Room scenario and higher than 2m in the TiZ Entrance Hall scenario.

Like this, the evaluation of the implemented ML algorithms using the ML evaluation metrics will be useful to understand which one is better, which would not happen if they all had Hamming Loss close to 100% and Exact Match Ratio close to 0 %.

Although the results that matter for comparisons are the Mean Distance Errors obtained by each test, the statistical comparisons have to use as input the variables that generates them. Because of the fact that the information that is commonly used by the approach of [C12] and the one implemented throughout this Dissertation is the distances from the mobile device to the existent Access Points – that is calculated by the Lateration algorithm – and since those distances are dependent on the RSSI signals sent by each Access Point, it won't be detailed any statistical evaluation of those in this document, due to the fact that it was previously done by [C12] in similar testing conditions. Hence, comparisons between the approaches that do not use any Artificial Intelligence will be presented and explained through descriptive statistical analysis, with no statistical correlation established between them.

Relatively to comparisons done between the approach with no AI and the approaches where ML algorithms were implemented, correlations will be established. The first step will be of proving/disproving the null hypothesis and only then comparisons between them can be done. The data gathered either in the Meeting Room scenario and the TiZ Entrance Hall scenario, throughout the tests done in each one of them, varies from test to test. The intention of comparing results between ML techniques is to understand which Training configurations improve or deteriorate the variation of the calculated Distance Error between the real position of the mobile device and the position inferred by it. Because the amounts of data gathered per each test done were small and due to the fact that comparisons between each training configuration were needed in order to prove the Hypothesis established in Chapter 4.1, it was decided to use the Wilcoxon Signed Ranks as a statistical analysis tool.

The Wilcoxon Signed Ranks test has several steps. It starts by formulating 2 Hypothesis, being Hypothesis 0 the null hypothesis and Hypothesis 1 the research Hypothesis. The null Hypothesis defines that there was no changed between the first set of values and the second to be compared, while the research Hypothesis asserts the opposite. The next step is to compute the differences between the values in the first and second sets and after they're ranked in ascending order. Posteriorly, it's calculated the numbers of positive and negative differences between the values in the first and second set. The smallest number of these two numbers is used to calculate z. Then, after z is calculated, it will be checked if the z value is lower than -1.96 or higher than 1.96 (in order to be able to achieve conclusions with a confidence interval of 95%). If it is, the null Hypothesis is rejected, hence it's possible to infer that there was a change from the first to the second set; if not, the research Hypothesis is rejected, hence it's possible to infer the opposite. In the context of the current Dissertation, since the goal is to minimize the Distance Error between the real position of the mobile device and the inferred position it calculates, if the number of positive differences calculated is higher than the number of negative differences and if z is lower than -1.96 or higher than 1.96, then it's possible to conclude there was an improvement in the Distance Error, meaning that the Mean Distance Error is smaller in the second set. If the number of positive differences is lower than the number

of negative differences and if z is lower than -1.96 and higher than 1.96, it's possible to conclude that the Mean Distance Error is higher in the second set, meaning there was a deterioration in the accuracy of the position inference.

Decisions relatively to which algorithms perform better or worse in each scenario and in each test, will be taken accordingly to the results obtained by the Wilcoxon Signed Ranks test validation. Further, detailed comparisons will be established according to each algorithm, if the validation occurs; if not, a smaller comparison will be done since the algorithm did not improve in that scenario with that Training configuration.

4.2 Scenarios Setups

Although the goals of the current Dissertation do not include the deployment of a commercial application that infers positions in different scenarios with high accuracy, LatitudeN intends to be able to release commercially such piece of technology in some time. Therefore, the application is targeted mainly to shopping centers and museums, or using other words, spaces that vary either from 15m² to 250m². Henceforth, two different testing scenarios were chosen. The first tests were done in the meeting room of LatitudeN headquarters' office, representing the small scenario setup, with dimensions of 3.12m x 6.25m (19.5 m²). The second tests were performed in the main hall of the TiZ building in Darmstadt, representing the wide areas' scenario specification, with dimensions of 14m x 17.1m (approximately 250 m²).

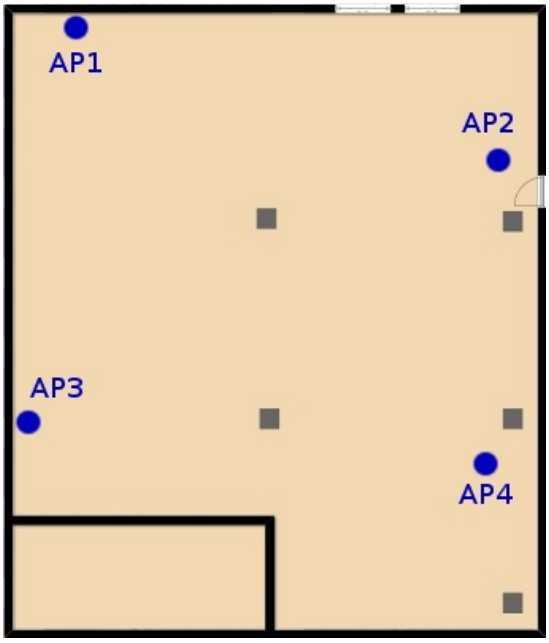

Figure 4.1: TiZ Entrance Hall testing scenario (250 m²).

Table 4.1: Access Points coordinates in the TiZ Entrance Hall scenario.

AP ID	X	Y
1	2	0.4
2	12.5	4.65
3	0.1	9.65
4	12.9	11.8

The blue points in Figure 4.1 represent the Access Points that were used to emit RSSI signals to the mobile device. These positions were not chosen randomly: they follow the same configuration as the one of [C12]. Table 4.1 shows the numerical positions of each of the Access Points.

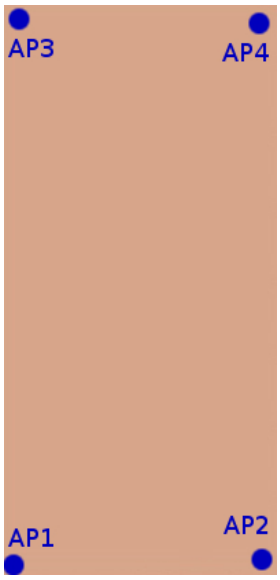

Figure 4.2: Meeting Room scenario (19.5 m²).

Table 4.2: Access Points coordinates in the Meeting Room scenario.

AP ID	X	Y
1	0	6.25
2	3	6.1
3	0.1	0.2
4	3.02	0.1

The blue points in Table 4.2 also represent the Access Points in the Meeting Room scenario. They were placed as close as possible to the corners of the room, so the mobile device used for testing could cover the best signal possible from each one of them in each one of the Measured Positions.

Relatively to the Measured Positions used for data gathering in order to train the algorithms, they differ from test to test. Several different tests were done in an initial testing

phase, starting by doing them in the Meeting Room scenario and moving on after to the TiZ Entrance Hall scenario. The goal of this initial phase of testing was to test Hypothesis 1, in order to confirm that the implemented ML algorithms were performing better than the approach that does not use any Artificial Intelligence, which means they were achieving a lower Mean Distance Error than it. The tests done after were in the TiZ Entrance Hall scenario, where all the other stated Hypothesis – apart from Hypothesis 6, in this initial phase – were put to test.

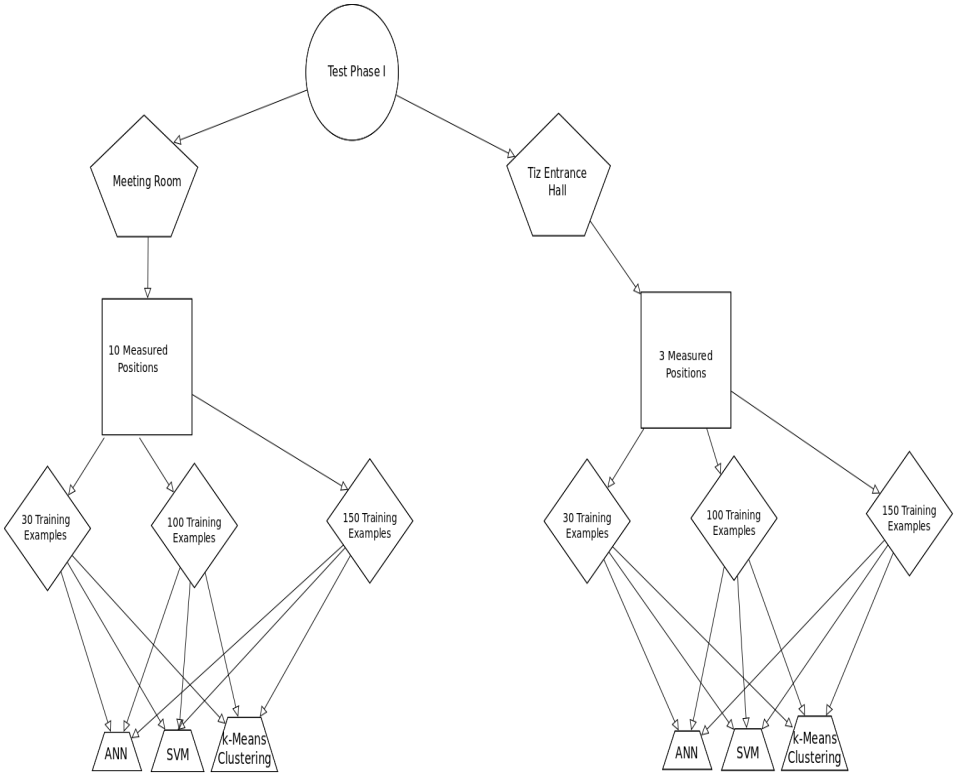

Figure 4.3: Test phase I in both scenarios.

Two kinds of tests were done in the first phase of testing: one using the gathered Training Examples and each new classification done would not count to the next – to which it was called Static Training – and other that each new entry added to the database would count for the next classification process as a Training Example – to which it was decided to call Continuous Training. In other words, the Static Training mode always used the same amount of Training Entries while the Continuous Training one would add each of the classified examples to the next classification to be done. This was done to test the Hypothesis 8.

As it can be seen in Figure 4.3, in the first phase of testing involved testing the 3 implemented ML algorithms in each one of the scenarios, using 10 Measured Positions in the Meeting Room scenario and 3 Measured Positions in the TiZ Entrance Hall scenario. For each one of the scenarios, three different kinds of tests were done using 30, 100 and 150 Training Examples. For the case of the Meeting Room scenario test using 10 different Measured Positions, it was gathered 3, 10 and 15 Training examples per Measured Position and for the

tests in the TiZ Entrance Hall using 3 different Measured Positions the number of Training Examples per Measured Position was of:

- 10 entries per Measured Position for the 30 Training Examples test;
- 34 entries for Measured Position 1, 33 entries for the Measured Position 2 and 33 entries for the Measured Position 3 for the 100 Training Examples test;
- 50 entries per Measured Position for the 150 Training Examples test.

Either Static Training and Continuous Training was done per each one of the test configurations explained previously, in order to be able to test Hypothesis 8.

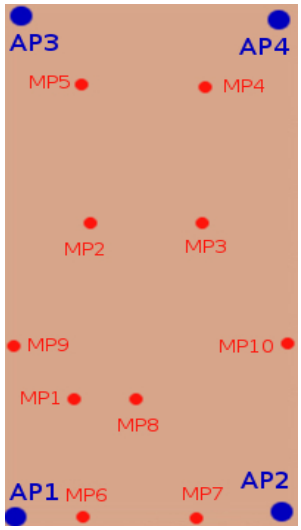

Figure 4.4: Meeting Room scenario with Access Points and Measured Positions.

Table 4.3: Coordinates of the Measured Positions in the Meeting Room scenario.

MP ID	X	Y
1	1	4,65
2	1,2	2,35
3	2,3	2,35

4	2,35	0,85
5	1	0,85
6	1	6,25
7	1,7	5,95
8	1,5	4,65
9	0,3	3,95
10	3,1	3,95

As it's described in Figure 4.4 and detailed in Table 4.3, 10 different positions were measured to use as Training for the ML algorithms implemented. The choice of these positions combined several factors for the Measured Position selection: first, positions were chosen according to some reference points in the room, in order to be avoiding the work of manually measure distances every time tests were done, such as MP1, MP2, MP3, MP4, MP5 and MP8; second, other positions were chosen due to supposed difficulties in obtaining good results, such as MP6, MP7. MP9 and MP10.

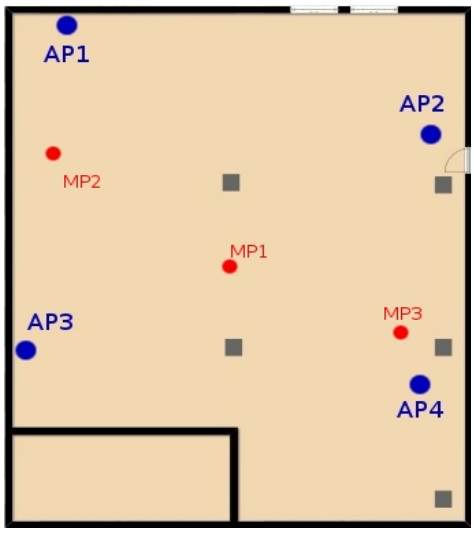

Figure 4.5: TiZ Entrance Hall scenario with APs and MPs .

Table 4.4: Coordinates of the Measured Positions in the TiZ Entrance Hall scenario.

MP ID	X	Y
1	7,2	8
2	1,45	4
3	11	10

In the case of the positions that can be seen at Figure 4.5 and whose positions are detailed at Table 4.4, the choice of them was made taking into account the fact that [C12] did tests in the same positions and henceforth, the 3 Measured Positions are the same as the ones used there. This was done with the goal of easing comparisons between the implemented ML algorithms and the algorithm with no Artificial Intelligence, in order to understand which were the improvements brought to his approach, in terms of Mean Distance Error minimization.

Due to the fact that it was necessary to test Hypothesis 6 in the TiZ Entrance Hall scenario, in order to conclude posteriorly about the dependency of the Mean Distance Error of a wide area scenario with the number of Measured Positions, it was necessary to do another phase of testing. This second phase has different characteristics in relationship to the first phase: in the end of the first phase of testing, statistical analysis was done and tests were eliminated according to following established criteria: each algorithm that the Mean Distance Error rises with the growth of the Number of Training entries, being the type of Training Static or Continuous, will not be tested again.

This was decided because of several reasons: first, the amount of time spent configuring the databases for Training and actually doing the tests was quite high; second, it was necessary to start making decisions out of the gathered data and cutting off options, using the tests done until then; third, the second reason does not invalidate the test of any of the established Hypothesis since, up to the moment of the beginning of the second phase of testing, all of them except Hypothesis 6 can be answered positively or negatively and justified accordingly. Hence, the tests done will be detailed in the sub-chapters ahead with a special focus on the Hypothesis established and on making a final decision about which algorithm behaves better in which situation. Also, all the statistical analysis done will be presented in order to guarantee that the results obtained are valid to justify those hypothesis and consequently the conclusions that will be taken from them.

4.3 Position Inference without ML Techniques

The positioning method that [C12] implemented is done through Lateration (see Chapter 2.2.1). Several approaches were tested and the one chosen was the Linear Least Squares Lateration technique. The goal of the current Dissertation, as it was explained before, is to improve the results achieved by that technique through the add-on of different Machine Learning algorithms and compare either between them to choose the one that behaves better and with the results achieved by [C12]. Because the testing scenarios changed, before testing the ML techniques, it was necessary to gather data to train the algorithms. That process was done using exclusively the approach of [C12] in the testing scenarios.

The first scenario where the data was gathered was the Meeting Room of LatitudeN's office. The data corresponds to 10 different Measured Positions that were used posteriorly to train the algorithms. The results achieved can be seen in Table 4.5.

Table 4.5: Statistical metrics of the Meeting Room scenario using No-AI.

	Number of Training Examples		
	30	**100**	**150**
Mean Distance Error (m)	1.9	1.93	2.2
Mean Variance of Distance Error (m)	0.83	1.09	2.03
Mean Bias X (m)	0.1	0.17	0.24
Mean Bias Y (m)	0.74	0.44	0.53
Mean Standard Deviation of DE (m)	0.91	1.04	1.42
Maximum Distance Error (m)	4.89	4.89	6.2
Minimum Distance Error (m)	0.36	0.1	0.1

The statistical metrics gathered show that the MDE grows in a non-significant way from the 30 to the 100 Training Examples and rises a bit more from the 100 to the 150 TE. Still, the difference of MDE between the 30 TE and the 150 TE (5 times more data gathered) is of 0.3m, which is not a significant change, In terms of Mean Variance of Distance Error, there are barely no differences between 30 and 100 TE, but from 100 to the 150 TE population it doubles. This means that, between the 100 TE population and the 150 one, the variation between the mean difference between the samples' DE is the double in the 150 TE case. The Mean Bias of X and the Mean Bias of Y was stable in the 3 cases. The Maximum DE is the same in the 30 TE and 100 TE cases and higher in the 150 TE (difference of 1.31m between the 30/100 and the 150 TE case) and Minimum DE was close to 0 in the 100 and 150 TE cases and of 0.36m in the 30 TE case which is a small value too.

These 3 presented populations of data were used in both the Continuous and Static Training test configurations for all the 3 implemented ML algorithms in the tests done in the Meeting Room scenario. This means that the calculated statistics for each of the 3 examples had an influence either in the way the algorithms behaved and in the way comparisons will be established.

The second scenario where the data was gathered was the Entrance Hall of the building where LatitudeN's headquarters are. The first test was done using 3 different Measured Positions to train the algorithms. The results achieved can be seen in Table 4.6.

Table 4.6: Statistical metrics of the TiZ Entrance Hall (3 Mps) scenario using No-AI.

	Number of Training Examples		
	30	**100**	**150**
Mean Distance Error (m)	3.24	3.4	3.45

Mean Variance of Distance Error (m)	1.57	1.75	1.47
Mean Bias X (m)	0.44	0.45	0.42
Mean Bias Y (m)	0.98	0.79	0.56
Mean Standard Deviation of DE (m)	1.25	1.32	1.22
Maximum Distance Error (m)	6.07	6.48	6.48
Minimum Distance Error (m)	1.45	0.67	0.67

As Table 4.6 shows, the MDE grows lightly as the number of TE grows from 30 to 150. That growth is a bit more accentuated between the 30 and the 100 TE examples than between the 100 and 150 TE examples. The Mean Variance of DE is stable in the 3 different cases and such is the Mean Standard Deviation of DE and the Mean Bias of X, representing no significant change between the different populations of data. The Mean Bias of Y decreases from 30 to 100 and from 100 to 150 TE cases, meaning that the inferred positions are more variate. The Maximum Distance Error is increases from the 30 to the 100 TE case and stabilizes from the 100 to the 150 case. The Minimum Distance Error decreases from the 30 to the 100 TE case and also maintains the same value from the 100 to the 150 TE population.

Because of the fact that in the 3 different populations gathered the MDE increases as the number of TE per Measured Position increases and the only statistical variable that changes consistently as they do is the Mean Bias of Y, it can be asserted that the fact that the algorithm is inferring more variate final values of y is what is provoking the growth in the MDE. This affirmation can be sustained by observing the change of the Mean Variance of DE, that does not grow consistently as the number of TE per MP increases, which means that it's not affecting the change of the MDE as much as the Mean Bias of Y is, in this case.

Since one of the Hypothesis to test was the influence of the growth in the number of MP in the MDE using the implemented ML algorithms and due to the fact that it's necessary to establish comparisons between the results achieved by them and the ones achieved without them, data was gathered using 6 Measured Positions, for 30, 100 and 150 TE.

Table 4.7: Statistical metrics of the TiZ Entrance Hall (6 MPs) using No-AI.

