INTERNATIONAL
HELLENIC
UNIVERSITY

Validation of simple RSSI based fingerprint algorithms for positioning in wireless networks

Vasiliki Tzouveli

SID: 3301110013

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Information and Communication Systems

OCTOBER 2012

THESSALONIKI – GREECE

Validation of simple RSSI based fingerprint algorithms for positioning in wireless networks

Vasiliki Tzouveli

SID: 3301110013

Supervisor:	Prof. Maria Papadopouli
Supervising Committee Members:	Prof. Vasilis Katos
	Dr. George Koutitas

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Information and Communication Systems

OCTOBER 2012

THESSALONIKI – GREECE

Abstract

This dissertation was written as a part of the MSc in ICT Systems at the International Hellenic University.

It is based on the concept of fingerprinting which is used for positioning in wireless networks, primarily. Fingerprints can be generated using statistical information of the collected signal-strength measurements from various landmarks (e.g. Access Points). In particular, in the context of the algorithms for location-sensing systems, we implemented two simple algorithms, the Nearest Neighbor(s) in Signal Space algorithm – as described in RADAR system [5] – and the Percentiles method [8]. In order to obtain the respective simulation results, we have been provided with two datasets (training and runtime) with signal-strength values from known positions and an unknown position, respectively, from the Telecommunications and Networks Laboratory (TNL) at FORTH.

At this point, I would like to sincerely thank Prof. Maria Papadopouli and Dr. George Koutitas for their guidance, instructions during this work as well as their significant support and patience. I would like also to mention that without the support and the continuous encouragement from my family and friends, as well, this project could not be achieved.

Vasiliki Tzouveli

29 Oct. 2012

Contents

1 Introduction

"Difficulties are opportunities to better things,
they are stepping stones to greater experience."
~ Bryan Adams (from 'How to Succeed') ~

"Emerging mobile computing applications often need to know where things are physically located. To meet this need, many different location systems and technologies have been developed." [2] Over the last few years, significant research has been done in the domain of location-sensing using signal-strength measurements. The framework of the following location-sensing systems will be analyzed in this project, with a Literature Review on Positioning, presenting the basic properties and techniques used for location-sensing as well as a survey of research related work (Section 2).

In Section 3, we will describe the framework for the existing fingerprinting methods (percentiles, confidence intervals, empirical distribution and Multivariate Gaussian model). Afterwards, a comparative performance analysis will be analysed with some simulation results, as they were presented at the respective Paper [8] for the testbed of Telecommunication and Network Lab (TNL) at FORTH.

Finally, in Section 4, there will be a description of the development of two simple algorithms for a fingerprinting location-sensing system in order to infer position, using the methods of RADAR and Percentiles. The fingerprinting position system will read RSSI values per AP from training and runtime cells. Finally, the cell with the minimum Euclidian Distance/weight will be reported as the estimated position.

2 Literature Review on Positioning

In this Chapter, we will analyze the basic characteristics of location-sensing systems, based on the research that conducted on relative publications. Specifically, the location-sensing systems are classified based on their location-sensing properties. We will describe the IEEE 802.11 technology that is usually used in location-sensing systems as well as the basic concepts of each principal technique used for location-sensing. Furthermore, we will focus on the generation of statistical-based fingerprints using the collected RSSI measurements per Access Point. Finally, we will end up with the related work in the area of location-sensing using signal-strength measurements.

2.1 Location-Sensing Properties

Location-sensing system can be classified in several categories based on their properties that are presented below. This classification forms a reasonable taxonomy for characterizing or evaluating location systems, independent of the technologies or techniques they use. [2]

2.1.1 Description of Position

First of all, the location of a respective system can be described as either *physical* or *symbolic*. In the first case, the location-sensing system provides the physical position (e.g. GPS, which is a clearly physical positioning technology) while in the second case, the system provides a virtual-grid representation of the physical space (e.g. barcode scanners and systems that monitor computer login activity).

In addition, coordination systems can be characterized as *absolute* or *relative*. An absolute location system uses a shared reference grid for all located objects (e.g. all GPS receivers use latitude, longitude, and altitude). In a relative system, each object can have its own frame of reference (e.g. 2 meters from an Access Point).