	Number of Training Examples		
	30	**100**	**150**
Mean Distance Error (m)	4.62	4.66	4.31
Mean Variance of Distance Error (m)	6.92	5.7	6.69
Mean Bias X (m)	0.9	0.84	0.72
Mean Bias Y (m)	1.9	0.77	0.65
Mean Standard Deviation of DE (m)	2.63	2.39	2.59
Maximum Distance Error (m)	10.77	10.77	13.25

Minimum Distance Error (m)	0.99	0.1	0.1

Table 4.7 shows the values of the statistical metrics calculated using the data gathered in the TiZ Entrance Hall scenario for 6 different Measured Positions. The MDE increases from the 30 to the 100 TE populations and it decreases from the 100 to the 150 TE cases, although the change between them is in the order of the tens of centimeters, which is not significant. The Mean Variance of DE significantly decreases from the 30 to the 100 TE cases and increases again from the 100 to the 150 TE populations. The same happens with the Mean Standard Deviation of DE, although the change is less significant than in the Mean Variance of DE metric. The Means Bias of X decreases slightly as the number of TE per Measured Position grows, which means the algorithm is generating more variate x coordinates. In terms of Means Bias of Y, from the 30 to the 100 TE it decreases from 1.92m to 0.77m, which is a huge change. From the 100 to the 150 TE cases it also decreases, but this time the change is not as high as from 30 to 100 TE. The Maximum Distance Error has the value of 10.77m in both the 30 and 100 TE and it finds a new maximum value of 13.25m in the 150 TE case. The Minimum Distance Value goes from 0.99m to 0.1m in the 30 and 100 TE respectively. In the 150 TE case it stays 0.1m.

In these specific sets of tests, the MDE depends less on the Mean Bias of both x and y because the Mean Variance is quite high in all of them, creating instability in the position inference and huge either small and huge distances between the real position and the inferred position. From this fact, it can be concluded that, more than just trying to keep a stabilized value of the Mean Bias of x and y, it's also necessary to try to keep a low value of the Mean Variance of DE.

Due to the fact that every time a data set of data had been gathered in the TiZ Entrance Hall scenario, it had been followed by tests using the implemented ML techniques in the same scenario with the same configurations, in the case of the 9 MPs test, the only number of TE that was necessary to gather because of the exclusion of tests done between each test phase, was the 30 TE case, which the statistical metrics are presented in Table 4.8.

Table 4.8: Statistical metrics of the TiZ Entrance Hall without the usage of ML for 9 MPs.

	Number of Training Examples
	30
Mean Distance Error (m)	5.6
Mean Variance of Distance Error (m)	7.33
Mean Bias X (m)	0.97
Mean Bias Y (m)	2.49
Mean Standard Deviation of DE (m)	2.71

Maximum Distance Error (m)	10.77
Minimum Distance Error (m)	1.45

In this case, the MDE was of 5.6m, the Mean Variance of the DE, the Mean Standard Deviation of DE and the Mean Bias of Y were quite high, although the Mean Bias of X is low. The Maximum Distance Error obtained was of 10.77m and the Minimum Distance Error of 1.45m. Because of the high Mean Variance of the DE and the Mean Bias of y high values, the MDE reached the highest value seen without usage of ML techniques to infer position.

4.4 Position Inference using ML Techniques

Three different ML techniques were implemented as an approach to the Distance Error minimization problem: Artificial Neural Networks, Support Vector Machines and k-Means Clustering. This sub-chapter demonstrates the results obtained from the tests done with each one of them in the two testing scenarios using different test configurations. For each one of the algorithms, the process to demonstrate their practical performance will have the following steps:

1. Demonstrate the correlation between the Distance Errors obtained in each one of the testing configurations for the two scenarios.
2. Establish comparisons between the descriptive statistics metrics.
3. Establish comparisons between the ML evaluation metrics.

As it was stated before, the depth of these comparisons will be done depending on the improvement or deterioration the technique that is being analyzed shows. This means that algorithms that, in a certain scenario configuration, show that the Mean Distance Error was reduced from a number of Training Entries to another or from a number of Measured Positions to the next will be compared more minutely than ones that show deterioration with the same scenario factors change.

4.4.1 Using Artificial Neural Networks

Let's start by the tests done in the Meeting Room Scenario (Figure 4.4). Three tests were done in the Meeting Room of LatitudeN's office for each of the types of Training (Static and Continuous), with 10 Measured Positions used to feed the Artificial Neural Network. These tests were done in order to study the behavior of the ANN approach in a small-sized scenario.

Table 4.9: MDE using ANN in the Meeting Room Scenario (Static Training).

	Number of Training Examples		
	30	**100**	**150**

Mean Distance Error (m)	2,57	1,73	1,23
Mean Distance Error Change from the Previous test (%)	-	35,54%	23,74%

As it can be seen in Table 4.9, the Mean Distance Error was reduced from 2.57m in the test using 30 Training Entries to 1.73m using 100 Training Entries and to 1.23m using 150 Training Entries. It should be noted that there was no change in the Measured Positions that were used to feed the ANN and that the number of samples gathered was of 39 (3 for each one of the 10 Measured Positions, equalizing 30, and 3 for each one of the New Positions, equalizing 9). The Mean Distance Error change from the 30 Training Entries test to the 100 Training Entries test was of 35.54% and the MDE change from the 100 Training Entries test to the 150 Training Entries test was of 23.74%. The MDE change from 30 Training Entries to 150 Training Entries (5 times more samples that fed the algorithm) is even more significant, with a value of 50.84 %. In terms of accuracy, this means that the ANN in the Meeting Room scenario increased its accuracy in 50.84% with only 5 times more samples used to train it.

The question after this first calculation, right after finishing the tests, was if these improvements were because the algorithm was actually improving the accuracy in the testing configuration and if there was enough change in the positioning inferences in order to show that they were valid results. In order to prove that, correlations between the Distance Errors of each of the tests were done using the Wilcoxon Signed Rank test.

Table 4.10: WSR test for ANN between the Meeting Room scenario tests (Static Training).

	Training Examples Correlations		
	30 – 100	**30 - 150**	**100 - 150**
Number of positive differences	30	31	21
Number of negative differences	9	8	18
T	45	36	171
z	-4,81	-4,94	-3,06

Table 4.10 shows the variables output from the Wilcoxon Signed Rank Test. Since $\alpha = 0.05$, then z had to be lower than -1.96 or higher than 1.96. As it can be seen, z is lower than -1.96 in each one of the correlations done and the number of positive differences between the first and second sets of Distance Errors is higher than the number of negative ones, which means that as the number of entries per position, in the Meeting Room scenario using Static Training improves the accuracy of the solution using ANN.

Table 4.11: Statistical metrics of ANN in the Meeting Room Scenario (Static Training).

Number of Training Examples

	30	100	150
Mean Variance of DE (m)	1,8	1,18	0,56
Mean Standard Deviation of DE (m)	1,34	1,34	0,75
Mean Bias Inferred x (m)	0,04	0,13	0,12
Mean Bias Inferred y (m)	1,94	0,96	0,3
Max DE (m)	5,5	5,55	2,95
Min DE (m)	0,35	0,18	0,13
Hamming Loss (%)	82 %	62%	59 %
Exact Match Ratio (%)	0 %	0 %	0 %

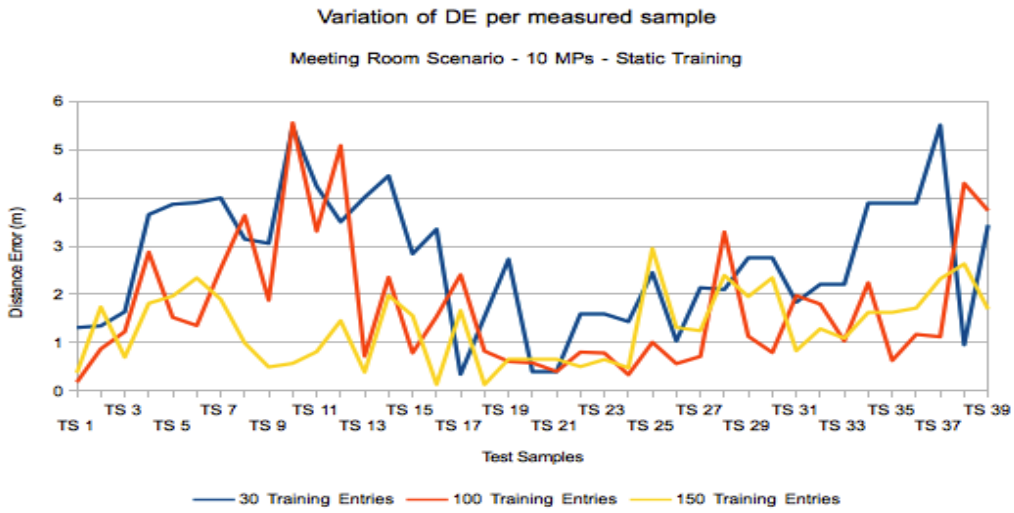

Figure 4.6: Variation of DE per Tested Sample using ST in the MR scenario.

Figure 4.6 shows the variation of DE per Measured testing sample. As it can be seen in the chart, the tests with 30 and 100 Training Examples present huge differences between the Maximum and Minimum Distance Errors and strong variations between position inferences, when compared against the 150 TE test.

Table 4.11 confirms it, by showing that the Mean Variance of either the 30 and the 100 Training Examples tests was above 1m (1.8m and 1.18m, respectively). The Mean Variance of the 150 TE test was of 0.56m, showing an improvement in the Mean Variance of 68.8% relatively to the 30 TE test and of 52.5 % in comparison to the 100 TE test. In terms of Mean Standard Deviation of DE, the 150 TE test had the lowest also, meaning that all the Distance Errors were closer to the Mean DE than in the cases of the 30 and 100 TE test. This becomes clear just by observing the Figure 4.6, where it can be seen that the variations in the Distance Error per measured testing sample was lower than the ones either in the 100 and the 150 TE tests. The Mean Variance of DE and the Mean Standard Deviation of DE shown in Table 4.11 prove the observation.

When it comes to analyzing the Mean Bias of x, Table 4.11 shows that it was quite stable for all the tests. On the other hand, the Mean Bias of y was problematic for the 30 and 100 Training Examples tests, because the first was of 1.94m and the second of 0.96m. The lowest of the Mean Bias of the y inference was the one of 150 TE test, which was of 0.3m. Due to the fact that the Mean Bias of x and y metrics are related with the variation of inferences of both coordinates in the scenario and because the measured Test samples go throughout the whole scenario area – meaning that they are quite diverse -, this allows to conclude that the Mean DE of the 150 TE test was lower than the ones of 30 and 100 Training Examples because the inference of the y coordinate in these last ones had generally more DE than the inference of the x coordinate. The Max DE was the lowest also in the 150 TE test (2.95m) when compared to the 30 and the 100 TE tests (5.5 and 5.55m, respectively). The Min DE was also the lowest in the 150 TE test, although they are quite stable and close to 0 in all the three different tests.

Regarding the ML evaluation metrics, the Hamming Loss was the highest in the 30 TE test and the lowest in the 150 TE test (82% and 59% respectively), although the change was more significant from the 30 to the 100 TE test (from 82 % to 62%, hence a reduction of 20% in the Distance Errors above 1m). The Exact Match Ratio, representing the percentage of times the positioning inference was totally accurate – Distance Error between the real position and the inferred one of 0m -, is 0% for all the cases, hence none of the tests had a 0m Distance Error case.

Some practical conclusions can be taken from these 3 tests in the Meeting Room scenario. As the number of Training Examples rise, the Mean Variance, the Mean Standard Deviation, the Mean Bias of the inferred y coordinate and, consequently, the Maximum and Minimum DE obtained decrease. As a result from that, the Hamming Loss ML metric decreases too, meaning the accuracy of the algorithm is improving as the number of Training Examples rise. Although there was no change in the Exact Match Ratio in each of the tests, if the same statistical characteristics were maintained as the number of Training Examples would rise from 150, most probably the Exact Match Ratio would increase too, decreasing the MDE.

Table 4.12: MDE using ANN in the Meeting Room Scenario (Continuous Training).

	Number of Training Examples		
	30	**100**	**150**
Mean Distance Error (m)	1,71	1,76	1,47
Mean Distance Error Change from the Previous test (%)	-	-3.46%	16.52%

When Static Training tests were finished, Continuous Training tests started for the Meeting Room scenario. As Table 4.12 shows, the variation of the MDE between the 30 and 100 Training Examples test is close to 0 and the one between 100 and 150 TE tests is of 16.52%. The same process for validation was used as in the Static Training case, where the step right after doing the tests was of evaluating if the improvement achieved in this case between 30

and 150 TE for training of the ANN was due to the fact that the algorithm performed better when the number of TE would grow. Hence, using the same statistical test as in the Static Training statistical evaluation case, it was chosen to use the Wilcoxon Signed Rank test, due to the same reasons as then.

Table 4.13: WSR test between the Meeting Room scenario tests (Continuous Training).

	Training Examples Correlations		
	30 – 100	**30 - 150**	**100 - 150**
Number of positive differences	16	25	29
Number of negative differences	23	14	10
T	504	105	55
z	1.59	-3.97	-4.67

Table 4.13 shows the output variables from the Wilcoxon Signed Rank test. Because $\alpha = 0.05$, then z had to be lower than -1.96 or higher than 1.96 for the null Hypothesis to be rejected. In the case of the 30 to 100 TE correlation, z = 1.59, which is lower than 1.96 and higher than -1.96, meaning that the null Hypothesis can't be rejected, hence there is no difference from the DE obtained by each population. The correlations 30-150 and 100-150 TE tests show values below -1.96, which means that, in both cases, the null Hypothesis has to be rejected, henceforth there was change in the two populations compared. Because the number of negative differences is lower than the number of positive differences in the 30-150 and 100-150 TE correlations, it can be concluded that the existent change by increasing the number of Training Examples per Measured Position enhanced the accuracy of the algorithm to infer positions.

Table 4.14: Statistical metrics of ANN in the Meeting Room Scenario (CT).

	Number of Training Examples		
	30	**100**	**150**
Mean Variance of DE (m)	1,48	1,7	1,26
Mean Standard Deviation of DE (m)	1,24	1,3	1,12
Mean Bias Inferred x (m)	0,01	0,2	0,18
Mean Bias Inferred y (m)	0,04	0,07	0,42
Max DE (m)	4,72	4,99	5,81
Min DE (m)	0,07	0.44	0,1
Hamming Loss (%)	62 %	33.3%	59%
Exact Match Ratio (%)	0	0	0

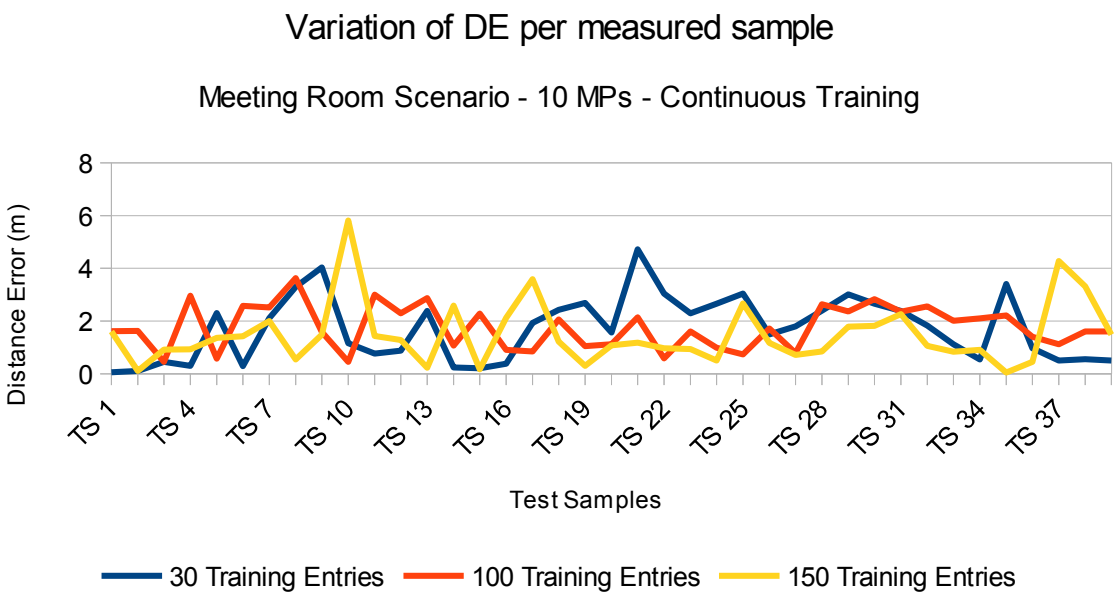

Figure 4.7: Variation of DE per Tested Sample using ST in the MR scenario.

As it can be seen in Figure 4.7, in all the 3 tests the variation of DE was full of highs and lows. By looking at the Mean Variance of DE and the Mean Standard Deviation of DE inTable 4.14, it can be seen that, although either the Mean Variance of DE and the Mean Standard Deviation of DE values were quite similar with each others, they were all high values in relationship to the scenario's dimensions. This explains the variation that can be observed in Figure 4.7.

Relatively to Mean Bias of the inferred x and y coordinates, all the values were low too, meaning that the algorithms inferred a range of positions close to the dimensions of the testing scenario, which was a good sign due to the fact the measured Testing Positions were spread all over the room. Still, because the Mean Variance of DE and the Mean Standard Deviation of DE for all the tests was so high and the Mean Bias of either x and y is so low for the same cases, it was concluded that using Continuous Training and ANN would produce errors almost as high as the scenarios size. For instance, in the Meeting Room scenario that has dimensions of 3.12m x 6.25m, if a test is done in the position (0m,0m) the algorithm will have times that it will infer a really close position and other times that it will infer, as an example, (3m,3m). The Maximum DE were all really high for the 3 tests and the Minimum DE were all close to 0, specially in the 30 TE test and the 150 TE test. It was slightly higher in the 100 TE test, with the value of 0.44m.

In terms of ML metrics evaluation, the Hamming Loss values were of 62% for the 30 TE test, of 72% for the 100 TE test and of 59% for the 150 TE test. There was a slight deterioration of accuracy from the 30 TE test to the 100 TE test of 10% less and a 13% more, respectively. In terms of Exact Match Ratio, none of the tests obtained a single sample that had 0 of DE.

There are no practical conclusions that can be taken directly from the analysis of the Continuous Training tests in the Meeting Room scenario, apart from the fact that there is no equilibrium between Mean Bias and Mean Variance, which results in a strange progression of MDE as the number of Training Examples grow.

Table 4.15: Comparison of MDE between ST and CT in the MR scenario.

	Number of Training Examples		
	30	**100**	**150**
MDE (m) of Static Training	2,69	1,73	1,23
MDE (m) of Continuous Training	1,71	1,73	1,47

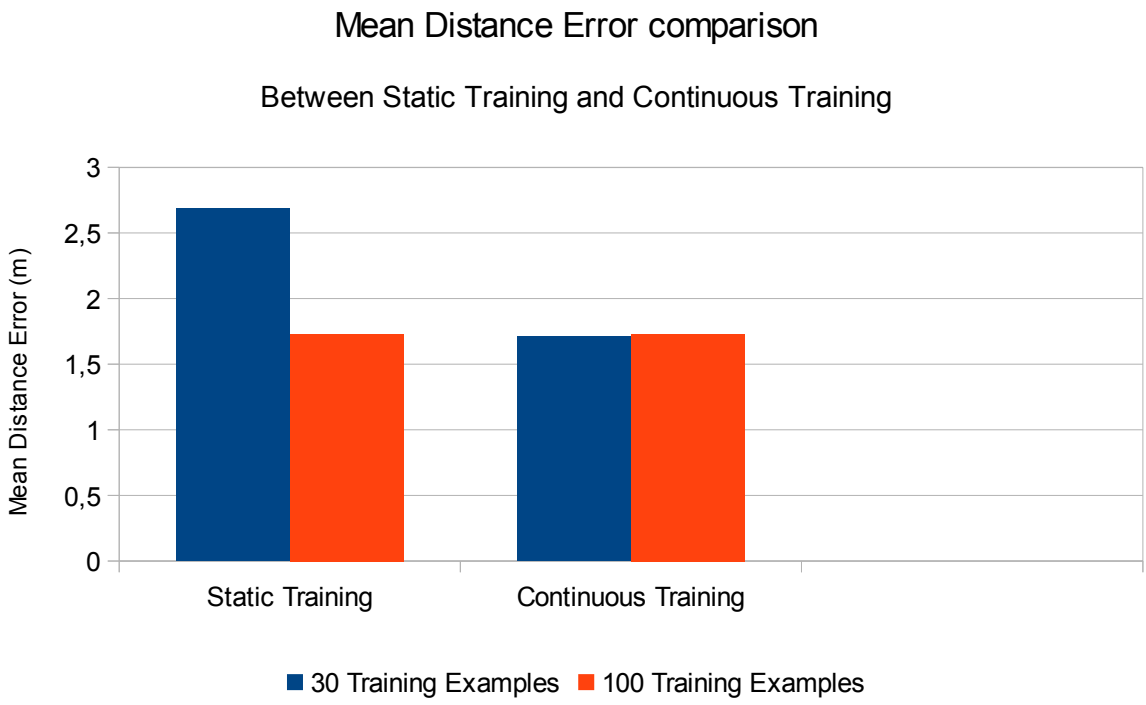

Figure 4.8: Comparison of MDE between Static and Continuous Training.

As it can be seen in Figure 4.8 and in Table 4.15, the Static Training starts with a higher MDE than the Continuous Training approach (2.69m and 1.71m, respectively) in the 30 TE tests, achieving the same MDE in the 100 TE test (1.73m both) and in the 150 TE test, the Static approach had a significantly lower DE.

Even though these comparisons can be established directly, because both the approaches were done in the same scenario with the same conditions, there were variables that demonstrate how these changes happened. The Static Training approach suffered a decrease in the Mean Variance levels as the number of TE rose, while the Continuous Training method had its Mean Variance stabilized. The same happened in terms of Mean Standard Deviation and in terms of Mean Bias of y, which allowed the inferred positions to approximate further to the real positions as the number of TE increased.

Conclusively about the ANN approach in the Meeting Room scenario, the Continuous Training was shown not to be able to improve the results as well as the Static Training in terms of MDE minimization using ANN. Henceforth, from that moment on, Continuous Training stopped being an option when using ANN to infer positions, even when the testing scenario changed to the TiZ Entrance Hall.

Table 4.16: MDE using ANN in the TiZ Entrance Hall Scenario (3 MPs).

Number of Training Examples

	30	100	150
Mean Distance Error (m)	1.83	1.73	1.44
Mean Distance Error Change from the Previous test (%)	-	5,46 %	16,9 %

As Table 4.16 shows, the MDE in TiZ Entrance Hall using 3 MPs test and Static Training as training configuration decreases as the number of Training Examples increase. From the 30 to the 100 TE tests, the decrease was of 5.46% and from the 100 to the 150 TE tests it was of 16.9 %. The same approach of using the Wilcoxon Signed Rank test used before was done for these populations also, in order to understand if there was a significant change in the DE values between populations of data.

Table 4.17: WSR test between the TiZ Entrance Hall scenario tests (3 MPs).

Training Examples Correlations

	30 - 100	30 - 150	100 - 150
Number of positive differences	8	7	11
Number of negative differences	7	8	4
T	77	84	10
z	0.97	1.36	-2.83

Table 4.17 shows the output variables of the Wilcoxon Signed Rank test. As it can be seen, from the 30-100 TE correlation we can conclude there was no significant change, confirming the null Hypothesis, since the value of z = 0.97 is lower than 1.96 and higher than -1.96. The same happened from the 30-150 TE correlation, where z = 1.36 was lower than 1.96 and higher -1.96, confirming the null hypothesis for this correlation. Between 100 and 150 TE though, the value of z was of -2.83, which was lower than -1.96, confirming the research hypothesis. In this last correlation, the number of positive differences is much higher than the number of negative differences, which indicates that the change was towards the improvement of the accuracy of the algorithm.

Table 4.18: Statistical metrics of ANN in the TiZ Entrance Hall scenario (ST – 3 MPs)

Number of Training Examples

	30	100	150
Mean Variance of DE (m)	1,48	1,7	1,26
Mean Standard Deviation of DE (m)	1,24	1,3	1,12
Mean Bias Inferred x (m)	0,01	0,2	0,18
Mean Bias Inferred y (m)	0,04	0,07	0,42
Max DE (m)	4,72	4,99	5,81

Min DE (m)	0,07	0.44	0,1
Hamming Loss (%)	62 %	72%	59%
Exact Match Ratio (%)	0	0	0

Table 4.18 presents the values obtained by doing the statistical analysis of the algorithm using different numbers of Training Examples, where it can be seen that the values of Mean Variance of DE and the Mean Standard Deviation of DE are quite stable between all the tests, being slightly higher in the 100 TE test than in the other two. The Mean Bias of Inferred x is close to 0 in the 30 TE test and grows slightly to 0.2 in the 100 TE test, stabilizing in 0.78 in the 150 TE case. The Mean Bias of Inferred y is close to 0 in the 30 TE test, grows lightly in the 100 TE case and grows severely in the 150 TE test. This indicates a positive uplift, since a higher Mean Bias means less positions inferred by the algorithm and in this case the tests were done in only 3 MPs, which is a small value of Measured Positions.

The Hamming Loss values increase 10% from the 30 to the 100 TE test and decrease 13 % from the 100 to the 150 TE case, which means that more inferred positions possessed a DE lower than 2 meters in the last test. The Exact Match Ratio is 0 % for the 3 cases, hence no position inference achieved full accuracy in relationship to the real position of the mobile device.

As it was explained before, no tests were done using the Continuous Training configuration since it was shown that it was not bringing any improvements to the minimization of the DE in each inference. The step after was to do exactly the same tests in the TiZ Entrance Hall using 6 MPs, in order to study if it would improve the accuracy of ANN in such a big-sized scenario.

Table 4.19: MDE using ANN in the TiZ Entrance Hall Scenario (6 MPs).

	Number of Training Examples		
	30	**100**	**150**
Mean Distance Error (m)	1.91	2.4	2.35
Mean Distance Error Change from the Previous test (%)	0 %	-25,65%	2,08%

Table 4.19 shows the MDE obtained by the 30,100 and 150 TE tests using 6 Measured Positions in the TiZ Entrance Hall scenario. The 30 TE test obtained a MDE of 1.91m for the 30 TE test, 2.4m for the 100 TE test and 2.35m in the 150 TE test – from the 30 to the 100 TE test there was a deterioration of the MDE of 25.65 % and from the 100 to the 150 TE test there was an improvement of 2.08%, which is not significant. Comparing with the 3 MPs tests, the 30 TE test of the 6 MPs MDE value is quite close to the one obtained with the 3 MPs 30 TE value (1.91 and 1.83m, respectively), which does not happen between both the 100 TE tests and the 150 TE tests where the differences between the respective Mean Distance Errors are much higher (1.73m to 2.4m in the 100 TE tests and 1.44m to 2.35m in the 150 TE cases).

Despite this deterioration when rising the number of Measured Positions in the 100 and 150 TE cases, correlations between the populations were done anyway, in order to be able to understand if the diminish in the accuracy of the algorithm was related with the algorithm itself or to any other external factor, like the chaos in the RSSI signal transmissions.

Table 4.20: WSR test between the TiZ Entrance Hall scenario tests (6 MPs).

	Training Entries Correlations		
	30 - 100	**30 - 150**	**100 - 150**
Number of positive differences	8	6	5
Number of negative differences	22	24	25
T	312	374	345
z	1.63	2.91	2.31

Table 4.20 shows the values output by the Wilcoxon Signed Rank test from the correlations between 30 and 100 TE tests, 30 and 150 TE tests and 100 and 150 TE tests. The value of z is of 1.63 in the 30 and 100 TE correlation test, which is lower than 1.96 and higher than -1.96, confirming the null hypothesis, proving there are no changes between the Distance Errors obtained in the first and second tests done for 6 MPs. The value of z in the 30 and 150 TE correlation test was of 2.91, which is higher than 1.96, meaning that the null hypothesis was rejected and that there is change between the two populations. The same happened in the 100 to 150 TE correlation test, which z had a value of 2.31 that is higher also than 1.96, hence the research hypothesis was confirmed and it's proved there is change between these 2 populations of Distance Errors.

In the 30 to 150 TE correlation test, the number of negative differences is much higher than the number of positive ones, indicating that the change between the 2 populations is towards the deterioration of the accuracy of the ANN in this environment, confirming what Table 4.19 shows. The same happens with between the 100 and 150 TE tests, meaning that although Table 4.19 shows an improvement of 2.08% in the MDE, the change is still negative in relationship to the growth of the number of Training Examples between the 2 tests.

Table 4.21: Statistical metrics of ANN in the TiZ Entrance Hall scenario (ST – 6 MPs)

	Number of Training Examples		
	30	**100**	**150**
Mean Variance of DE (m)	1,48	1,7	1,26
Mean Standard Deviation of DE (m)	1,24	1,3	1,12
Mean Bias Inferred x (m)	0,01	0,2	0,18
Mean Bias Inferred y (m)	0,04	0,07	0,42

Max DE (m)	4,72	4,99	5,81
Min DE (m)	0,07	0.44	0,1
Hamming Loss (%)	62 %	72%	59%
Exact Match Ratio (%)	0	0	0

Table 4.21 presents the descriptive statistical metrics obtained by the 3 tests done using Static Training in the TiZ Entrance Hall scenario for 6 MPs. As it can be observed, the value of the Mean Variance of the DE and the value of the Mean Standard Deviation of DE does not change significantly between tests. Also, these values are quite similar to the ones obtained by the 3 MPs test, which means that the algorithm behaved similarly in terms of position inference for the 2 different configurations. The Mean Bias either of x and y also have the same kind of progression as the ones in the 3 MPs test, strengthening the affirmation done in the previous sentence. The Maximum DE grows progressively when comparing test-by-test, from the value of 4.72m in the 30 TE test, to 4.99m in the 100 TE test and to 5.81m in the 150 TE test. The Minimum DE is close to 0 in the 30 and 150 TE tests and of 0.44m in the 100 TE test, which is also not far from the real position in that single case in such a wide scenario as the TiZ Entrance Hall is.

The Hamming Loss and the Exact Match Ratio have the same exact values as the ones obtained by the 3 MPs test, which shows that the algorithm did not improvement its performance by adding more different Measured Positions to the training data set.

The last test done using ANN to infer positions in the TiZ Entrance Hall was done using 9 MPs and the only test done was with 30 TE. This was a decision taken based on the lack of improvements in the MDE in the 6 MPs tests by adding a higher number of TE.

Table 4.22: MDE using ANN in the TiZ Entrance Hall Scenario (9 MPs).

Number of Training Entries	
	30
Mean Distance Error (m)	2,07

The obtained MDE in the 9 MPs test using 30 TE was of 2.07m. It shows again a very slight deterioration when comparing with the 3 MPs and the 6 MPs 30 Training Examples tests, which allowed the conclusion that the increase of number of MPs in the TiZ Entrance Hall scenario did not improve the accuracy of the ANN algorithm when inferring a position. Still, due to the fact that in each one of the tests more MPs were added but the total number of TE was maintained, this small worsening in the MDE as the number of MPs rise can be assigned to the fact that the number of TE per MP decreased from test to test – in the case of the 3 MP test, the number of TE per MP was of 10, in the case of the 6 MP test the number of TE per MP was

of 5 and in the case of the 9 MP test the number of TE per MP was of 3 or 4, in some randomly chosen Measured Positions.

Table 4.23: Statistical metrics of ANN in the TiZ Entrance Hall scenario (ST – 9 MPs)

	Number of Training Examples
	30
Mean Variance of DE (m)	1,48
Mean Standard Deviation of DE (m)	1,24
Mean Bias Inferred x (m)	0,01
Mean Bias Inferred y (m)	0,04
Max DE (m)	4,72
Min DE (m)	0,07
Hamming Loss (%)	62 %
Exact Match Ratio (%)	0

The value of the obtained statistics of the 9 MPs 30 TE test are very similar or equal, in some cases, to the ones obtained either in the 3 MPs 30 TE test and the 6 MPs 30 TE test. This proves that the behavior of the ANN algorithm, as the number of MP increased, did not change much in terms of position inference, although the number of TE per MP changed from test to test.

Table 4.24: Comparison of MDE of ANN in the TiZ Entrance Hall Scenario between 3, 6 and 9 MPs.

Number of Training Entries

	30	100	150
MDE of 3 MPs (m)	1.83	1.73	1.44
MDE of 6 MPs (m)	1.91	2.4	2.35
MDE of 9 MPs (m)	2,07	-	-

As it can be observed in Table 4.24, in the case of the 3 MPs test, as the number of TE grows the MDE decreases, which means that the rise of the number of the TE increase the accuracy of the ANN algorithm. The 6 MPs shows a different change, since when the number of TE grows from 30 to 100, the MDE increases and it stabilizes from 100 to 150 TE. Comparing vertically the results, the only case between different number of MPs used as test in which the MDE does not increase much is the 30 TE case.

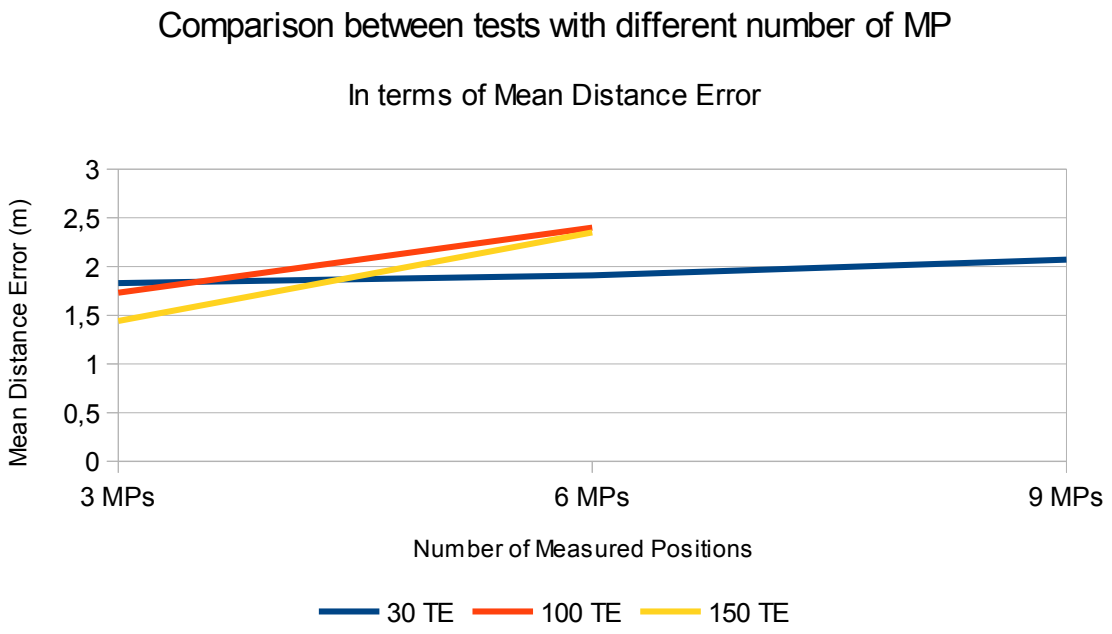

Figure 4.9: Comparison of MDE between 30, 100 and 150 TE.

Figure 4.9 proves that. It can be seen that in the 30 TE case, the MDE slightly increases, although that increase can be asserted to the fact that the number of TE per MP decreases when the number of MP rises, as it was stated before. In the other hand, either the 100 and the 150 TE cases show a growing curve when the number of MP rises. This was the main reason why in the 9 MP tests, the only one that was done was the one that didn't show deterioration in terms of accuracy, hence only the 30 TE test was chosen. Since it also did not show improvements in the accuracy, it was decided not to do another tests with 12 MP using ANN.

4.4.2 Using Support Vector Machines

Exactly like with the ANN approach, the first tests done using the SVM algorithm were in the Meeting Room scenario, right after the data without usage of ML was gathered. The criteria to remove further tests followed the same rules as in the ANN approach, meaning that if Continuous Training didn't show improvements in terms of accuracy in a specific testing scenario and if it didn't improve the accuracy when comparing with the results of Static Training, it would be removed from further tests; also, if some configuration using the SVM obtained results was worse than the approach without the usage of ML algorithms, further tests with the SVM algorithm would not be done.

Table 4.25: MDE using SVM in the Meeting Room Scenario (Static Training).

	Number of Training Examples		
	30	**100**	**150**
Mean Distance Error (m)	2.37	2.27	2.66
Mean Distance Error Change from the Previous test (%)	-	4.2 %	-17,18 %

The first test done with the SVM algorithm was using Static Training as training configuration and using the same tests structured used until this point. As Table 4.25 demonstrates, the MDE in the Meeting Room scenario decrease slightly from the 30 TE test to the 100 TE test and increased a bit more slightly from the 100 TE test to the 150 TE one. Still, again with the goal of proving or disproving the hypothesis that the change between the generated Distance Errors of all the inferred positions using SVM did not change between the different number of TE, the Wilcoxon Signed Rank test was used.

Table 4.26: WSR test for SVM between the Meeting Room scenario tests (ST).

	Training Examples Correlations		
	30 – 100	**30 - 150**	**100 - 150**
Number of positive differences	23	18	14
Number of negative differences	16	21	25
T	136	549	455
z	-3.54	2.21	0.9

Table 4.26 shows the output variables of the Wilcoxon Signed Rank test, used to correlate the populations generated by the usage of the SVM algorithm with different Training Examples. It can be observed that the value of z in the 30 TE to 100 TE correlation test is of

-3.54, which is lower than -1.96, hence the null hypothesis is rejected and the research hypothesis can be asserted. The same happened in the correlation between 30 and 150 Training Examples' tests, with an output z of 2.21, higher than 1.96. In the 100 to 150 correlation test, the output z is of 0.9, which is higher than -1.96 and lower than 1.96, proving the null hypothesis, meaning that there is no change between those two tests. In the cases where there was change, the change was towards more accuracy in the 30 TE to the 100 TE correlation test, due to the fact that there are more positive differences than negative ones and towards less accuracy in the inferred positions by the mobile device in the 30 TE to 150 TE correlation tests, since there are more negative differences than positive ones. This proves what Table 4.25 shows in terms of percentile change from test to test.

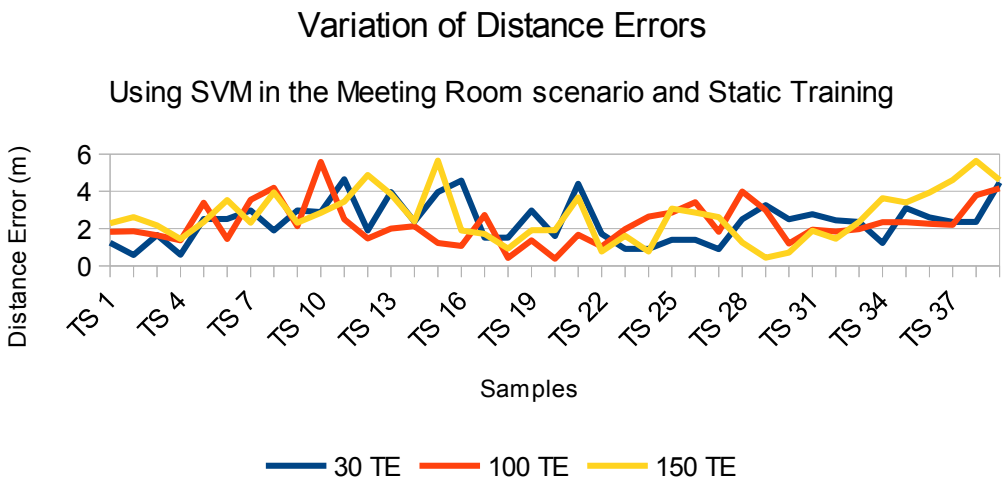

Figure 4.10: Variation of DE in the 3 tests done in the MR scenario using ST.

Figure 4.10 shows that the variation between Distance Errors obtained in the 150 TE test is bigger than in the other 2 tests. Also, the number of samples which the Distance Error is above 2m of distance is quite big for all the tests. In order to understand better these facts, a descriptive statistical analysis was done, evaluating several metrics that were obtained from the population of test samples gathered.

Table 4.27: Statistical metrics of SVM in the Meeting Room Scenario (ST).

	Number of Training Examples		
	30	**100**	**150**
Mean Variance of DE (m)	1.24	1.21	1.8
Mean Standard Deviation of DE (m)	1.12	1.1	1.34
Mean Bias Inferred x (m)	0.92	0.21	0.54
Mean Bias Inferred y (m)	0.8	0.92	1.48
Max DE (m)	4.64	5.58	5.64
Min DE (m)	0.58	0.39	0.43

Hamming Loss (%)	87.18 %	94.49 %	87.18%
Exact Match Ratio (%)	0 %	0 %	0 %

In terms of descriptive statistics generated using the obtained populations of data from the tests using the SVM algorithm in the Meeting Room scenario and Static Training as training configuration, Table 4.27 shows that the Mean Variance of DE and the Standard Deviation of DE slightly decreases from the 30 to 100 TE cases and increases fairly again in the 150 TE test. This may be one of the reasons why the MDE in the 150 TE test increased. In terms of Mean Bias of Inferred x, decreased significantly from the 30 TE test to the 100 TE test, indicating more variations in the x coordinate inference and increases again between the 100 TE to the 150 TE tests. The Mean Bias of Inferred y is quite stable in the 30 TE and 100 TE tests and increases significantly from the 100 TE test to the 150 TE one. This may be another reason why the MDE increased in the 150 TE test, when compared to the previous ones. The Maximum DE is high in the 3 cases, being the lowest in the 30 TE one. The Minimum DE is low for the 3 TE tests and the change between them is also low, in the order of tens of centimeters.