Some significant requirements needed for evaluating location-sensing systems is the distinction between *accuracy* and *precision*. Accuracy is the difference between the estimated position from the real-value position, by reporting the location error. A result is

considered to be accurate, if it is consistent with the true or accepted value for that result. Precision of a system is the repeatability of a measurement or the percentage of times that position will be in an acceptable region of errors, after having put a threshold, indicating how often we can expect to get that accuracy. It does not require to us to know the correct or true value. The main goal of all location-sensing systems is to achieve both high accuracy and high precision and be considered robust. [1], [2], [3]

2.1.2 Location Computation

Another classification concerning the computations is whether the collection of the measurements takes place *locally* or *remotely*. Locally means that the measurement collection takes place in the same device that is going to compute the user's position later on (like GPS [2] or RADAR [5]), whereas remotely means that a device (i.e. laptop, PDA) does the collection of measurements and send them to another device (e.g. a server) which makes all the appropriate computations for estimating the position of the mobile device. Small devices (e.g. smartphones) use remote computations βecause of their size. If all these computations took place at such a device, more processing power would be required, thus, the operations of the device would be very slow.

2.1.3 Methodology

Each location-sensing system can estimate the distances, orientation, and position, using one of the methods below:

- *Signal-strength measurements* (e.g., Radar [5]), if the velocity of the signal and a signal attenuation model for the given environment, respectively, are known. Received signal-strength indication (RSSI) is a relative measurement of the strength of a radio signal's energy as measured by a wireless network interface card (NIC). RSSI is used by a wireless NIC to determine if other devices are transmitting on the channel or if it is "C*lear To Send*" (CTS). It is also used to determine when a wireless signal decreases enough that the device should attempt to roam to a new device, called the “roaming threshold”. The RSSI scale is different for each NIC vendor because the vendor defines its own RSSI maximum which is less than or equal to 255. In order to compute an average RSSI, we use a free wireless network monitoring tool that measures the RSSI values at regular intervals and divides their sum by the number of observations. [9]

- *Time of Arrival – ToA* (e.g. GPS [2]) is a travel time measurement. It is about a metric that can comment on the quality of the link.
- *Angle of Arrival*; this is another method to take into consideration since it is used in the Angulation technique, as described below. [2]

2.1.4 Scale

A location-sensing system may be able to locate objects worldwide, within a metropolitan area, throughout a campus, in a particular building, or within a single room. Further, the number of objects the system can locate with a certain amount of infrastructure amount of infrastructure or over a given time may be limited. For example, GPS can serve an unlimited number of receivers worldwide using 24 satellites plus three redundant backups. On the other hand, some electronic tag readers cannot read any tag if more than one is within range.

2.1.5 Recognition

An automatic identification mechanism is needed for applications that have to recognize or classify located objects for a specific action based on their location. For example, a modern airport baggage handling system automatically routes outbound and inbound luggage to the correct flight or claim carousel, through tag scanners installed at key locations. In contrast, GPS satellites have no inherent mechanism for recognizing individual receivers. Systems with recognition capability like cameras and vision systems may recognize only some feature types (e.g. color or shape of an object but not individual people), thus, the question is if there are capabilities for identification of the user in terms of location-sensing.

2.1.6 Cost

There are various types of cost for a location-sensing system. Time costs include factors such as installation process's length and system's administration needs. Space costs involve the amount of installed infrastructure and the hardware's size. Capital costs include factors such as the price per mobile unit or infrastructure element and the salaries of support personnel that will monitor the system. [2]

2.1.7 Limitations and Dependencies

Positioning systems can be classified depending on their specialized infrastructure, their hardware availability as well as the different modalities that may employ.

With regards to *specialized infrastructure*, there are various systems depending on infrastructure and other that do not. For example, there are systems that receive positioning information from an AP which is wired connected with the Internet and, as a result, the users can have Internet connection. However, a dense deployment of a wireless infrastructure for communication and location-sensing may not be feasible due to environmental, cost, and regulatory barriers. In such situations, ad-hoc location-sensing systems determine positioning mechanisms so as to exploit cooperation by enabling devices to share positioning estimates (e.g. CLS [3]). However, the ad-hoc approach is still under research but it has not been applied in a real-life testbed. [10]

In terms of *limitations*, there are systems that do not function in certain environments. For instance, GPS is very useful outdoors but it is ineffective indoors because buildings block LOS from the GPS transmission. [2]

Regarding *hardware availability*, there are location-sensing systems that depend on specialized hardware (i.e. tags used for placing QR codes for traceability reasons) to locate a wireless device. However, there are others (like fingerprinting methods that will be described below) that have no need of specialized hardware, but they are based on existing infrastructure, such as the IEEE 802.11 infrastructure of Access Points (APs). The devices that have *a priori* knowledge of their location (e.g. APs configured with their positioning coordinates) will be referred as *landmarks*. The landmarks can broadcast positioning information (e.g. local id, maximum wireless range and estimated position) in the form of beacon.

Finally, positioning systems may employ *different modalities*, such as:

- IEEE802.11 (Radar [5], CLS [3])
- Ultrasonic (Cricket [12, 13], Active Bat [14])
- Infrared (Active Badge [16])
- Bluetooth
- 4G
- Vision
- Physical contact with pressure (Smart Floor), touch sensors or capacitive detectors

The popularity of IEEE802.11 infrastructures, their low deployment cost, and the advantages of using them for both communication and positioning, make them an attractive choice. [3]

2.2 IEEE 802.11

IEEE 802.11 is a set of standards (802.11a, 802.11b, 802.11g, or 802.11n) for implementing Wireless Local Area Network (WLAN) communication in the 2.4, 3.6 and 5 GHz frequency bands. These standards provide the basis for wireless network products using the Wi-Fi brand name. [11]

2.3 Location-Sensing Techniques

Most of the signal-strength based localization systems can be classiffed into the following two categories, namely the distance-prediction based and the signature or map-based. We will also examine the Proximity location-sensing technique.

2.3.1 Distance-prediction based techniques

This type uses signal-strength values to estimate the distance of a wireless client from any landmark (i.e. an AP) based on radio propagation models. What is applied is the triangulation technique, which uses the geometric properties of triangles to compute object locations. Triangulation is divisible into the subcategories of lateration, using distance measurements, and angulation, using primarily angle or bearing measurements. A survey of positioning systems can be found in [2].

Lateration

Lateration computes the position of an object by measuring its distance from multiple reference positions. Calculating an object's position in two dimensions requires distance measurements from 3 non-collinear points as shown in Figure 1. In 3 dimensions, distance measurements from 4 non-coplanar points are required. Domain-specific knowledge may reduce the number of required distance measurements.

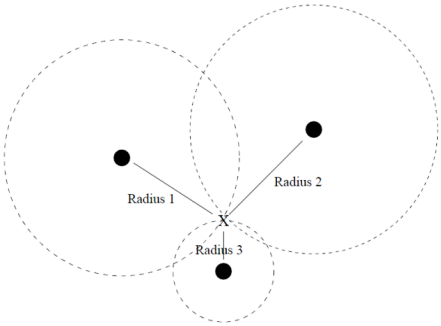


Figure 1: Determining 2D position using lateration requires distance measurements between the object 'X' and 3 non-collinear points.

In order to measure the distances required by the lateration technique, there are three general approaches:

1. *Direct measurement of distance*, which uses a physical action or movement, which is simple to understand but difficult to obtain automatically due to the complexities involved in coordinating autonomous physical movement. [2]

2. *Path Loss Attenuation* (in dB), which is calculated based on the transmitted and the received signal strength and is proportional to the square of the distance between the transmitter and receiver. The measurement of path loss provides a distance estimate between the mobile object and the base station. This means that the mobile object must lie on a circle which has as a center the base station and as radius the distance between them. The path loss can be found if the mobile object knows the power transmitted from the base station and the received power. The general path loss model is shown below:

$$P(d) = P_0 - 10n \log_{10}(d / d_0)$$

n: path loss exponent
P(d): the average received power in dB at distance d
P_0: the received power in dB at a short distance d_0

Thus, using three base stations as it is represented in Figure 1, the system can compute the distance (e.g. the radius of each circle) between the mobile object and each base station, respectively, as an equation for a given velocity and time of arrival of the signal. The intersection of the 3 circles (or the solution of the system of the 3 equations that are coordinated) is the estimated position of the user.