The Hamming Loss is always close to 90% in the 3 cases, meaning that the majority of the inferred positions are more than 2 meters of distance away of the real positions, which for such a small scenario is huge. The Exact Match Ratio is 0 in every sample of the 3 tests done.

After the Static Training test, it was done a Continuous Training test in the same scenario using SVM and with the same test structure as the ones before. The goal was to compare the behavior between Static Training and Continuous Training tests.

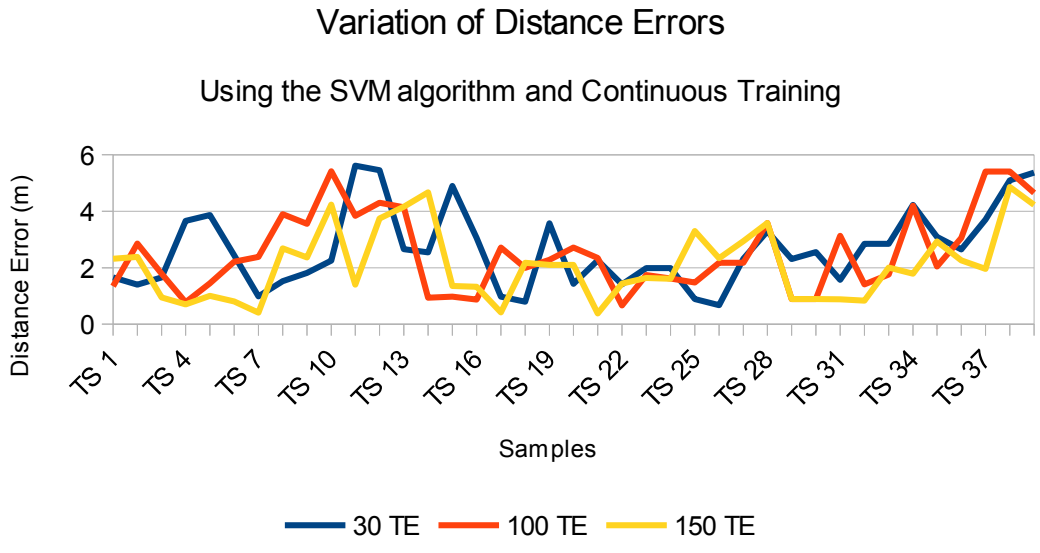

Figure 4.11: Variation of DE in the 3 tests done in the MR scenario using CT.

Figure 4.11 shows the variation of Distance Errors in the Meeting Room scenario using Continuous Training. As it can be observed, all the tests done contain big differences between their Maximum Distance Errors and Minimum Distance Errors, which means that although the

Mean Variance of DE and Mean Standard Deviation of DE were stable when compared with each others, it was still high.

Table 4.28: MDE using SVM in the Meeting Room Scenario (CT).

	Number of Training Examples		
	30	**100**	**150**
Mean Distance Error (m)	2.65	2.54	2.1
Mean Distance Error Change from the Previous test (%)	0	4.15 %	17.32 %

As Table 4.28 shows, the MDE decreases as the number of TE increases in this case. It starts at 2.65m in the 30 TE, going to 2.54m in the 100 TE test and finishing in 2.1m in the 150 TE one. This is an indicator that the Continuous Training in the Meeting Room scenario performed better than the Static Training approach in terms of position inference accuracy. Still, it was necessary to correlate the data of the different tests done in order to prove/disprove the hypothesis that this change really exists or if the data gathered was biased by external factors.

Table 4.29: WSR test for SVM between the Meeting Room scenario tests (CT).

	Training Examples Correlations		
	30 – 100	**30 - 150**	**100 - 150**
Number of positive differences	21	25	24
Number of negative differences	18	14	15
T	171	105	78
z	-0.31	-0.4	-0.44

These correlations were done through the Wilcoxon Signed Rank test. The results output by the WSR test are organized in Table 4.29, where it can be observed that none of the z values is lower than 1.96 or higher than 1.96, meaning the null hypothesis is confirmed for all the correlations, which indicates that there was no significant change between tests when $\alpha = 0.05$.

Because Continuous Training does not improve the accuracy of the position inference by adding Training Examples in the Meeting Room scenario, it was decided to remove it from the tests in the TiZ Entrance Hall scenario, a much bigger and wider one, where this training configuration would not improve the results also, like it didn't in the Meeting Room one.

The step after was to test the Static Training approach in the TiZ Entrance Hall scenario with the goal of evaluating how the algorithm behaved in a bigger area to infer positions and to compare its results with the ones of the other implemented algorithms. The test structure followed the same one that was used in all the previous tests.

Table 4.30: MDE using SVM in the TiZ Entrance Hall Scenario (ST – 3 MPs).

Number of Training Examples

	30	**100**	**150**
Mean Distance Error (m)	5.52	3.96	4,56
Mean Distance Error Change from the Previous test (%)	-	28.26 %	-15.15 %

The Mean Distance Error obtained by the test done using 3 Measured Positions and the SVM algorithm in the TiZ Entrance Hall scenario using Static Training as training configuration are organized in Table 4.30. As it can be seen, the value of MDE decreased significantly in the 100 TE test when comparing with the 30 TE one and increased again in the 150 TE test when comparing with the 100 TE one. In order to verify the correlations between the different tests, the Wilcoxon Signed Rank test was done.

Table 4.31: WSR test for SVM between the TiZ Entrance Hall Scenario tests (ST – 3 MPs).

Training Examples Correlations

	30 – 100	**30 - 150**	**100 - 150**
Number of positive differences	9	10	5
Number of negative differences	6	5	10
T	21	15	65
z	-2.21	-.2.56	0.28

Table 4.31 shows the output values of the WSR test. As it can be seen by the row of the values of z, the null hypothesis of the correlations 30-100 and 30-150 was rejected, since the values of z are below -1.96 and the research hypothesis in the correlation 100-150 was rejected, since the value of z is higher than -1.96 and lower than 1.96, when $\alpha = 0.05$. This means that between the 30 TE test and the 100 TE test and between the 30 TE test and the 150 TE test there was an improvement of the accuracy of the SVM algorithm, due to the fact that the number of positive differences is higher than the number of negative ones in both the correlations.

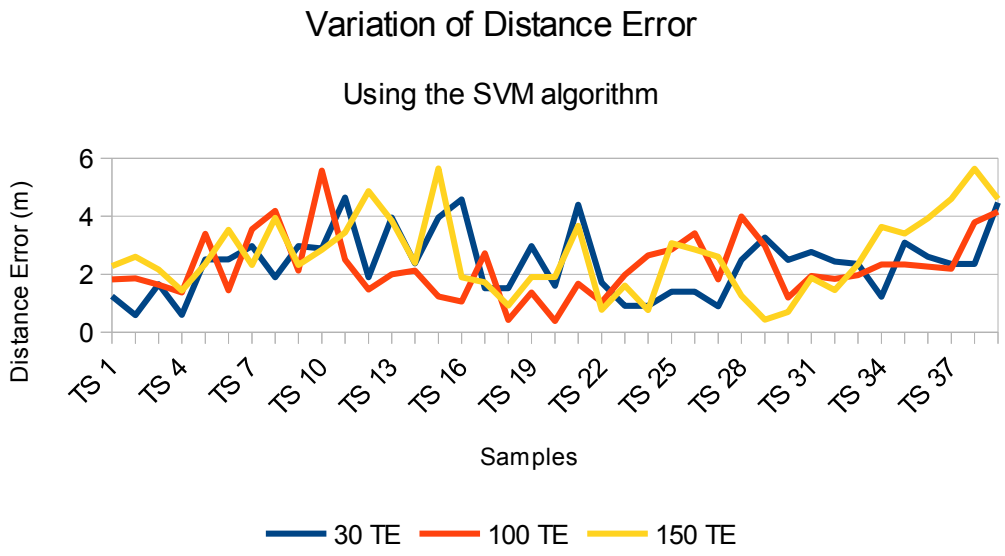

Figure 4.12: Variation of DE using SVM in the TiZ Entrance Hall and ST (3 MPs).

Figure 4.12 shows the Distance Errors per each of the training samples gathered in the TiZ Entrance Hall scenario using Static Training. As it can be seen, the variation of Distance Errors obtained is big for the 3 tests done, being more marked in the 30 TE test (with more highs and lows). The 100 TE test was the one with the DE are more equilibrated and the 150 TE test also possessed big variations, specially in the last inferences done where the DE grows almost consistently. In order to understand better these variations in the Distance Errors of the samples, a statistical analysis that evaluates several metrics was done.

Table 4.32: Statistical metrics of SVM in the TiZ Entrance Hall Scenario (ST – 3 MPs).

	Number of Training Examples		
	30	**100**	**150**
Mean Variance of DE (m)	15.67	2.96	6,48
Mean Standard Deviation of DE (m)	3.95	1.72	2,54
Mean Bias Inferred x (m)	3.25	1.21	2,34
Mean Bias Inferred y (m)	2.52	1.93	1,05
Max DE (m)	13.82	6.14	11,14
Min DE (m)	0	1.98	1,39
Hamming Loss (%)	86.67%	100%	80%
Exact Match Ratio (%)	6.67%	0%	0%

Although the WSR test confirms improvement in the accuracy of the technique if instead of 30 TE the algorithms use 100 or 150 TE, the calculated statistical metrics organized in Table 4.32 demonstrate an abnormal amount of Mean Variance of DE, Mean Standard

Deviation of DE and Mean Bias for both x and y, both in the 30 TE test and in the 150 one. In the case of the 100 TE test, the Mean Variance of DE is still high but a bit more acceptable than in the other cases. Still, the Hamming Loss is 100% in the 100 TE case, which means all the inferred values were more than 2 meters of distance away from the real positions of the mobile device.

Due to the fact that the MDE obtained in the tests done using 3 MPs in the TiZ Entrance Hall scenario are worse than the ones obtained without usage of any ML techniques, no more tests were done using this algorithm.

4.4.3 Using k-Means Clustering

The tests done using the k-Means Clustering approach followed exactly the same structure as the ones using the ANN and SVM algorithms. It was started by doing tests in the Meeting Room scenario, then it was evaluated the produced results and only after tests in the TiZ Entrance Hall scenario were done. The tests exclusion criteria are the same as the ones used for the tests of the other 2 implemented algorithms.

Table 4.33: MDE using k-Means Clustering in the MR Scenario (ST).

	Number of Training Examples		
	30	**100**	**150**
Mean Distance Error (m)	2.43	1.37	1.52
Mean Distance Error Change from the Previous test (%)	-	43,62 %	-10,94 %

The first testing scenario used was the Meeting Room of Latitude N's office. As usual, 3 tests with 3 different Measured Positions using Static Training were done. The Mean Distance Errors obtained can be seen in Table 4.33, where the 30 TE test had a 2.43m MDE, the 100 TE test had a 1.37m MDE and the 150 TE test had a 1.52m MDE. The accuracy of the position inference increase significantly from the 30 TE test to the 100 TE test – in 43.62 % - and decreased slightly from the 100 TE test to the 150 TE one – in 10.94%. Still, before comparisons were done, it was necessary to evaluate if the change in the populations of Distance Errors obtained actually existed or not. In order to do this, correlations between the results of the tests were done using the Wilcoxon Signed Rank test.

Table 4.34: WSR test for k-Means Clustering between the MR scenario tests (ST).

Training Examples Correlations

	30 – 100	30 - 150	100 - 150
Number of positive differences	30	29	19
Number of negative differences	9	10	20
T	45	55	570
z	-4.81	-4.67	2.51

Table 4.34 has the outputs of the Wilcoxon Signed Rank tests done between the different Training Examples populations of Distance Errors. It can be observed that all the correlations possess values of z either below -1.96 and above 2.51, which means that for all of them the null hypothesis was rejected and the research hypothesis was confirmed. In the 30 – 100 Training Examples and in the 30 - 150 correlations, the number of positive differences exceeds the number of negative differences, which means that there was an improvement in the accuracy of the position inference by adding Training Examples in these two cases. In the 100-150 TE correlation, there were more negative differences than positive ones, which means that there was a negative correlation between the 2 tests, even though it was small since the number of negative differences exceeds by only 1 the number of positive ones.

Table 4.35: Statistical metrics of k-Means Clustering in the MR Scenario (ST).

	Number of Training Examples		
	30	**100**	**150**
Mean Variance of DE (m)	1.23	0.68	0.68
Mean Standard Deviation of DE (m)	1.11	0.82	0.82
Mean Bias Inferred x (m)	0	0.28	0
Mean Bias Inferred y (m)	1.29	0.05	0.35
Max DE (m)	4.58	4.52	4.4
Min DE (m)	0.64	0.31	0.24
Hamming Loss (%)	92.31%	58.97 %	69.23%
Exact Match Ratio (%)	0 %	0 %	0 %

Table 4.35 presents the statistics obtained by the usage of the k-Means Clustering algorithm in the Meeting Room scenario using Static Training. As it can be seen, the Mean Variance of the DE and the Mean Standard Deviation of the DE decreases from the 30 TE test to the 100 TE test and is maintained from the 100 TE test to the 150 TE one. The values of the Mean Bias of Inferred x are always close to 0, and it can be seen that the test where the MDE was smaller is the one that possesses the Mean Bias of x that is 0.28 – the 100 TE test. The Mean Bias of Inferred y is quite high in the 30 TE test comparatively to the Mean Bias of Inferred y of the 100 TE and the 150 TE tests, that are close to 0m. The Maximum DE decreases

slightly as the number of Training Examples increased from test to test and the same happened with the Minimum DE.

The Hamming Loss is of 92.63% in the 30 TE test, which reflects a very high number of position inferences with DE above 1m. Comparatively to the 30 TE test, the Hamming Loss decreases significantly in the 100 TE test with a value of 58.67% and it increases lightly again in the 150 TE test to a value of 69.23%. The Exact Match Ratio is 0 % in all the 3 tests done, demonstrating that there were no position inferences done with full accuracy.

The tests done after the Static Training ones in the Meeting Room scenario were the same kind of tests but then using the Continuous Training configuration. The goal was of comparing the 2 approaches results using the same data set and the same training and testing conditions.

Table 4.36: MDE using k-Means Clustering in the MR Scenario (CT).

	Number of Training Examples		
	30	**100**	**150**
Mean Distance Error (m)	1.74	1.99	1.85
Mean Distance Error Change from the Previous test (%)	0 %	-14.38 %	7.04 %

Table 4.36 present the obtained MDE for the 3 tests done using Continuous Training in the Meeting Room scenario. The first test, using 30 TE, obtained a MDE of 1.74m; the second test, using 100 TE, obtained a MDE of 1.99m; the third, using 150 TE, obtained a MDE of 1.85m. Although apparently there were significant changes in the population of DE obtained by each one of the tests, that resulted in different Mean Distance Errors, the populations of data were submitted to the Wilcoxon Signed Rank test to confirm it.

Table 4.37: WSR test for k-Means Clustering between the MR scenario tests (CT).

	Training Examples Correlations		
	30 – 100	**30 - 150**	**100 - 150**
Number of positive differences	19	18	23
Number of negative differences	20	21	16
T	570	549	136

z	2.51	2.21	-3.54

Table 4.37 presents the output values of the WSR test. As it can be observed, the values of z are all below -1.96 or above 1.96, using $\alpha = 0.05$, which means that there was change between the populations correlated. In the first case, the correlation between the 30 TE test and the 100 TE test, show that the number of negative differences is higher than the number of positive ones, meaning that from the 30 TE test to the 100 TE one there was deterioration of the accuracy of the positioning inferences. The same happened in the second correlation, from the 30 TE to the 150 TE tests, confirming the MDE data available in Table 4.36. The third correlation is the only one of the 3 that shows improvement in the accuracy of the positioning inferences, from the 100 TE test to the 150 TE one, since it has more positive differences than negative ones. This means that, using this configuration in a small scenario like the Meeting Room scenario, increasing the number of Training Examples from 100 to 150 will produce better results than any other change. Still, the test that presents the lower MDE is the 30 TE one, as it was stated previously.

Table 4.38: Statistical metrics of k-Means Clustering in the MR Scenario (CT).

	Number of Training Examples		
	30	**100**	**150**
Mean Variance of DE (m)	1.7	1.33	1.85
Mean Standard Deviation of DE (m)	1.3	1.15	1.18
Mean Bias Inferred x (m)	0.2	0.42	0.4
Mean Bias Inferred y (m)	0.08	0.36	0.31
Max DE (m)	4.99	4.62	6.08
Min DE (m)	0	0	0.16
Hamming Loss (%)	71.79 %	74.36 %	76.92 %
Exact Match Ratio (%)	5.13 %	2.56 %	0 %

Table 4.38 contains the statistical metrics obtained by the populations of the 3 tests done in the Meeting Room scenario using Continuous Training. As it can be seen, the Mean Variance of DE was lower in the 100 TE test with a value of 1.33m, when in the 30 TE and 150 TE cases the values are of 1.7 and 1.85m, respectively. The Standard Deviation of DE is similar in the 3 tests and such is the Mean Bias of the Inferred x. The Mean Bias of the Inferred y is close to 0 in the 30 TE test while it's a bit higher in the 100 and 150 Training Examples cases (0.36m and 0.31m, respectively). This means that in these 2 last tests, the variety of positions inferred is bigger. The Maximum DE is of 4.99m in the 30 TE case, lightly lower on the 100 TE test – with a value of 4.62m – and reaches the highest of the 3 tests in the 150 TE case, with a value of 6.08m. The Minimum DE is of 0m in both the 30 TE and the 100 TE cases and of

0.16m in the 150 TE test. Relatively to ML statistical metrics, the Hamming Loss is quite similar in the 3 tests, as it can be observed. The Exact Match Ratio is of 5.13% in the 30 TE test, of 2.56% in the 100 TE test and of 0% in the 150 TE case.

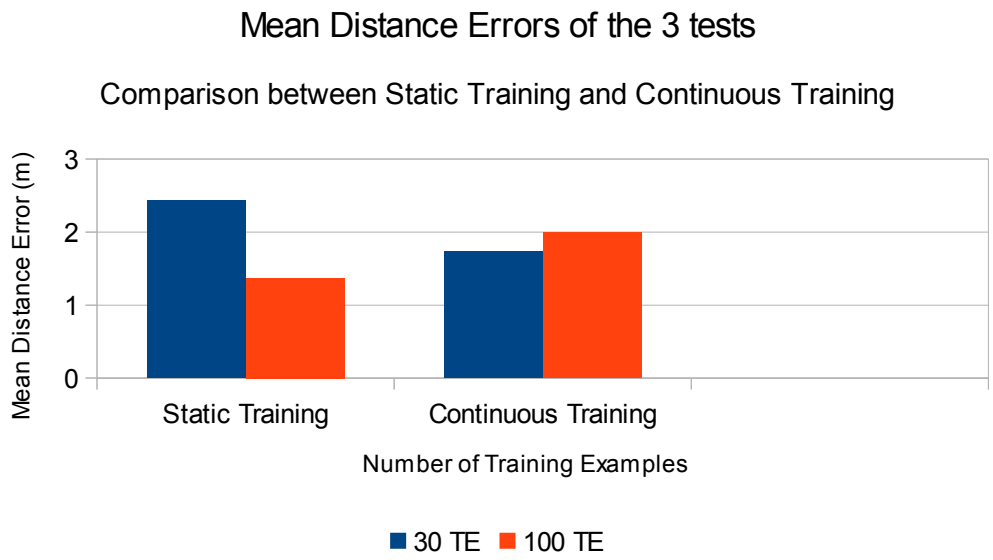

Figure 4.13: Comparison between the MDE of the 3 tests using ST and CT.

As it happened in the tests done using the SVM algorithm, the Continuous Training approach behaves worse than Static Training. Figure 4.13 compares the MDE obtained by the 2 different training configurations for the 3 different tests. The only time Continuous Training obtained a lower MDE than the Static Training configuration was in the 30 TE case. In the other 2 tests done, Static Training performs better in terms of algorithm accuracy than the Continuous Training one. This conclusion allowed for the Continuous Training approach not to be used in the tests in the TiZ Entrance Hall scenario, since it meets one of the tests' exclusion criteria.