However, attenuation varies based on the environment. Signal propagation issues such as shadowing and multipath cause the attenuation to correlate poorly with distance re-

sulting in inaccurate and imprecise distance estimates. [2] Therefore, examining the radio propagation models, we can conclude the following:

- The *Free Space model* is an Empiric Model and is based on the several measurements of the signals with the path loss exponent to be equal to two. Because of the reasons that were indicated above, this model does not correspond to the real environmental situations.
- The *Two-Ray model* is a Deterministic Model and observes the existence of two empiric phenomena; the reflection of the transmitted signal and the Line-Of-Sight (LOS). The path loss exponent in this case is equal to four. This model is more complex than the previews one, but it does not incorporate the need for precise environmental profiling.
- The *Probabilistic Model* is suggested due to the fact that it models the environment by fitting the signal strength measurements in probability distributions with the path loss exponent to be equal to n. This model tries to find the probability that the estimated position of the user will be in an acceptable region of errors (e.g. fading, shadowing) that occur during the radio propagation. For example, Gaussian distribution describes very well the case when there are no errors during propagation, while Rayleigh distribution is appropriate for describing fading in NLOS situations. [1], [7]

3. *Time-of-flight*, which measures the time it takes the emitted signal to travel from an object to some point P, requiring a known velocity. For example, sound waves have a velocity of approximately 344 m/s in 21°C air. Therefore, an ultrasound pulse sent by an object and arriving at point P 14.5 milliseconds later allows us to conclude that the object is 5 meters away from point P. Measuring the time-of-flight of light or radio may also require clocks with much higher resolution (by six orders of magnitude) than those used for timing ultrasound. For example, a light pulse emitted by the object has a velocity of 299,792,458 m/s and will travel the 5 meters to point P in 16.7 nanoseconds. Some time-of-flight location-sensing systems include GPS and the Cricket Location System [12]. Ignoring pulses arriving at point P via an indirect (and hence longer) path caused by reflections in the environment is a challenge in measuring time-of-flight since direct and reflected pulses look identical. [1]

GPS, an excellent lateration framework using time-of-flight approach

The Global Positioning System is a space-based satellite navigation system and is perhaps the most widely publicized location-sensing system, based on an unobstructed LOS from four or more GPS satellites which are at fixed positions. GPS is actually a constellation of 27 earth-orbiting satellites (24 in operation and three extras in case one fails). The orbit altitude is such that the satellites repeat the same track and configuration over any point approximately each 24 hours.

The receiver is not synchronized with the satellite transmitters and thus cannot precisely measure the time it took the signal to reach the ground from space. Therefore, GPS satellites are precisely synchronized with each other and transmit their local time in the signal, allowing receivers to compute the difference in time-of-flight.

There are four unknowns to describe the position of a GPS receiver; not only the three components of GPS receiver position (longitude, latitude and elevation) but also the clock bias, e.g. the error between the receiver clock and the synchronized satellite clocks [x, y, z, b]. Because of the forth unknown, an additional measurement from a fourth satellite is required in GPS receiver instead of only 3 satellites that would normally be required to estimate a 3D position. As a consequence, with four equations (4 satellite signals) and four unknowns, the GPS receiver solves for the unknowns. Assuming that the GPS receiver knows the location of 4 satellites in orbit and the distance from each satellite to the receiver, he can calculate its position in time and space. [2,15]

It is worth mentioning that other GPS error sources are noise, troposphere delays due to weather changes (e.g. temperature, pressure, humidity), ionospheric delays (the speed of a electromagnetic wave will deviate from the speed of light in a vacuum as the wave encounters water vapor and other atmospheric material) and multipath which is very difficult to be detected and sometimes hard to be avoided (caused by reflected signals near the receiver that can either interfere with or be mistaken for the signal that follows the LOS from the satellite).

To maintain synchronization, each of the 27 GPS satellites contains four cesium/rubidium atomic clocks which are locally averaged to maintain accuracy and are updated daily by US Air Force GPS ground control. [1, 2, 15] For an excellent summary of GPS theory, you may refer to [2].

Angulation

Angulation is similar to lateration except, instead of distances, angles are used for determining an object's position. 2D angulation requires at least two angle measurements and the distance measurement between two reference points (Figure 2). Phased antenna arrays are an excellent enabling technology. Multiple antennas with known separation measure the ToA of a signal. Given the differences in arrival times and the geometry of the receiving array, it is possible to compute the angle from which the emission originated. If there are enough elements in the array and large enough separations, the angulation calculation can be performed. [2]

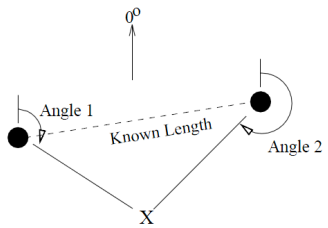


Figure 2: 2D angulation, for locating the object 'X' using angles relative to a 0° reference vector and the distance between two reference points.

2.3.2 Signature or map-based techniques

This type creates a *Signal-Strength (SS) signature or map of the physical space* after gathering RSSI values from beacons collected from APs at various positions (APs are at a fixed and known positions). The physical space is represented as a grid of cells with fixed size and well-known coordinates. The basic concept of this technique is that SS is a function of the user location, i.e. a location-sensing system may estimate the position by applying pattern matching algorithms that relate SS measurements, acquired from messages exchanged between devices, to their position on the terrain or their distance in wireless networks [1], [3]. A description of such algorithms is presented at the following chapters.

The advantage of signature-based technique is that the objects' location can be determined using passive observation and features that do not correspond to geometric angles or distances. However, their main disadvantage is that environmental changes alter the signal strength, thus a reconstruction of the predefined dataset or retrieval of an entirely new dataset may be needed. [2] Signature-based techniques will be analyzed in this project, starting from Chapter 2.4.

2.3.3 Proximity

A proximity location-sensing technique entails determining when an object is “near” a known location. The object's presence is sensed using a physical phenomenon with limited range. There are three general approaches for sensing proximity:

1. *Detecting physical contact with an object*, using e.g. pressure, touch and capacitive field detectors. Capacitive detection has been used to replace the Computer Mouse and implement a Touch Mouse using direct contact with a person’s skin [2].

2. *Monitoring the connectivity of a mobile device when it is in the range of one or more access point in a wireless cellular network*. The cell geometry is associated with the wireless technology used in this implementation. For example, a radio cellular network cell may have the shape of the region containing object 'X' while a diffuse infrared cell in a room is constrained by the walls resulting in a square shape (used by Active Badge [16]).

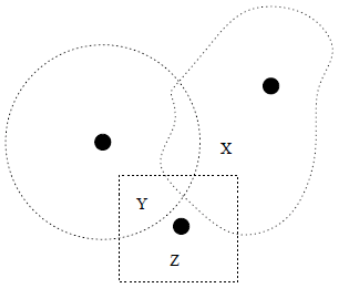


Figure 3: Objects 'X', 'Y', and 'Z' are located by monitoring their connectivity to one or more access point in a wireless cellular network.

3. *Observing automatic ID systems*, using automatic identification systems such as credit card point-of-sale terminals, computer login histories and identification tags (e.g. electronic highway E-Toll systems and UPC product codes. If the device scanning the label, interrogating the tag, or monitoring the transaction has a known location, then the location of the mobile object can be determined.

Proximity approaches may need to be combined with identification systems. For example, the Contact system enables communication between the objects that a user is touching. Then, all these objects can exchange identification information over the same communication channel. In contrast, the Touch Mouse and pressure sensors require an additional identification system since the method used to detect proximity does not provide identification directly. [2]

2.4 Generation of Fingerprints

As we mentioned previously, the physical space is represented as a grid of cells with fixed size and well-known coordinates. A wireless device that listens to a channel receives the beacons sent by APs (at that channel) periodically and records their RSSI values. Wireless devices that run fingerprint based positioning systems acquire such measurements and generate statistical fingerprints, each of them representing different environmental conditions, for a position using these measurements. [8]

The concept of such systems is that during the *Training Phase*, there is a mobile user with a mobile device that walks to different, known locations, associating the observed signal-strength measurements (ss_j) acquired from the j^{th} Access Point with the physical coordinates *(x, y, z)* at each cell *i*. Thus, a training fingerprint is generated e.g. a vector with entries of the form $(x_i, y_i, z_i, ss_{j\ (j=1...n)})$. Each entry of the vector corresponds to one AP.