Table 4.39: MDE using k-Means Clustering in the TiZ Entrance Scenario (ST – 3 MPs).

	Number of Training Examples		
	30	**100**	**150**
Mean Distance Error (m)	2.93	2.2	1.9
Mean Distance Error Change from the Previous test (%)	-	24.89 %	13.96 %

The next step was to test the k-Means Clustering approach in the TiZ Entrance Hall scenario using 3 MPs. Table 4.39 presents the obtained Mean Distance Errors obtained for each

one of the different tests using Static Training. The MDE decreased as the number of TE increased, as it can be observed. The next step was to correlate the data to confirm this improvement in the accuracy of the technique as the number of TE rose. To do that, the Wilcoxon Signed Rank test was used, once again.

Table 4.40: WSR test for k-Means Clustering between the TiZ Entrance Hall (ST – 3 MPs).

Training Examples Correlations

	30 – 100	**30 - 150**	**100 - 150**
Number of positive differences	11	11	9
Number of negative differences	4	4	6
T	10	10	21
z	-2.84	-2.84	-2.22

Table 4.40 presents the output data from the Wilcoxon Signed Rank test that correlates the populations of the tests done with different TE in the TiZ Entrance Scenario using Static Training and 3 different Measured Positions. Due to the fact that all the values of z are either above 1.96 or below -1.96, it means that in all the 3 shown correlations the null hypothesis was rejected and the research hypothesis was accepted, meaning that there was change as the number of TE rose. That change is positive in all the 3 correlations, since the number of positive differences is higher than the number of negative differences. This means that from 30 TE to 100 TE, from 30 TE to 150 TE and from 100 TE to 150 TE, the accuracy of the position inference is enhanced.

Table 4.41: Statistical metrics of k-Means Clustering in the TiZ Entrance Hall (ST – 3 MPs).

Number of Training Examples

	30	**100**	**150**
Mean Variance of DE (m)	2.68	1.2	0.73
Mean Standard Deviation of DE (m)	1.64	1.10	0.85
Mean Bias Inferred x (m)	1.16	0.63	0.86
Mean Bias Inferred y (m)	0.6	0.21	0.21
Max DE (m)	6.12	4.49	3.08
Min DE (m)	0.75	0.7	0.32

Hamming Loss (%)	66.67%	53.33%	40 %
Exact Match Ratio (%)	0 %	0 %	0 %

Ahead, it was generated the statistics that correspond to each one of the 3 tests done using the k-Means algorithm. Table 4.41 contains the values obtained. As it can be observed, the Mean Variance of DE and the Mean Standard Deviation of DE, decrease as the number of TE increase. They were actually quite high in the 30 TE test, but from the 100 TE test on it reaches common values. The Mean Bias of the Inferred x and y are also higher in the 30 TE test and stabilize in the 100 and 150 TE tests. The Maximum DE and Minimum DE decrease also as the number of TE increases, achieving an unseen low value for the Minimum DE in this testing scenario of 3.08m in the 150 TE test.

The Hamming Loss decreases also as the number of TE increase, hitting 40% in the 150 TE test, which means that only 40% of the data gathered had DE higher than 2m. The Exact Match Ratio had 0% for all the 3 tests, meaning that no entries were gathered where the accuracy in the position inference was total.

Since the Mean Distance Errors obtained by the k-Means Clustering approach were so low and were lower than the ones of the technique that does not use any AI, it was decided to repeat the test in the same scenario but then with 6 MPs, in order to study the behavior of the algorithm with this new configuration and compare it with the one with 3 MPs.

Table 4.42: MDE using k-Means Clustering in the TiZ Entrance Hall (ST– 6 MPs).

	Number of Training Examples		
	30	**100**	**150**
Mean Distance Error (m)	3.24	3.23	2.7
Mean Distance Error Change from the Previous test (%)	0 %	0.3 %	16.4 %

Table 4.42 shows the MDE for each of the tests done using 6 MPs. As it can be observed, the MDE decreases as the number of TE increase, starting from an MDE of 3.24m in the 30 TE test, decreasing very slightly to 3.23m in the 100 TE test and achieving the value of 2.7m in the 150 TE test. Hence, this configuration shows the same behavior as the one with 3 MPs, although the Mean Distance Errors are higher in this configuration. In order to correlate the populations of Distance Errors obtained and verify if there was significant change between them, the Wilcoxon Signed Rank test was used.

Table 4.43: WSR test for k-Means Clustering between the TiZ Entrance Hall (ST – 6 MPs).

Training Examples Correlations

	30 – 100	**30 - 150**	**100 - 150**
Number of positive differences	15	16	19
Number of negative differences	15	14	11
T	345	105	66
z	2.31	-2.62	-3.42

Table 4.43 contains the output values of the WSR test. The z value for all the correlations is either above 1.96 or below -1.96, which means that in all the correlations the null hypothesis was rejected and the research hypothesis accepted, hence there were significant changes between them. The change is positive in the 30-150 TE tests correlation and in the 100-150 TE tests one, since the number of positive differences is higher than the number of negative ones. In the 30-100 TE tests correlation, the number of positive differences is equal to the number of negative ones, meaning that even with the null hypothesis being rejected, the change is not significant.

Table 4.44: Statistical metrics of k-Means Clustering in the TiZ Entrance Hall (ST – 6 MPs).

Number of Training Examples

	30	**100**	**150**
Mean Variance of DE (m)	4.74	6.17	4
Mean Standard Deviation of DE (m)	2.18	2.48	2
Mean Bias Inferred x (m)	0.58	0.44	0.31
Mean Bias Inferred y (m)	0.98	0.1	0.4
Maximum DE (m)	8.22	8.44	10.17
Minimum DE (m)	0.01	0.2	0.37
Hamming Loss (%)	73.33%	63.33 %	63.33 %
Exact Match Ratio (%)	0 %	0 %	0 %

As Table 4.44 shows, the Mean Variance of DE and the Mean Standard Deviation of DE are quite high. This allows the conclusion that as the number of MPs rise, the chaos in the current model rises too, allowing for higher Maximum DE and really low Minimum DE. The

Mean Inferred x is similar in the 3 tests done and the Mean Inferred y is higher in the 30 TE example, lowering considerably in the 100 TE example and achieving an expected bias value in the 150 TE case. Relatively to the ML statistics, the Hamming Loss assumed a value of 73.33% in the 30 TE test and of 63.33% in the 100 and 150 TE tests. The Exact Match Ratio is of 0% in all the 3 cases.

Table 4.45: Comparison of MDE of the k-Means Clustering in the TiZ Entrance Hall (3-6 MPs).

Number of Training Entries

	30	**100**	**150**
MDE of 3 MPs (m)	2.93	2.2	1.9
MDE of 6 MPs (m)	3.24	3.23	2.7

Although as the number of Training Examples increases and the Mean Distance Error decreases using 6 Measured Positions, the MDE obtained by the 3 tests done in the current test were worse than the ones obtained in the 3 MPs tests. Henceforth, this was the last test done in the TiZ Entrance Hall scenario, due to the fact that the data gathered until that moment was enough to prove/disprove the hypothesis formulated initially and properly justify them. Table 4.46 contains the MDE of all the tests done both with 3 and 6 MPs using Static Training.

4.5 Intelligent Position Inference vs. Common Position Inference

Due to the fact that the amount of tests done is quite high and the goal of this sub-chapter is to compare the ML approaches with the data gathered in the same environments and with the same configurations, only the tests with the best scores for each scenario will be submit to comparison with its equivalent No-ML data.

In the majority of the comparisons that will be established throughout this sub-chapter, the amount of data of the No-ML approaches were submit to a reduction of the amount of samples to be used. This was done because the number of examples gathered using the implemented ML algorithms was usually less than the number of samples gathered by the No-ML approach. However, the selection of the No-ML samples used for comparison was done carefully, taking into consideration the fact that they had to correspond to the same real positions and in order that the Mean Distance Error obtained did not change significantly.

For each of the algorithms, comparisons will be made towards the 2 different testing scenarios (the Meeting Room and the TiZ Entrance Hall, respectively). Also, the statistical metrics will all be presented and compared, in order to understand better which were the major differences between the populations generated by each of the algorithms.

4.5.1 ANN vs No-ML

The configuration that achieved the lowest MDE in the Meeting Room scenario using Artificial Neural Networks to infer position was the one of 150 Training Examples using Static Training. Hence, for the Meeting Room scenario, the values used for comparison with the approach that does not use any ML algorithm to infer position will be the ones of the specific test. But first of all, it's necessary to evaluate if there was significant change between the two populations to be compared and if there was, to evaluate if that change is positive or negative. The Wilcoxon Signed Rank test was used for that purpose and it output z with the value of -4.35 and with 27 out of 39 positive differences. This means that when comparing the No-ML technique and the ANN approach with this configuration, there was an improvement in the accuracy of the positioning inference.

Table 4.46: Results of the best ANN test and the No-ML approach in the MR.

	No-ML	**ANN**	**Change (%)**
Mean Distance Error (m)	2.2	1.23	-83.3
Mean Variance of DE (m)	2.03	0.56	-262.5
Mean Standard Deviation (m)	1.42	0.75	-89.3
Mean Bias of Inferred x (m)	0.24	0.12	-100
Mean Bias of Inferred y (m)	0.53	0.3	-73,3
Maximum DE (m)	6.19	2.95	-109,8
Minimum DE (m)	0.1	0.13	23.07
Hamming Loss (%)	-	59	-
Exact Match Ratio (%)	-	0	-

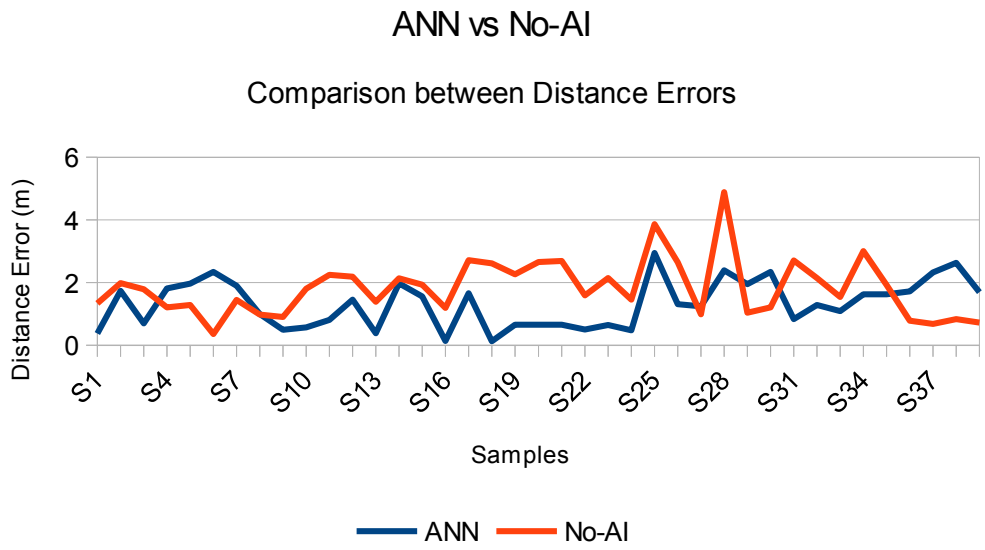

Figure 4.14: Best MDE of ANN in the MR scenario in comparison with No-AI.

As Table 4.46 shows, the Mean Distance Error of the No-ML approach using 150 TE and Static Training in the Meeting Room scenario was of 2.2m, while the MDE of the ANN approach with the same configuration was of 1.23m, a cutback of 83.3%. The Mean Variance of DE is 262.5 % higher than in the ANN approach – 2.03m against 0.56m. The Mean Standard Deviation of DE 1.42m and the one of ANN is of 0.75m, a reduction of 89.3 %. The Mean Bias of Inferred x is of 0.24m and the one of the ANN approach is of 0.12m, which represents a retrenchment of 100%. In the case of the Bias of the Inferred x, although the one of the ANN approach is lower, the values don't make that much numerical difference, which ends up being not so much significant. With the Mean Bias of Inferred y the same thing happens, with a cutback of 73.3% although it's not very significant since both values are pretty similar. The Maximum DE was reduced from 6.19m to 2.95m, representing a shrinkage of 109.8 %. The Minimum DE was the only statistical metric that rose, even though the difference is of only 0.03m. Figure 4.14 shows the Distance Errors per samples that correspond to the same real positions. It can be noticed that there is a clear gap between the No-AI and the ANN approaches, which indicates a big difference between the amount of DE produce by each one of the methods.

In the TiZ Entrance Hall scenario, the test that possessed the lowest MDE was the one of 150 TE using 3 MPs. Hence, the statistics produced by that test will be the ones compared with the ones from the No-AI population statistics. As usual, the first step is to verify if there was change between the 2 populations of DE to be compared posteriorly. It was used the Wilcoxon Signed Rank test to check that. The test output z with the value of -2.83 and 11 positive differences out of 15. This means that there was an improvement in the accuracy of the positioning inference technique, using the same testing conditions as the ones in the 150 TE using 3 MPs test.

Table 4.47: Results of the best ANN test and the No-AI in the TiZ Entrance Hall.

	No-ML	ANN	Change (%)
Mean Distance Error (m)	3.45	1.43	-141.26
Mean Variance of DE (m)	1.48	1.41	-4.9
Mean Standard Deviation (m)	1.22	1.19	-2.52
Mean Bias of Inferred x (m)	0.42	0.57	14.04
Mean Bias of Inferred y (m)	0.56	0.18	-211.1
Maximum DE (m)	6.48	3.65	-77,53
Minimum DE (m)	0.67	0.37	-81.08
Hamming Loss (%)	-	33.3	-
Exact Match Ratio (%)	-	0	-

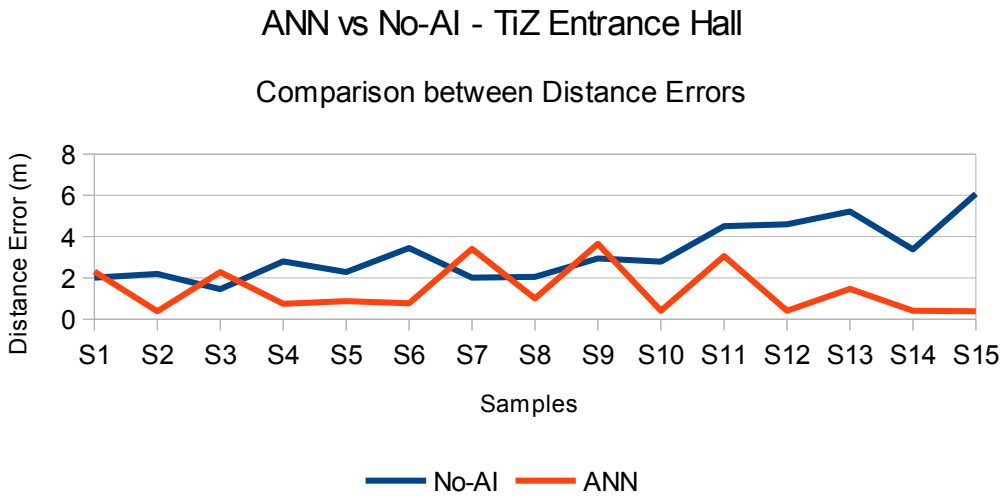

Figure 4.15: Best MDE of ANN in the TiZ Entrance Hall in comparison with No-AI.

Table 4.47 contains the descriptive statistical metrics of the ANN approach with 150 TE using 3 MPs in the TiZ Entrance Hall scenario in direct comparison with the same approach using No-ML to infer positions. As it can be seen, there was a decrease of 141.26 % in the Mean Distance Error, from 3.45m in the No-ML approach to 1.43m using the ANN technique. In terms of Mean Variance of DE, the values are quite similar, differing both techniques by 4.9%, which is not a significant change. The Mean Standard Deviation change between approaches is even lower, with a reduction of 2.52% from the first to the second. The Mean Bias of Inferred x increase 14.04%, from the value of 0.42m in the No-ML approach to 0.57m of the ANN algorithm. The Mean Bias of Inferred y decreased 211.1%, although numerically the differences are not severe – from 0.56m to 0.18m, in the No-ML and the ANN approaches, respectively. The Maximum DE reduced from 6.48m in the No-ML technique to 3.65m using the ANN algorithm, which is reflected also in the Mean Distance Error values. The Minimum

DE suffered a cutback of 81.08 %, from 0.67m to 0.37, although the change is not so significant due to low numerical difference and no reflection in any of the other metrics or on the Mean Distance Error. Figure 4.15 confirm the huge improvement in terms of accuracy introduced by the ANN algorithm when comparing to the No-AI approach, where it's noticeable a gap between the evolution of Distance Errors through the generated data during the tests.

4.5.2 SVM vs No-ML

The configuration that obtained the best results when using the Support Vector Machines approach in the Meeting Room scenario was the one with 150 TE and Continuous Training. Hence, the comparisons for the Meeting Room scenario will be done using the data from the position inferences using that configuration and the data from the respective population using no ML technique. First of all, it was necessary to compare the 2 populations in order to verify if there was some significant change between them. To do that, the Wilcoxon Signed Rank test was used and the output value of z was of -2.79, which is lower than -1.96, meaning that there is an improvement in the accuracy of the position inference, since the number of positive differences was higher than number of negative ones.

Table 4.48: Results of the best SVM test and the No-ML approach in the MR.

	No-ML	**SVM**	**Change (%)**
Mean Distance Error (m)	3.45	2.1	-64.29
Mean Variance of DE (m)	1.48	1.49	0,67
Mean Standard Deviation (m)	1.22	1.22	0
Mean Bias of Inferred x (m)	0.42	0.35	-20
Mean Bias of Inferred y (m)	0.56	0.99	-43.43
Maximum DE (m)	6.48	4.86	-33.33
Minimum DE (m)	0.67	0.83	19.28
Hamming Loss (%)	-	74.36	-
Exact Match Ratio (%)	-	0	-

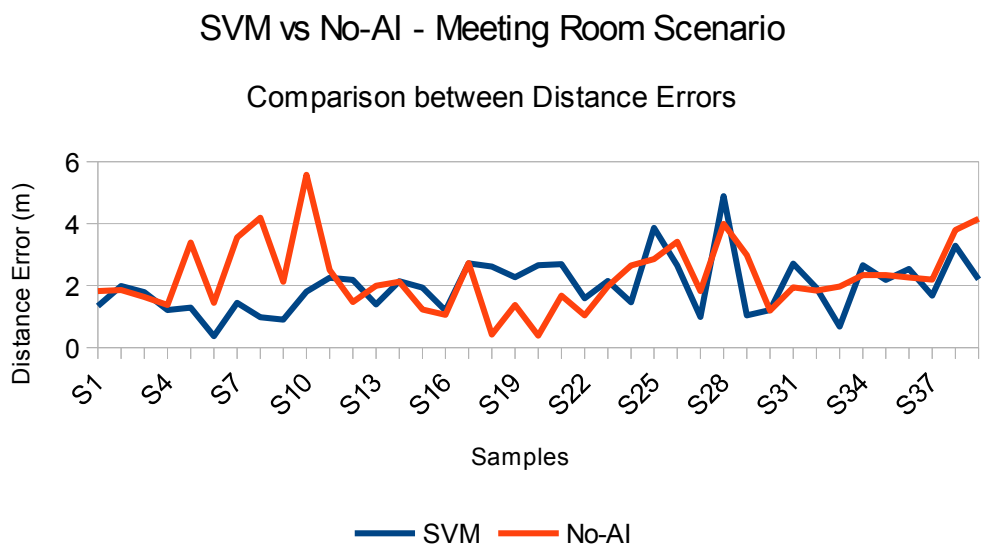

Figure 4.16: Best MDE of SVM in the Meeting Room scenario in comparison with No-AI.

According to Table 4.48, Mean Distance Error was reduced in 64.29% from the No-ML approach to the one using SVM – from 3.45m to 2.1m. Both the Mean Variance of DE and the Mean Standard Deviation of DE have almost the same values, demonstrating that the differences in the distributions of data are almost none. The Mean Bias of Inferred x has a difference of 20% from the No-ML approach to the SVM algorithm and the Mean Bias of Inferred y decreased 43.33% between the first and the second techniques. The Maximum DE is 33.33% lower in the SVM algorithm approach and the Minimum DE is 19.28% higher than in the No-ML technique, although the values are quite similar numerical, having no practical influence in the MDE of both the approaches.

Relatively to the TiZ Entrance Hall scenario, the tests done with the SVM algorithm demonstrate that the Mean Distance Error obtained by them is higher than the MDE obtained by the approach without ML. This means that the SVM algorithm failed also to improve the accuracy of the existent indoor positioning system in the TiZ Entrance Hall scenario.

4.5.3 Clustering vs No-ML

The setup that achieved the best results when using the k-Means Clustering technique in the Meeting Room scenario was the one with the 100 TE and Continuous Training. Henceforth, the comparisons for the Meeting Room scenario will be done using the data from the position inferences using that configuration and the data from the respective population using no ML technique. The first step was to verify if the changes between the 2 populations were significant or not. It was used the Wilcoxon Signed Rank test and the output value of z was of -4.67, which means the null hypothesis is rejected and the research hypothesis is accepted and because there were more positive differences than negative ones, it can be concluded there was an improvement in the accuracy of the position inference by adding the k-Means Clustering algorithm.

Table 4.49: Results of the best k-Means Clustering test and the No-AI approach in the MR.

	No-ML	**K-Means Clustering**	**Change (%)**
Mean Distance Error (m)	1.93	1.37	-40.88
Mean Variance of DE (m)	1.07	0.68	-57.35
Mean Standard Deviation (m)	1.04	0.82	-26.83
Mean Bias of Inferred x (m)	0.17	0.28	39.29
Mean Bias of Inferred y (m)	0.44	0.05	-780
Maximum DE (m)	4.89	4.52	-8,19
Minimum DE (m)	0.1	0.31	67,77

Hamming Loss (%)	-	58.97 %	-
Exact Match Ratio (%)	-	0 %	-

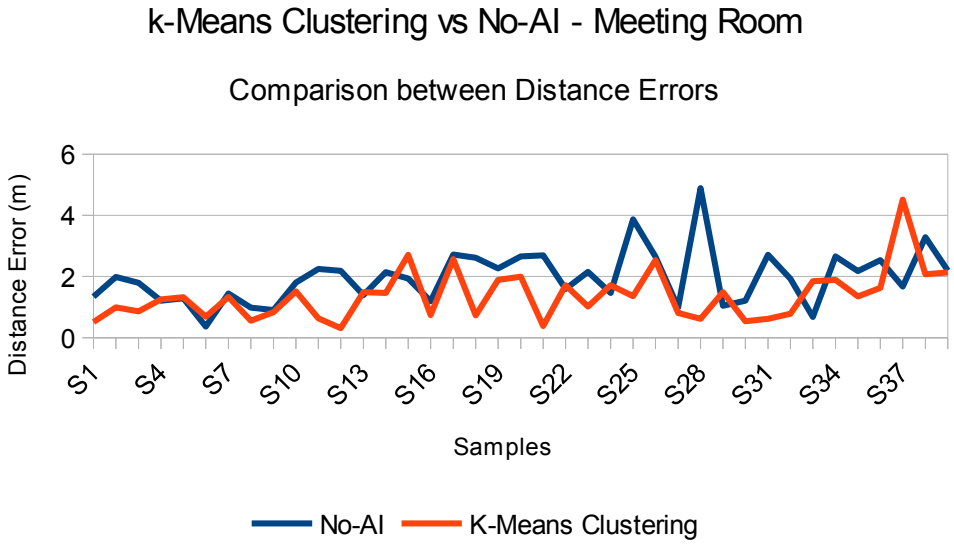

Figure 4.17: Best MDE of k-Means in the MR in comparison with No-AI.

Table 4.49 presents the results of the approach of the k-Means Clustering algorithm that obtained the best value of MDE – 100 TE using Continuous Training - and the results of the No-ML algorithm for the same configuration in the Meeting Room scenario. The MDE was reduce by 40.88% from the No-ML approach to the k-Means Clustering one, from the value of 1.97m to 1.37m, the Mean Variance of DE was cutback from the value of 1.07m in the technique that does not use ML to 0.68m of the k-Means Clustering – representing a curtailment of 57.35% and the Mean Standard Deviation of DE was retrenched in 26.83 % - from 1.04m to 0.82m in the No-ML technique and in the k-Means Clustering, respectively. All the other values are very similar numerically, although some of the changes were accentuated in terms of percentage.

With respect to the TiZ Entrance Hall scenario, the configuration in which the k-Means obtained the best results is the one with 150 Training Examples and Static Training using 3 Measured Positions for training. Before comparisons were established, it was necessary to correlate the populations of Distance Errors of both the approaches. To do that, it was use the Wilcoxon Signed Rank test with the goal of understanding if the change between the two populations of Distance Errors was significant or not. The WSR test output z with a value of -2.22, which allowed the rejection of the null hypothesis and the acceptation of the research hypothesis, meaning that there was change between the two populations of Distance Errors. That change is positive, since the number of positive differences is higher than the number of negative differences. This allowed the conclusion that the k-Means Clustering algorithm improved the accuracy of the positioning inferences in relationship to the no-ML technique.

Table 4.50: Results of the best k-Means and the No-ML in the TiZ Entrance Hall.

	No-ML	**K-Means Clustering**	**Change (%)**
Mean Distance Error (m)	3.45	1.9	-81.58
Mean Variance of DE (m)	1.48	0.73	-102.74
Mean Standard Deviation (m)	1.22	0.85	-43.53
Mean Bias of Inferred x (m)	0.42	0.86	51,16
Mean Bias of Inferred y (m)	0.56	0.21	-166.67
Maximum DE (m)	6.48	3.08	-110.39
Minimum DE (m)	0.67	0.32	-109.38
Hamming Loss (%)	-	0.4	-
Exact Match Ratio (%)	-	0	-

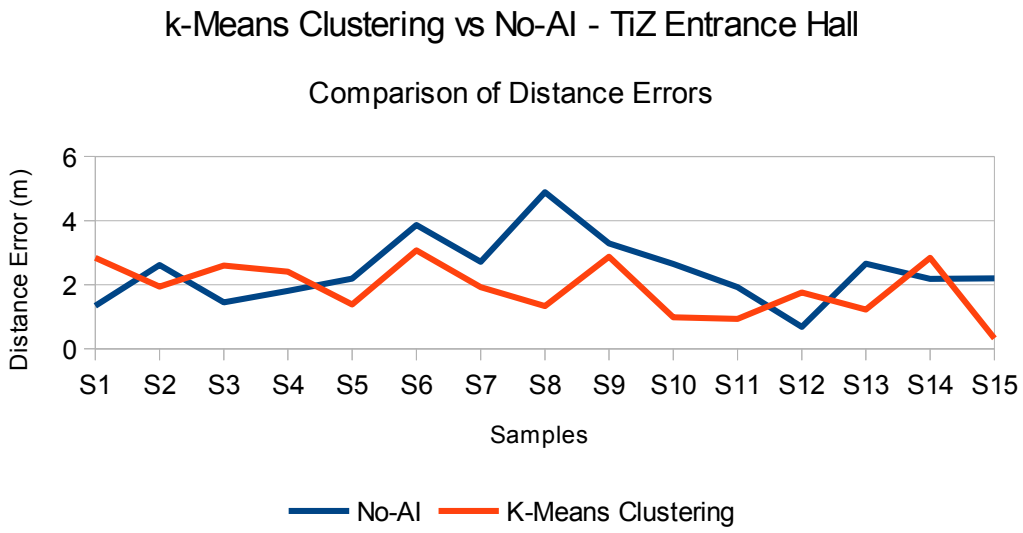

Figure 4.18: Best MDE of k-Means in the TiZ Entrance Hall in comparison with No-AI.

Table 4.50 presents the results obtained both by the No-ML algorithm in the 150 TE case in the TiZ Entrance Hall scenario and the k-Means Clustering algorithm using also 150 TE and Static Training. It was proven before that this ML approach increased the accuracy of the indoor positioning system in relationship to the No-ML technique, but the Table 4.50 and the Figure 4.18 demonstrates it more minutely. The Mean Distance Error decreased 81.58 %, from the value of 3.45m in the No-ML algorithm to the value of 1.9m in the k-Means Clustering approach. The Mean Variance of De decreased 102.74%, from 1.48m to 0.73m, while the value of the Mean Standard Deviation of DE decrease 43.53 % - from 1.22m to 0.85m. The Maximum DE and Minimum DE decrease both to a bit less than half of the values of the No-ML technique and although there were some changes in the Mean Bias both for x and y, they remained quite stable. Figure Error: Reference source not found represents the different Distance Errors obtained by each one of the samples of both populations. It's noticeable a gap between the No-AI Distance Errors and the k-Means Clustering algorithm, where the No-AI DE are almost in all the cases above the k-Means Clustering Distance Errors.

4.6 Related Works

The comparisons between the results obtained through the test of the 3 implemented Machine Learning algorithms in the 2 different scenarios and the ones expressed in the studied literature are of difficult achievement, mainly due to the fact that the equipment used is different, the training setups are not the same and the scenario configurations are not alike. Also, none of the papers tests their approaches in 2 different scenarios, which means that the comparisons per each of the algorithms implemented will be done only with one of the obtained results in relationship always with the most similar scenario in terms of dimensions or then with both of them but without the necessary rigor to validate them properly.

4.6.1 ANN

As it was stated before, the training configuration that presents the lower Mean Distance Error is the one with 150 Training Examples and using Static Training as training configuration. The MDE obtained is of 1.23m in the Meeting Room scenario. The best approach found in the Literature Review phase was [MTT10]. In this approach, the authors implemented a ANN with 4 Input Neurons and 2 Output Neurons, using a Sigmoid Activation function, 4 Hidden Layers and 5000 training iterations. The MDE obtained by the authors was of 1.43m in a 8x9 m scenario, which is bigger than the Meeting Room scenario used for testing in this Dissertation's case.

Still, if it's compared directly the Meeting Room scenario's obtained results with the ones obtained by the authors, in the Meeting Room scenario this implementation's best result is of 1.23m in an 3.12x6.25m scenario against 1.43m in a 8x9 scenario and of 1.44m in an 14x17.1m against, again, 1,43 in a 8x9 scenario. This means that, when compared to the literature results using ANN, the results that the ANN approach here implemented was able to obtain are quite positive and optimistic, even though direct comparisons are not possible due to the reasons stated before.

4.6.2 SVM

The article which was selected as the one that the testing scenario was the most similar with the ones tests during this Dissertation was [FAWJC12]. The authors developed and experimented several different kinds of Support Vector Machines and tested their approaches in a 49x19m (941 m^2 of total area) scenario with several different rooms and several obstacles and the comparisons were done between SVM approaches with different kinds of *kernelization* algorithms and it was proven that a-priori information can enhance the performance of positioning systems. The results are not displayed correctly in order to compare with the ones achieved with this implemented of the SVM algorithm, hence, direct comparisons can't be established with this approach.

4.6.3 k-Means Clustering

The article which was selected as the one which application and tests could be more useful to compare against was the [MLTL08]. The authors tested a variant of the kNN algorithm applied with Clustering that they developed themselves and they were able to show it would outperform the kNN approach. The dimensions of the scenario were not revealed in the paper and the results were shown as 50% of the errors obtained were in the interval [0m ; 1.2m] and the other 50% were in the interval [1.21m;2.2m].

The best results obtained in the approach this Dissertation developed of the k-Means Clustering algorithm, achieved a 1.37m MDE in the Meeting Room scenario and of 1.9m in the

TiZ Entrance Hall scenario. The population of DE obtained is far from being 50% in the interval [0m;1.2m] and the other 50% on the interval [1.21m;2.2m], hence it's possible that the algorithm the authors implemented easily outperforms the one that was implemented throughout this work, using the current training configuration.

4.7 Summary

This Chapter's goal were to enunciate the Hypothesis that needed to be tested, to describe the testing scenarios, to compare the implemented algorithms between each others, to analyze statistically their performances in different environments with different configurations and to compare the results obtained from the implemented approaches with the approach implemented by [C12] and with the ones studied and chosen as the ones which performed better in the Literature Review phase (Chapter 2).

This was all achieved throughout the current Chapter. The Hypothesis have been explained and all the tests done were in order to be able to respond to them. The statistical metrics used for comparison were also detailed and so were the ML intrinsic statistics for multi-class multi-label classification, that is the type of classification problem that is being dealt with in this Dissertation. The structure of the tests was also referred several times and so was the tests exclusion criteria. The results were shown, analyzed and compared, always oriented towards the answering to the Hypothesis established in the beginning of the current Chapter.

Several conclusions were taken throughout the Chapter, relatively to the algorithms' behavior in the different scenarios and using different configurations. From those conclusions taken it's easier to compare the implemented approaches and the statistical outcomes that they accomplish. Those conclusions will all be detailed in the next Chapter, where the initially established Hypothesis will be put to test using the data presented in the current Chapter.

5 Conclusions

5.1 Overall Analysis

The first phase of every study, dissertation or thesis, after defining the goals of it and the process of how to achieve it, is to review the existent literature related with the field of study and with the subjects that compose the theme. When that phase is over, it's necessary to define which kinds of variables are in need of an evaluation, in order to add value to the subject being studied. In order to do that, the right approach is to elaborate several Hypothesis that either haven't been answered specifically by the reviewed literature or that, by providing answers to them, value is added to the current field of study.

That is precisely what has been done before the implementation phase in this dissertation. Several hypothesis have been established and written down and all the efforts put in the testing phase are in order to answer them. Each one of those hypothesis should be answered easily and directly after the tests are done and the results analyzed, although to do it it may be needed to do more than just one test. This chapter is dedicated to the answering of those Hypothesis, justifying the answers with the data produced from the tests done and with comparisons already established in Chapter 4.

Hypothesis 1: *All of the implemented ML algorithms obtain a lower Mean Distance Error than the approach that does not use any Artificial Intelligence, in the same Measured Positions and in positions that were not used for training of the implemented approaches.*

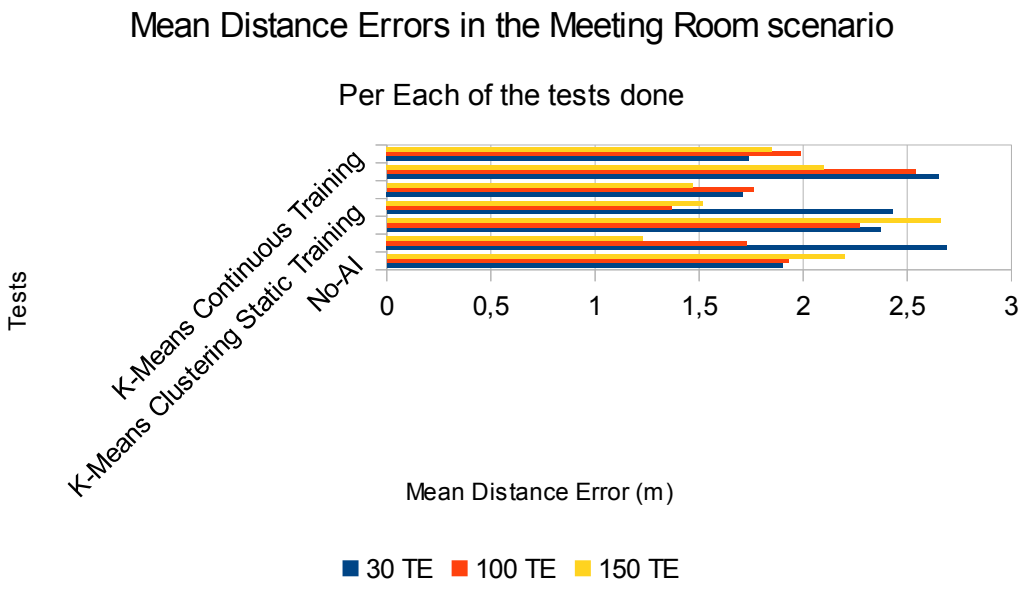

Figure 5.1: MDE in the Meeting Room scenario per tested approach.

As Figure 5.1 shows, from all the tests done in the Meeting Room scenario, the only ones that did not achieve a lower Mean Distance Error than the No-AI approach were the ANN Static Training 30 TE, the k-Means Clustering Static Training using 30 TE, the k-Means Continuous Training using 100 TE and the whole SVM tests. All the other performed tests showed a lower MDE than the No-AI approach.

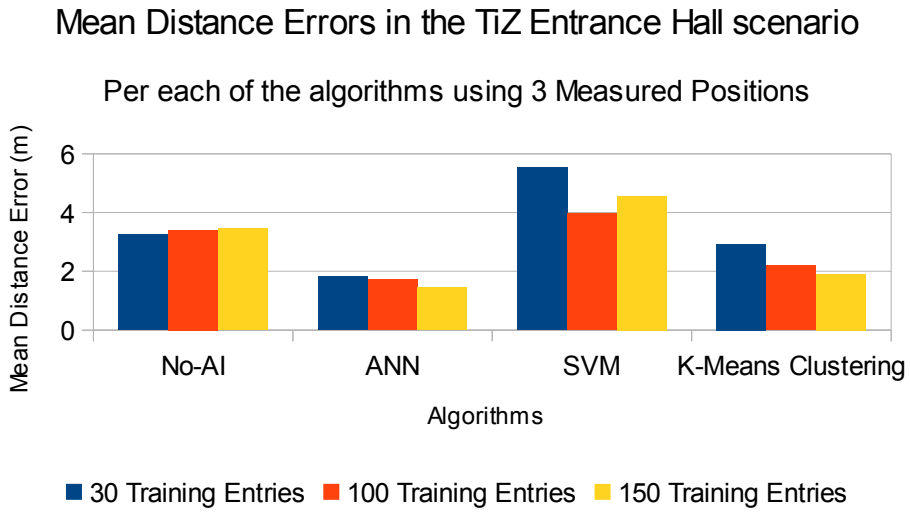

Figure 5.2: MDE in the TiZ Entrance Hall using 3 MP per approach tested.

Figure 5.2 demonstrates the MDE obtained in the TiZ Entrance Hall scenario using 3 Measured Positions for all the approaches tested. As it can be observed, the only approaches that performed worse than the No-AI approach, in these set of tests, were the ones using the SVM algorithm. All the approaches using the ANN and the k-Means Clustering algorithms obtained lower MDE than the No-AI approach in the same circumstances.

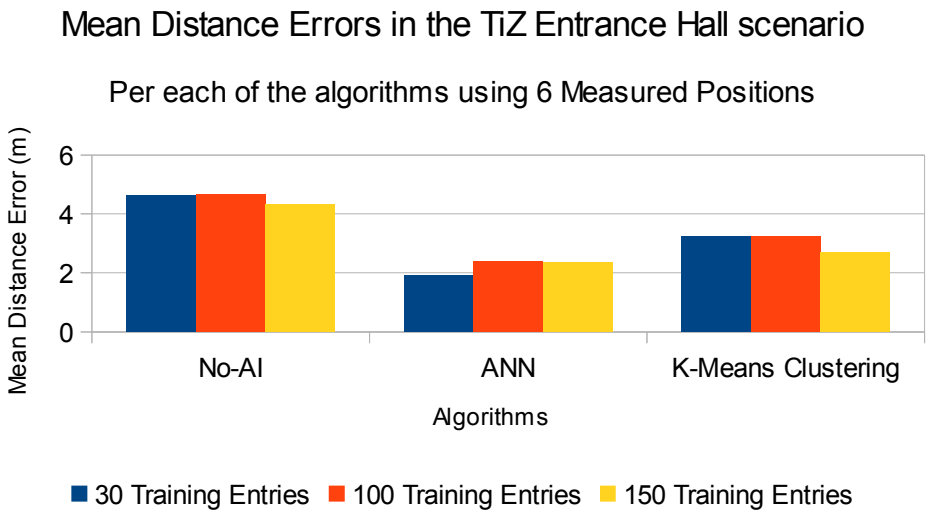

Figure 5.3: MDE in the TiZ Entrance Hall using 6 MP per approach tested.

Figure 5.3 shows the obtained MDE in the tests done in the TiZ Entrance Hall scenario using 6 Measured Positions per each approach tested. It confirms the fact that each test done, either with the ANN and the k-Means Clustering approaches, have lower Mean Distance Errors than the No-AI approach.

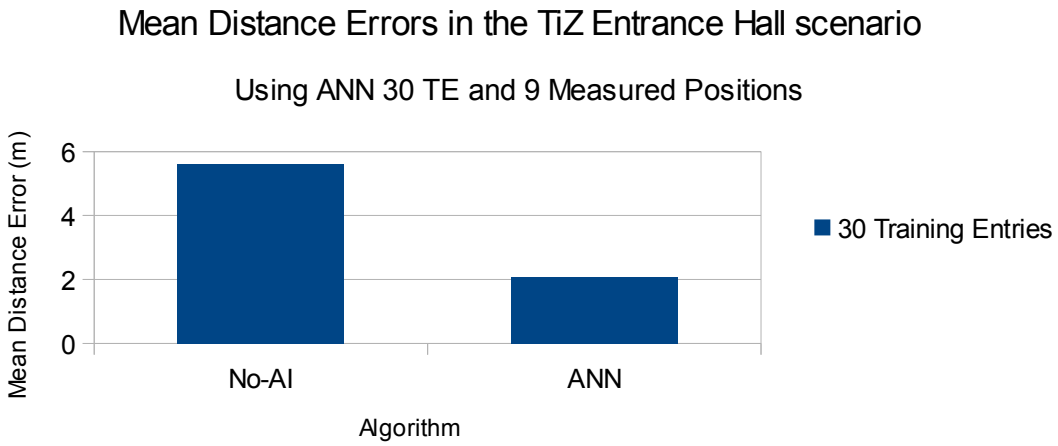

Figure 5.4: MDE in the TiZ Entrance Hall using 9 MP using ANN 30 TE.

Figure 5.4 shows the obtained MDE in the ANN 30 TE test done in the TiZ Entrance Hall scenario using 9 Measured Positions. It demonstrates that the MDE of the ANN 30 TE approach using 9 MPs is almost 3 times smaller that the MDE of the No-AI approach.

It can be concluded that not all of the implemented ML algorithms in all the situations show a lower MDE than the No-AI approach, although in the majority of them do, as it was shown and discussed above.

Hypothesis 2: *One of the implemented ML algorithms obtains a lower Mean Distance Error than the other implemented ML algorithms.*

In order to answer to this Hypothesis, data will be used from Hypothesis 1, in order to establish a comparison between all the tests that obtained a lower Mean Distance Error than the approach that does not used AI. From all of those approaches, there has to be at least one that obtains the lowest of the Mean Distance Errors. That approach will be the answer to Hypothesis 2.

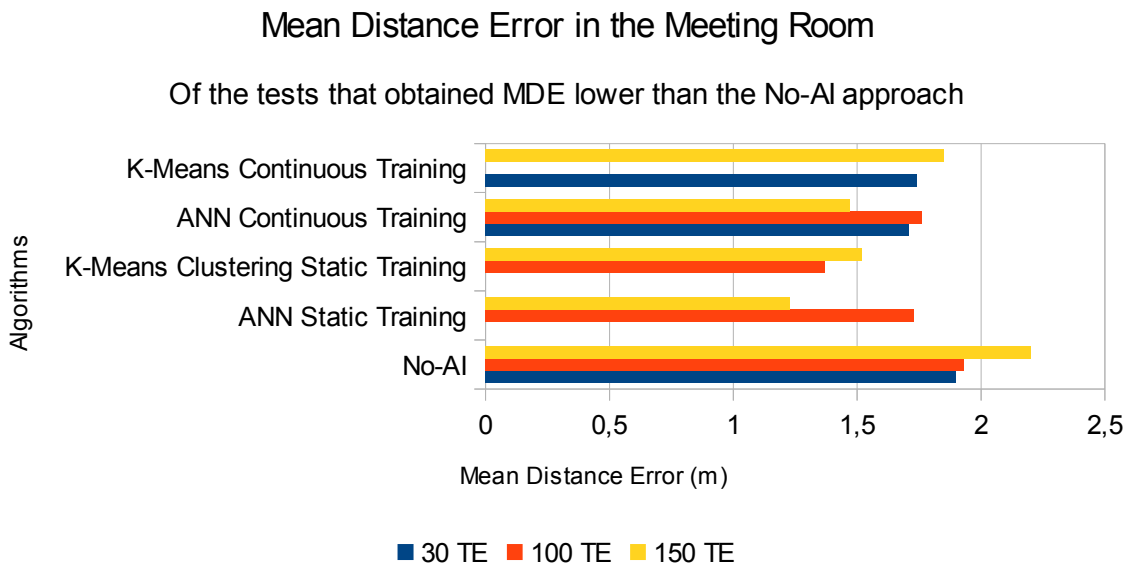

Figure 5.5: MDE of tests that obtained lower MDE than No-AI.

Figure 5.5 shows the Mean Distance Errors of the tests that obtained lower MDE than the No-AI approach. The answer for Hypothesis 2 in the Meeting Room scenario can be obtained by the observation of it. The answer is that the configuration that obtained the lowest Mean Distance Error in the Meeting Room scenario is the ANN using Static Training and with 150 TE.

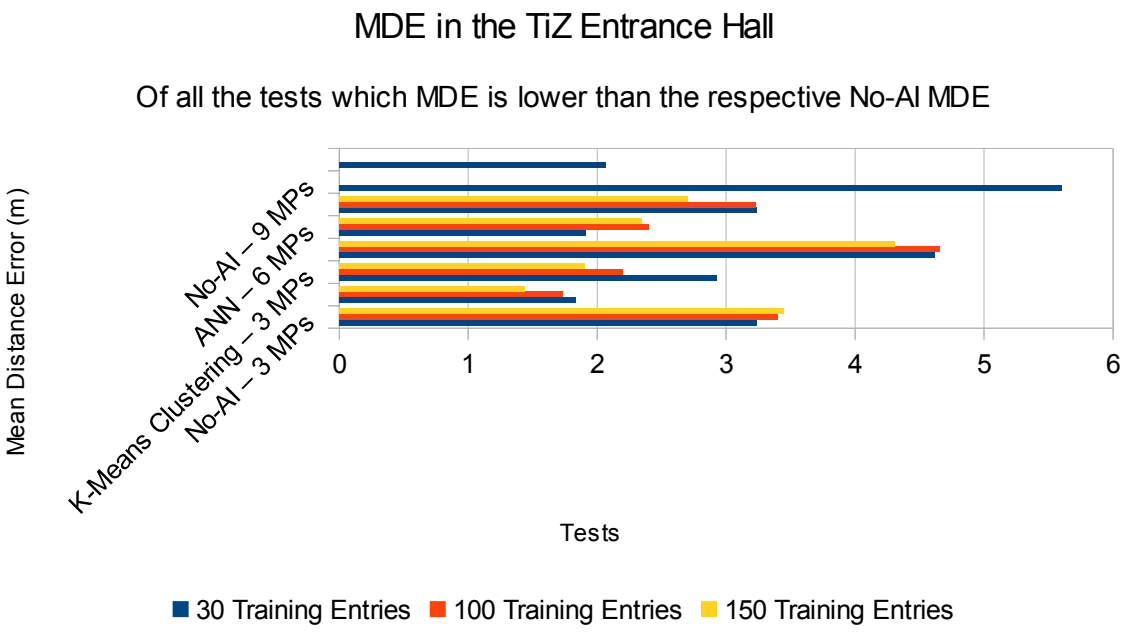

Figure 5.6: MDE of tests which MDE is lower than the respective No-AI MDE.

Figure 5.6 demonstrates the Mean Distance Errors of the tests that obtained lower MDE than the respective No-AI MDE approach. The answer for Hypothesis 2 in the TiZ Entrance Hall scenario can be obtained through the observation of it. The answer is that the test that obtained the lowest Mean Distance Error in the TiZ Entrance Hall is the ANN approach using 3 MPs, Static Training and 150 Training Examples.

Henceforth, it can be concluded from the tests done towards the answer of this Hypothesis, that in the Meeting Room scenario the approach that obtained the lowest MDE was the ANN using Static Training and 150 Training Entries with the value of 1.23m and the approach that achieved the lowest MDE in the TiZ Entrance Hall scenario was the ANN using 3 MPs, Static Training and 150 Training Examples with the value of 1.44m.

Hypothesis 3: *One of the implemented ML algorithms learns at a faster rate as the number of Training Examples grows.*

Table 5.1: Change metrics in the MR scenario between TE.

	30-100 TE	100-150 TE	Mean Change	Change 30 – 150 TE
ANN – Static Training	-55,49%	-40,65%	-48,07%	-118,70%
ANN – Continuous Training	1,16%	-17,69%	-8,27%	-16,33%
SVM – Static Training	-4,40%	14,66%	5,13%	10,90%
SVM – Continuous Training	-4,33%	-20,96%	2,50%	5,95%
K-Means Clustering – Static Training	-77,37%	9,87%	-33,75%	-59,87%
K-Means Clustering – Continuous Training	12,56%	-7,57%	-12,64%	-26,19%

Table 5.1 aggregates the percentile changes from the previous test of the same test configuration, the Mean of percentile changes and the overall percentile change from 30 Training Examples to 150 Training Examples of the Mean Distance Errors results in the Meeting Room scenario. The highest percentile change from 30 to the 100 Training Examples test was the one of the k-Means Clustering algorithm using Static Training, with a reduction of 77.37 % in the Mean Distance Error; the highest percentile change from the 100 to the 150 Training Examples test was the one of the ANN approach using Static Training, with a shrinkage of 40.65%; the lowest value of the Mean Change, representing the highest mean curtailment of all the percentile changes was also the one of the ANN approach using Static Training, with a mean cutback of 48.07 %; the highest value of retrenchment between the 30 Training Examples test and the 150 Training Examples case was again the one of the ANN algorithm using Static Training, with the value of 118.7 % of reduction of the Mean Distance Error.

Commenting the statement that Hypothesis 3 does, the implemented ML algorithm that learns at a faster rate as the number of Training Examples grows is the ANN algorithm using Static Training for the case of the Meeting Room scenario.

Table 5.2: Percentile change metrics in the TiZ Entrance Hall between TE.

	30-100 TE	100-150 TE	Mean Change	Change 30-150
ANN – 3 MP	-5,78%	-20,14%	-12,96%	-27,08%
ANN – 6 MP	20,42%	-2,13%	9,14%	18,72%
SVM – 3 MP	-39,39%	13,16%	-13,12%	-21,05%
K-Means Clustering – 3 MP	-33,18%	-15,79%	-24,49%	-54,21%
K-Means Clustering – 6 MP	-0,31%	-19,63%	-9,96%	-20,00%

Table 5.2 contains the percentile changes from the test to test done in the TiZ Entrance Hall scenario when using each one of the algorithms and configurations tested. From the 30 Training Examples to the 100 TE tests, the approach that has the highest reduction is the one of the SVM algorithm using 3 Measured Positions; in the case of the percentile change between the 100 and 150 TE tests, the approach that has the highest shrinkage of the MDE is the ANN

algorithm using 3 Measured Positions, with a value of 20.14 %; the approach that had the highest Mean of percentile changes was the k-Means Clustering using 3 Measured Positions, with a value of 24.49%; the approach which, overall, had a higher percentile of reduction between the 30 TE test and the 150 TE test was also the k-Means Clustering with 3 Measured Positions, with a value of 54.21%.

Hence, the implemented ML algorithm that learns at a faster rate as the number of Training Examples rises in the is the k-Means Clustering algorithm using 3 different Measured Positions for training.

Hypothesis 4: *Measured Positions from which were gathered more samples have a lower Mean Distance Error than the ones from which were gathered less samples*.

Not necessarily. It depends from algorithm to algorithm. In the 3 MPs using ANN and Static Training case, for instance, as the number of TE grows the value of MDE decreases. But in the 6 MPs test that does not happen anymore. In this last case the value of MDE rises slightly, stabilizing as the number of TE grows. In the case of the k-Means Clustering algorithm, the same happens in the 3MPs and 6MPs case. With the k-Means algorithm and taking into consideration that this affirmation is based only in the gathered data and in the tests done with it in the TiZ Entrance Hall scenario, it's a fact that for 3 MPs and 6MPs, the MDE decreases as the number of Training Examples grows.

Hypothesis 5: *Until a certain limit of Measured Positions, the number of Measured Positions used for training decreases the Mean Distance Error of each one of the ML algorithms.*

It's false, according to the tests presented in this document (Chapter 4). As the number of Measured Positions rises, the MDE Error rises slightly too. In some cases it rises more than the first test done with the previous number of MPs. Still, the only algorithm that achieves a stabilization in the MDE metric as the number of MPs grow is the ANN.

Hypothesis 6: *One of the implemented ML algorithms learns at a faster rate than the other implemented ML techniques, as the number of samples from the same Measured Positions grows* .

The cases where the number of samples from the same Measured Positions influence the learning process were the ANN in the TiZ Entrance Hall Scenario (3 MPs) test, k-Means Clustering in the TiZ Entrance Scenario (Static Training – 3 MPs) and the k-Means Clustering in the TiZ Entrance Scenario (Static Training – 6 MPs) cases. The one that presents a faster learning rate as the number of Training Examples grow for the same Measured Positions is the k-Means Clustering in the TiZ Entrance Scenario (Static Training – 3 MPs).

Hypothesis 7: *The Mean Distance Error in smaller-sized scenarios is lower than in bigger-sized scenarios.*

True for all the tested cases, when comparing algorithm by algorithm using the same configuration. The bigger the scenario, the higher the MDE for the same algorithm using the same training configuration.

Hypothesis 8: *Using a kind of Training that per each classification entry added to the database uses that same entry as a valid Training Example for the next position inference has a lower Mean Distance Error than a Training with a static number of entries.*

It was an Hypothesis built around the concept that Continuous Training would bring improvements every time a new sample from the same real position was gathered, since it would allow the algorithms to train in the next iteration with the new data from the real position just before. It's actually false, because Continuous Training almost always behaves in the opposite direction, misleading the algorithm to infer more distance positions in relationship to the real one of the mobile device.

5.2 Contributions

Several contributions were given through the development of this project. The first of them was the implementation and test of 3 different Machine Learning algorithms with the goal to enhance the accuracy of the previous indoor positioning technique that did not use any Artificial Intelligence methodology.

Other of the contributions was the analysis of the obtained results and the correlations established between the influence that some descriptive statistical values have in the quality of the position inferences. The results gathered and the analysis done will allow the creation of better learning models based on several already calculated factors and metrics.

Other of the contributions is related with the answering to all the hypothesis formulated in the beginning of the project, which allowed to conclude about the influences that the number of Measured Positions and Training Examples have in this kind of learning process.

5.3 Benefits for the Company

Several decisions can be taken from the current Dissertation. First, the amount of decisions taken was based on either scientific facts or on parametrized and evaluated decisions. Hence, one of the benefits for the company in keep going with the development of such a project is that actually the work developed here will ease some decisions in the future and create shorter paths between the goals and their achievement. Second, although the resources and time were very limited, the presented work produces better results than the ones with the usage of a non-intelligent technique, which means that the project is going on the right direction, although there is still a lot of things to improve.

The company benefited from the data gathered in real testing scenarios. Even in the possibility that the company chooses not to use the current implementation of any of the algorithms, the testing data can be the same for the same scenarios, since its configuration does not change drastically meanwhile.

The company benefited from the knowledge acquired relatively to Machine Learning algorithms and their way of processing information and learning with it. This document makes a review of the latest ML approaches to the indoor positioning constraint and implements 3 of the ones that show the best results. The company benefits with that knowledge, that application of the ML techniques and the comparisons and conclusions taken from the whole study.

5.4 Future Improvements

Towards obtaining inferred positions closer to the real mobile device's real positions using the implemented, tested and analyzed indoor positioning techniques there are several possible improvements. They can be applied in different sectors of the implemented methodology to infer position in indoor environments. The first proposed improvement is related with developing a filter of the RSSI values. There are several different established methodologies that correlate space and time between consecutive RSSI signals - [SBDO13] - or that propose an improved approach for the Least-Squares algorithm [RA13]. The second proposed improvement, still related with the RSSI signal treatment is to add estimations of signal attenuations and change due to the presence of obstacles which characteristics and locations are determined - [CL13] and [SX13]. Theoretically it seems to be a good approach to determined the influence of certain obstacles – depending on their sizes and material characteristics - in the RSSI values. A study would have to be done in order to understand which attenuation each obstacles have in the RSSI values and a model should be proposed to add that to the current Lateration algorithm developed in the project.

Due to the fact that one of the global goals of the current project is that the produced software has the lowest configuration possible, one of the proposals for future improvements is to add evolutionary computing to generate populations of data based on previous populations' characteristics and, in the current project's case, in the scenario's configurations. Taking into account that the data available in the used data set for ML algorithm training are mainly distances to the Access Points and inferred positions, this data can be classified as Low-Quality Data (which is generated with high amounts of noise) - [PSC13]. Hence, up-sampling for skewed classes is necessary, in order that data from positions that do not exist in the data set starts existing. This process can be done via the generation of synthetic data using existing interpolation techniques - [D10] and [JZ10].

Several improvements can be done in the ML algorithms, mainly related with the process of cross-validation [O12]. This process can be improved either by tuning the model parameters - [SKM07] - and/or by analyzing the bias-variance trade-off and through modifications in the training process achieve some desired equilibrium in that correlation - [SNA14] and [B13]. In order to do that, a complete study will have to be developed applying

modifications to the models of the implemented algorithms or of the model of the algorithm that show the best performance in the position inference process.

The last proposed improvement is the division of the whole process using Machine Learning algorithms to infer positioning in two parts: a server-side part, that trains the chosen algorithm and generates the model outputs that will be used for the positioning inference, with the goal of sparing computational time in the consumers' mobile devices; a client-side that is basically the Android application that with the less amount of data possible accurately infers the position of the user's mobile device. This improvement would bring scalability to the whole system and would spare the client-side of the heaviest task of the whole process. Although one of the implemented techniques improved the accuracy in a ratio of more than 110 %, this process still needs to become more accurate. Hence, the first future efforts should be in that direction and only when the results would be satisfactory enough it should be done the last proposed improvement.

5.5 Lessons Learned

Although the goals of this dissertation were successfully achieved, some mistakes related with several aspects of the work, methodologies and choices were done, which contributed for the project to flow out in the most appropriate and productive way. This sub-chapter explains each one of those mistakes in a reflective mode, useful to the author himself in order t improve his work methodology and useful also to whoever works on top of the work done.

The mistakes identified are mainly related with the timing of some of the tasks done throughout the followed methodology and not with the content itself. These mistakes were done mainly because of lack of experience when implementing ML or Statistical Learning algorithms, of poor time management in some phase of the whole 10 months of this Dissertation's development and of big time restraints relatively to the development of some tasks.

Starting by the first phase of the Dissertation's development, the Literature Review phase, more approaches should have been considered to evaluate which algorithms to implement. Although choosing 3 algorithms to implement and test is already good content to guarantee some comparisons' quality, the inclusion of Statistical Learning algorithms was not even considered back in that phase. Due to the fact that Machine Learning and Statistical Learning are complementary fields nowadays, the integration of a Statistical Learning algorithm in the Artificial Neural Networks algorithm, for instance, would have guaranteed improved results, specially when considering the integration of one of those techniques in the cross-validation process. That would have allowed some extra capacity for the ANN algorithm to improve its solutions through time and data. On the other hand, as it was already stated, time constraints did not allow such study to happen, hence it was proposed in Chapter 5.4 as an add-on to the project – obviously, it will be necessary that some different approaches are studied and

chosen and that some of those are implemented, tested and compared in order to understand which performs better so it can be produced an improved solution to the current project's problem.

Still in the Literature Review, it should have been allocated time to study and implement the Future Improvements proposed by [C12]. It would have contributed highly for the understanding of how the RSSI signals should be treated, bringing progress to the promises of achieving higher quality entry data for the algorithms, more accurate results for the Lateration process and, consequently, a lower MDE overall.

In the Development and Testing phases, the main mistake done is related with the timings of how each task was done. In this Dissertation case, the approach was to develop the 3 algorithms, then test them all in different scenarios and only then treat the achieved results. Obviously, some conclusions were taken too late in the time available to finish the Dissertation, which meant that new tests could not be done taking into account conclusions taken in previous tests. Hence, it can be concluded that the right methodology should have been to implement one of the algorithms, test it and analyze the results, in order to understand and conclude about its performance and statistical metrics immediately and so it would be possible to apply some conclusions – mainly related with training configurations – in the tests to do after. This would have saved time in the testing phase and would have allowed to an earlier understanding of the results analysis, which would have meant that the testes to do after would have been reconsidered, in some cases.

Definitely these are learned lessons and there will be some effort put in avoiding these same mistakes in the future, contributing to more robust conclusions about the several available algorithms and about the right methodology to be followed. Surely this will allow the project to flow more naturally, avoiding time spent in tasks that are not necessary or do not matter much for the conclusions to take in each moment.

References

[ABFB10] Altini, Marco; Brunelli, Davide; Farella, Elisabetta and Luca Benini. "Bluetooth indoor localization with multiple neural networks", IEEE 5th International Symposium on Wireless Pervasive Computing 2010, 2010.

[ARS97] Alsabti, K.; Ranka, S. and V. Singh. "An efficient k-Means clustering algorithm", , 1997.

[AS11] Altintas, Bulut and Tacha Serif. "Improving RSS-based Indoor Positioning Algorithm via K-Means Clustering", , 2011.

[B13] Bartz, Daniel, et al.. "Directional Variance Adjustment: Bias reduction in covariance matrices based on factor analysis with an application to portfolio optimization", PloS one, 2013.

[BBV05] Brunato, M.; Battiti, R. and A. Villani. "Statistical learning theory for location fingerprinting in wireless LANs", Computer Networks, 2005.

[Ber06] P. Berkhin. "A Survey of Clustering Data Mining Techniques", , 2006.

[C12] Carvalho, R.M.S.. "Seamless Indoor/outdoor Positioning Using Smartphones", , 2012.

[CGGRC09] Chen, Y.; Garcia, E.K.; Gupta, M. R.; Rahimi, A. and L. Cazzanti. "Similarity-based Classification: Concepts and Algorithms", The Journal of Machine Learning Research, 2009.

[CL11] Chang, Chih-Chung, and Chih-Jen Lin. ""LIBSVM: a library for support vector machines", ACM Transactions on Intelligent Systems and Technology (TIST) 2, 2011.

[CL13] Cao, Zhi Chao and De Wei Li. "Exploration of Parameter Factor in the RSSI Model", Applied Mechanics and Materials, 2013.

[CMM83] Carbonell, J.G.; Michalski, R.S. and T.M. Mitchel. "An Overview of Machine Learning", , 1983.

[CV95] Cortes, C.; Vapnik, V. "Support Vector Networks", , 1995.

[D10] Dolson, Jennifer, et al.. "Upsampling range data in dynamic environments.", Computer Vision and Pattern Recognition (CVPR), 2010.

References

[DH04] Ding, C. and X. He. "K-Means Clustering via Principal Component Analysis", ICML '04 Proceedings of the twenty-first international conference on Machine learning, 2004.

[DLD01] Dayhoff, Judith E. and James M. DeLeo. "Artificial Neural Networks - Opening the Black Box", , 2001.

[FAWJC12] Figuera, Carlos; Rojo-Álvarez, José Luis; Wilby, Mark; Mora-Jiménez, Inmaculada and Antonio J. Caamaño . "Advanced support vector machines for 802.11 indoor location", , 2012.

[FNR11] Fonseca, Humphrey C.; Neves, Ana Régia de M. and Célia Ghedini Ralha. "A User Location Case Study Using Different WirelessProtocols", , 2011.

[FRM13] Farid, Zahid, Rosdiadee Nordin and Ismail Mahamod. "Recent Advances in Wireless Indoor Localization Techniques and System", , 2013.

[FS99] Freund, Y. and Schapire, E.. "Large margin classification using the perceptron algorithm", , 1999.

[Gen05] Genco, Alessandro. "LNCS 3479 - Three Step Bluetooth Positioning", , 2005.

[Hea02] Jeff Heaton. "Using JOONE for Artificial Intelligence Programming", , 2002.

[Hea10] Jeff Heaton. *Introduction to Encog 2.5 for Java.*. Heaton Research, , 2010.

[Her02] Herbrich, R.. "Learning Kernel Classifiers", , 2002.

[Hua98] Z. Huang. "Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values", Data Mining and Knowledge Discovery, 1998.

[Jai10] Anil K. Jain. "Data Clustering: 50 years beyond K-Means", 9th International Conference in Pattern Recognition , 2010.

[JBPA06] Ji, Yiming; Biaz, S.; Pandey, Santosh and Prathima Agrawal . "ARIADNE: a dynamic indoor signal map construction and localization system", , 2006.

[JTR10] Jiang, Jianmin; Trundle, P. and Jinchang Ren. "Medical image analysis with artificial neural networks", Computerized Medical Imaging and Graphics, 2010.

[JZ10] Jekabsons, Gints and Vadim Zuravlyov. "Refining Wi-Fi based indoor positioning", Proceedings of 4th International Scientific Conference Applied Information and Communication Technologies (AICT), 2010.

[LC07] E.E.L. Lau, W.Y. Chung. "Enhanced RSSI-based Real-Time User Location Tracking System for Indoor and Outdoor Environments", International Conference on Convergence Information Technology, 2007.

[LDBL07] Liu, Hui; Darabi, Houshang; Banerjee, Pat and Jing Liu. "Survey of Wireless Indoor Positioning Techniques and Systems", , 2007.

[LL05] Lin, Tsung-nan and Po-Chiang Lin. "Performance Comparison of Indoor Positioning Techniques based on Location Fingerprinting in Wireless Networks",

International Conference on Wireless Networks, Communications and Mobile Computing, 2005.

[LLCL12] Li, Shuai; Liu, Bo; Chen, Baogang and Yuesheng Lou. "Neural network based mobile phone localization using Bluetooth connectivity", Neural Computing and Applications, 2012.

[MCB12] Mohammad Gholami, Ningxu Cai and Robert W. Brennan. "Evaluating alternative approaches to mobile object localization in wireless sensor networks with passive architecture", Computers in Industry, 2012.

[MLTL08] Ma, J.; Li, X.; Tao, X. and J. Lu. "Cluster filtered KNN: A WLAN-based indoor positioning scheme", , 2008.

[MME10] Mengual, Luis; Marbán, Oscar and Santiago Eibe. "Clustering-based location in wireless networks", , 2010.

[MTT10] Mehmood, Hamid; Tripathi, Nitin K. and Taravudh Tipdecho. "Indoor Positioning System Using Artificial Neural Network", Journal of Computer Science, 2010.

[O12] Ounpraseuth, Songthip, et al. "Estimating misclassification error: a closer look at cross-validation based methods", BMC research notes, 2012.

[PaC11] Papapostolou, Apostolia, and Hakima Chaouchi. "RFIDassisted indoor localization and the impact of interference on its performance", Journal of Network and Computer Applications 34, 2011.

[Pai11] Paiva, Marco . "Study on Indoor Mapping Using Mobile Computing", , 2011.

[PC11] Papapostolou, Apostolia; Chaouchi, Hakima. "RFID-assisted indoor localization and the impact of interference on its performance", , 2011.

[PKY06] Pan, J.J.; Kwok, J.T. and Q. Yang . "Multidimensional vector regression for accurate and low-cost location estimation in pervasive computing", , 2006.

[PM00] Pelleg, Dan and Andrew Moore. "X-Means: Extending K-Means with Efficient Estimation of the Number of Clusters", , 2000.

[PPCTL12] Park, J.g.; Patel, A.; Curtis, D.; Teller, S. and J. Ledlie. "Online Pose Classification and Walking Speed Estimation using Handheld Devices", , 2012.

[PSC13] Palacios, Ana Maria; Sanchez, Luciano and Ines Couso. " A software tool for modeling and decision making with Low Quality Data", , 2013.

[RA13] Mehra, Rajesh and Abhishek Singh. "Real time RSSI error reduction in distance estimation using RLS algorithm.", Advance Computing Conference (IACC), 2013.

[RB93] Riedmiller, Martin and Heinrich Braun. "A direct adaptive method for faster backpropagation learning: The RPROP algorithm", IEEE International Conference on Neural Networks, 1993.

[RN10] Russell, Stuart and Peter Norvig. "Artificial Intelligence: A Modern Approach", , 2010.

References

[SBDO13] de San Bernabe, A., J. R. Dios, and A. Ollero. "Mechanisms for efficient integration of RSSI in localization and tracking with wireless camera networks", Intelligent Robots and Systems (IROS), 2013.

[SBWA04] Shawe-Taylor, J.; Bartlett, P.L.; Williamson, R.C. and M. Anthony. "Kernel methods for pattern analysis", , 2004.

[SCQL08] Sun, Zhuo; Chen, Yiqiang; Qi, Juan and Junfa Liu. "Adaptive Localization through Transfer Learning in Indoor Wi-Fi Environment", 2008 Seventh International Conference on Machine Learning and Applications, 2008.

[SCS06] Su, C.T.; Chen, Y.C. and D.Y. Sha. "Linking innovative product development with customer knowledge: a data-mining approach", , 2006.

[SKM07] Sugiyama, Masashi; Krauledat, Matthias and Klaus-Robert Muller . "Covariate shift adaptation by importance weighted cross validation", The Journal of Machine Learning Research, 2007.

[SNA14] Sharma, Rahul; Nori, Aditya V. and Alex Aiken. "Bias-variance tradeoffs in program analysis", Proceedings of the 41st annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages, 2014.

[SX13] Shen, Gang and Zegang Xie. "Wi-Fi RSS Based Indoor Positioning Using a Probabilistic Reduced Estimator", Active Media Technology, 2013.

[SZQ02] Sun, Y.; Zhu, Q. and Z. Chen. "An iterative initial-points refinement algorithm for categorical data clustering", Pattern Recognition Letters, 2002.

[TBPA11] Tapia, D.I.; Bajo, Javier; Paz, J.F. De and R.S. Alonso. "Using Multi-Layer Perceptrons to Enhance the Performance of Indoor RTLS", , 2011.

[Tho99] Joachims, Thorsten. "Svmlight: Support vector machine", SVM-Light Support Vector Machine http://svmlight. joachims. org/, University of Dortmund 19, no. 4, 1999.

[TII10] Tsoumakas, Grigorios; Ioannis, Katakis and Vlahavas Ioannis. "Mining multi-label data", Data mining and knowledge discovery handbook, 2010.

[TN08] Tran, D.a. and T. Nguyen. "Localization In Wireless Sensor Networks Based on Support Vector Machines", , 2008.

[TWH01] Tibshirani, R.; Walther, G. and Trevor Hastie. "Estimating the number of clusters in a data set via the gap statistic", , 2001.

[WBLP09] Wang, Xinwei; Bischoff, Ole; Laur, Rainer and Steffen Paul. "Localization in Wireless Ad-hoc Sensor Networks using Multilateration with RSSI for Logistic Applications", , 2009.

[WF05] Witten, I. H. and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. , , 2005.

References

[WOK13] Wozniak, Marek, Waldemar Odziemczyk, and Kamil Nagorski. "Investigation of practical and theoretical accuracy of wireless indoorpositioning system UBISENSE", EGU General Assembly Conference Abstracts, 2013.

[WSE12] Wang, He; Sen, Souvik and Ahmed Elgohary. "No need to war-drive: Unsupervised indoor localization", , 2012.

[YASN02] Youssef, M.A.; Agrawala, Ashok; Shankar, A.U. and S.H. Noh. "A probabilistic clustering-based indoor location determination system", , 2002.

[Yeg06] B. Yegnanarayana. *Artificial Neural Networks*. Prentice Hall of India, , 2006.

[YYC11] Yang, Jie, Yingying Chen, and Jerry Cheng. "Improving Localization Accuracy of RSS-Based Lateration Methods in Indoor Environments", Ad Hoc & Sensor Wireless Networks 11, 2011.

Appendix A: Calculation of Number of Hidden Layers and Hidden Nodes in each layer for ANN

Iteration	Number of Hidden Layers	Number of Nodes/HL	Accuracy (%)	Number of Training Epochs	Computational Time (s)
1	1	1	25.80645161	1500	0.989802
2	1	2	38.70967741	1500	0.933423
3	1	3	45.16129032	1500	1.056648
4	1	4	58.064516129	1500	1.114759
5	1	5	58.064516129	1500	1.371655
6	1	6	80.645161290	1500	1.322614
7	1	7	74.1935483	1500	1.353572
8	1	8	83.870967741	1500	1.497172
9	1	9	83.870967741	1500	1.632598
10	1	10	74.193548387	1500	1.691649
11	1	11	74.193548387	964	1.086988
12	1	12	77.419354838	1389	1.790549
13	1	13	83.870967741	1500	1.910371
14	1	14	80.645161290	1500	2.01684999
15	1	15	87.09677419	1500	2.098537
16	2	1	12.9032258	1500	0.899995
17	2	2	45.1612903	1500	1.080226
18	2	3	64.516129	1500	1.276193

Appendix A: Calculation of Number of Hidden Layers and Hidden Nodes in each layer for ANN

19	2	4	74.19354838	1427	1.368215
20	2	5	83.8709677	1500	1.626511
21	2	6	80.645161	855	0.9868149
22	2	7	80.645161	904	1.219611
23	2	8	83.870967	1359	2.131844
24	2	9	80.645161	758	1.223009
25	2	10	80.64516	380	0.715453
26	2	11	83.87096	575	1.052385
27	2	12	87.0967741	567	1.247403
28	2	13	87.09677419	579	1.31147
29	2	14	83.87096774	681	1.67885
30	2	15	80.645161	383	0.959101
31	3	1	29.03225	1500	0.899804
32	3	2	9.677419354	1500	1.099405
33	3	3	74.193548387	1500	1.501153
34	3	4	80.6451612	1500	1.741761
35	3	5	74.1935483	1143	1.501669
36	3	6	80.645161	1122	1.6320009
37	3	7	87.096774	874	1.52384
38	3	8	80.6451612	595	1.131878
39	3	9	87.09677	427	0.933737
40	3	10	80.64516129	538	1.249331
41	3	11	83.8709677	310	0.875379999
42	3	12	80.645161	460	1.373801
43	3	13	87.096774	427	1.33675
44	3	14	74.193548	496	1.749088
45	3	15	83.870967	331	1.287708
46	4	1	12.90322	1500	1.014183
47	4	2	38.7096774	1500	1.3603
48	4	3	48.387096	1500	1.6977
49	4	4	58.0645161	1500	1.910457
50	4	5	87.096774193	680	1.074357
51	4	6	87.096774193	846	1.074357
52	4	7	77.4193548	482	1.010293
53	4	8	83.8709677	458	1.061704999
54	4	9	80.6451612	315	0.800299
55	4	10	74.193548	633	1.829711

Appendix A: Calculation of Number of Hidden Layers and Hidden Nodes in each layer for ANN

56	4	11	80.64516129	489	1.740961
57	4	12	87.096774	429	1.618796
58	4	13	80.64516129	270	1.12945
59	4	14	87.0967741	400	1.843493
60	4	15	83.8709677	264	1.311692
61	5	1	29.032258	1500	0.985792
62	5	2	41.9354838	1500	1.4122799
63	5	3	32.258064	1500	2.022662001
64	5	4	54.8387096	1500	2.230267
65	5	5	54.8387096	1500	3.041701
66	5	6	83.870967	1035	2.266035999
67	5	7	74.19354838	1500	3.702572
68	5	8	74.19354838	1500	4.503097999
69	5	9	80.6451612	431	1.330525
70	5	10	45.16129	1500	5.93492599
71	5	11	80.64516129	373	1.495337
72	5	12	83.870967	599	2.693128
73	5	13	83.870967	341	1.639249
74	5	14	77.4193548	500	2.776538
75	5	15	74.1935483	396	2.446818
76	6	1	12.903225	1500	1.116268
77	6	2	9.6774193	1500	1.400619
78	6	3	48.3870967	1500	2.056651
79	6	4	51.6129032	1500	2.449998
80	6	5	80.64516129	1104	2.135619
81	6	6	41.9354838	1500	3.629033999
82	6	7	77.419354838	1199	3.147441
83	6	8	83.8709677	1089	3.499297
84	6	9	64.516129	1500	5.5723
85	6	10	74.19354838	551	2.250825
86	6	11	83.870967	894	4.0878889
87	6	12	83.870967	643	3.50676
88	6	13	77.4193548	318	1.889457
89	6	14	74.1935483	487	3.390775
90	6	15	77.41935483	452	3.156806
91	7	1	0.0	1500	1.143237
92	7	2	12.9032258	1500	1.7261159

Appendix A: Calculation of Number of Hidden Layers and Hidden Nodes in each layer for ANN

93	7	3	12.9032258	1500	1.99594
94	7	4	3.22580	1500	2.6714969
95	7	5	54.8387	1500	3.231032
96	7	6	45.16129	1500	4.031078
97	7	7	77.419354838	1500	4.668905
98	7	8	80.6451612	786	2.69309
99	7	9	77.4193548	565	2.387752999
100	7	10	93.548387	1500	7.266924
101	7	11	83.870967	1500	7.957937999
102	7	12	83.870967	597	3.688105
103	7	13	87.0967741	403	2.77424999
104	7	14	80.6451612	660	5.185798
105	7	15	83.870967	811	6.6860429
106	8	1	12.9032258	1500	1.2604989
107	8	2	12.9032258	1500	1.848066
108	8	3	35.4838709	1500	2.21327
109	8	4	51.612903	1500	3.072878
110	8	5	54.8387096	1500	3.558075
111	8	6	48.3870967	1500	4.521462
112	8	7	61.2903225	1500	5.256088
113	8	8	83.8709677	1441	5.80868
114	8	9	80.6451612	1500	7.344836
115	8	10	67.7419354	1500	7.8961
116	8	11	80.6451612	1500	9.511596
117	8	12	83.8709677	1500	10.425561
118	8	13	77.4193548	1500	12.16762
119	8	14	70.9677419	1500	13.262474
120	8	15	77.41935483	478	4.501536
121	9	1	12.9032258	1500	1.456837
122	9	2	25.806451	1500	1.9446659
123	9	3	3.225806	1500	2.583706
124	9	4	16.129032	1500	3.231458
125	9	5	3.225806	1500	3.899017998
126	9	6	38.709677	1500	5.099651
127	9	7	3.22580645	1500	5.487026
128	9	8	54.8387096	1500	7.061505
129	9	9	58.0645161	1500	7.751884

Appendix A: Calculation of Number of Hidden Layers and Hidden Nodes in each layer for ANN

130	9	10	64.516129	1500	9.171317
131	9	11	61.2903225	1500	10.493027
132	9	12	77.4193548	1500	11.557368
133	9	13	74.1935483	1500	13.565326
134	9	14	80.6451612	838	8.131084
135	9	15	83.87096774	1314	14.58897199
136	10	1	12.9032258	1500	1.455706999
137	10	2	12.9032258	1500	1.917088
138	10	3	25.8064516	1500	2.858711
139	10	4	9.677419354	1500	3.346744
140	10	5	29.032258	1500	4.38179
141	10	6	12.9032258	1500	5.203436
142	10	7	25.80645161	1500	6.359293
143	10	8	51.612903	1500	7.505463998
144	10	9	51.612903	1500	8.580824
145	10	10	64.516129	1500	10.297119
146	10	11	51.612903	1500	11.251641
147	10	12	67.7419354	1500	13.39752
148	10	13	64.516129	1500	14.421187
149	10	14	54.8387	1500	16.835034
150	10	15	87.0967741	1500	18.3140659
151	11	1	12.9032258	1500	1.464941
152	11	2	12.9032258	1500	2.253488
153	11	3	12.9032258	1500	2.724257
154	11	4	9.67741935	1500	3.784749
155	11	5	3.2258064	1500	4.455126
156	11	6	38.709677	1500	5.899604
157	11	7	35.483870	1500	6.719255
158	11	8	48.387096	1500	8.183712
159	11	9	38.709677	1500	9.554694
160	11	10	16.1290322	1500	10.8472489
161	11	11	74.1935483	1500	12.846828
162	11	12	70.9677419	1500	13.981578
163	11	13	38.709677	1500	16.6262
164	11	14	77.419354	1500	17.78696
165	11	15	83.870967	1500	20.469057002
166	12	1	12.9032258	1500	1.658594

Appendix A: Calculation of Number of Hidden Layers and Hidden Nodes in each layer for ANN

167	12	2	12.9032258	1500	2.216495
168	12	3	12.9032258	1500	3.157143
169	12	4	16.129032258	1500	3.900647
170	12	5	12.9032258	1500	5.068064
171	12	6	45.16129	1500	5.910134
172	12	7	74.193548	1500	7.552242
173	12	8	29.032258	1500	8.981192
174	12	9	19.3548387	1500	10.256665
175	12	10	19.3548387	1500	11.791726
176	12	11	35.48387	1500	13.588489
177	12	12	35.48387	1500	15.6256169
178	12	13	74.19354838	1500	17.596353
179	12	14	67.74193548	1500	20.0501759
180	12	15	58.0645161	1500	21.449264