Similarly, during the *Runtime Phase*, a static user tries to determine its position by measuring the signal strength of the j^{th} AP within range at the unknown position of the cell *i** with coordinates *(x*, y*, z*)*. The runtime fingerprint that is created is a vector with entries of the form $(x^*, y^*, z^*, ss_{j\ (j=1...n)})$. In order to estimate its position, the user is running a pattern-matching algorithm which compares the two fingerprints by finding the cell whose training fingerprint has the minimum "distance" from the runtime one.

There are various *methods* that can be derived for the fingerprint comparison, based on the statistical characteristics of the signal-strength measurements, such as confidence intervals and percentiles, the empirical distribution or the parameters of a theoretical distribution (e.g. Multivariate Gaussian) [3], [8]. Fingerprint comparison depends on the statistical properties of the fingerprint e.g. Euclidean distances of the vectors and Kullback-Leibler Divergence test. [1]

A fingerprint can be built using various statistical properties e.g. mean and standard deviation, percentiles, empirical distribution (entire set of signal strength values), theoretical models (e.g. multivariate Gaussian).

An extensive description of fingerprinting methods will follow at Chapter 3.

2.5 Related Work

Over the last few years, significant research has been done in the domain of location-sensing using signal-strength measurements. The framework of the following location-sensing systems will be analyzed.

2.5.1 RADAR

RADAR is a location and tracking system, developed by a Microsoft Research group in order to compute the 2D position of a user within few meters of his actual position on a single floor of a building. The basic system was built over an off-the shelf RF WLAN. Bahl *et al.* improved its performance using a WLAN under IEEE 802.11 protocol to increase the chances of large-scale deployments. The algorithm that was implemented was the *Nearest Neighbor(s) in Signal Space (NNSS)* which is based on the calculation of the Euclidian distance between the two vectors (e.g. training and runtime fingerprint, respectively) for each cell and their size is equal to the number of APs that appear in both the training and runtime measurements. [5]

Algorithm 1: NNSS based on the Euclidian Distance between 2 vectors

1. During Training phase:
   - Collect RSSI measurements from APs at each cell at known position
   - Create the training fingerprint Tij(c) which is an m×n matrix of training RSSI values collected for each cell $i=c$ $(i=1,\ldots,m)$ from $j=1,\ldots,n$ APs.
2. During Runtime phase:
   - Collect RSSI measurements from each AP at the unknown position.
   - Create the runtime fingerprint Rj which is an $1\times n$ vector of runtime RSSI values collected at the unknown cell $i=i^*$ from the j^{th} AP
3. During Runtime phase, compare the runtime with the training fingerprint by calculating the Euclidian distance between the 2 vectors (e.g. training and runtime fingerprints) for each cell 1.

$$d = \sqrt{\sum_{j=1}^{n} \left(R_j - T_{i,j}(c)\right)^2}\ ,\ i=1,\ldots,m \text{ and } j=1,\ldots,n$$

Where,

- *m is the number of cells*
- *n the number of APs, thus, the number of RSSI values acquired from APs*
- *Ti,j(c) is an mxn vector with training RSSI values from the j^{th} AP at the cell i=c,*
- *Rj is an 1xn vector with runtime RSSI values from the j^{th} AP at the unknown cell i=i*.*

4. Report as the estimated position the cell *c** with the minimum Euclidian distance.

For example, let's suppose that:

1. $R_j = [ss_1' \; ss_2']$ e.g. 2 RSSI values from 2 APs at cell *i** with coordinates (x^*,y^*,z^*).
2. $T_{ij} = [ss_1 \; ss_2; \; ss_3 \; ss_4]$ e.g. 2 RSSI values from 2 APs at the 2 different cells with co-ordinates (x_1, y_1, z_1) and (x_2, y_2, z_2), respectively.
3. Then, the minimum Euclidian distance corresponds to the cell

$$c^* = argmin\{sqrt((R_j - T_{ij}(1))^2 + \ldots + (R_j - T_{ij}(n))^2), \text{ for } i=1,2 \text{ and } j=1,2\} \text{ or}$$

$$argmin\{\underline{\mathbf{sqrt}}((ss_1'-ss_1')^2 + (ss_2'-ss_2')^2), \underline{\mathbf{sqrt}}((ss_1'-ss_3')^2+(ss_2' - ss_4')^2)\}$$

4. The minimum is the estimated position at cell *i** with coordinates (x^*, y^*, z^*) e.g.

$$If \; min\{(x^*, y^*, z^*) \text{ corresponds to } (x_1, y_1, z_1)\}$$

$$else \; \{(x^*, y^*, z^*) \text{ corresponds to } (x_2, y_2, z_2)\}$$

One variant of the basic NNSS algorithm is the *NNSS-AVG algorithm*, which is used in case that there is more than one SS tuple in the Radio Map "close" in signal space to measured SS tuple. Then, the NNSS-AVG algorithm picks a small number of closely matching tuples and averages their physical location to obtain an estimate of the user's location. Often this results in a better estimate than any individual tuple.

Performance of RADAR

Regarding the performance of the system, the NNSS algorithm computes the *location error* as the *Euclidian Distance* between the true location and the estimated location of the user. The location error has a median of 2 to 3 meters, about the size of a typical office room in the building.

Regarding the performance of the 2 deployments of RADAR, the mean location error and the 90th percentile of the error distance for the basic version are 2.65 m and 5.93 m, respectively, while the corresponding values for the extended version are 2.37 m and

5.97 m. Moreover, the number of APs with overlapping coverage affects the performance of the system, since there is a significant benefit in terms of location error to going from 1 AP to 2 APs and again from 2 APs to 3 APs; however, there is little benefit in going beyond 3 APs. This is because of the interference that imposes a limit on how accurately location can be inferred using the NNSS. Off course, the more APs, the more the deployment cost for the system.

The extended version of Radar was built in order to alleviate the side effects that result from SS nature e.g. aliasing and multipath. The three main improvements that were implemented in order to achieve the mentioned results were the following:

- The *continuous user tracking technique* to cope with aliasing, based on Viterbi-like algorithm [5]. Aliasing is the phenomenon that can happen because of the complex indoor propagation environment and depends on the building layout and the placement of APs. The signal strength at a point close to an AP may be similar to another point that is far away simply because of an obstruction (such as a wall) that attenuates the signal received at the former point while the latter point receives an unobstructed signal. In this case, the system uses past information (something that NNSS did not) so as to have a better guess of current location. The concept is that, due to physical constraints, the user cannot "jump around" over large distances at random, but it follows a path, based on past information. The mid-point of the path is guessed to be the user's location. This technique improved the accuracy of user location by over 33%.
- The *environmental profiling* to alleviate multipath phenomenon, achieved by using two different fingerprints. This technique improved the accuracy of the system by over a factor of 3.
- An *extension of the basic NNSS algorithm to 3D space* (i.e. to multiple floors in a building) to test if RADAR would work well, as well. The measurements were satisfied, however aliasing should be considered thoroughly (e.g. sending a document to a printer of another floor).

A big question is how well RADAR would perform in a "real-world" setting, such as a shopping mall, where the constant movement of a varying number of people changes the RF propagation environment considerably. To answer to this question, the group has some deployments and testing regarding a local mall. [5]

2.5.2 Practical Robust Localization System

The basic idea of this system as presented by Kavraki *et al.* [6] is based on the use of probabilistic techniques that remarkably determine the accurate location of a mobile device across an entire office building of over 12,000 m^2 under IEEE 802.11 protocol. The most important aspect in this system is the *scale*, since the mentioned building corresponds to a more realistic and practical use. The building was divided into 510 different cells (each cell represents a whole office with average size of 24.6 m^2) on the topological map in order to determine the position of a device. The fact that the location of the device is mapped to a cell instead of a point makes the system very **effective**, due to the fact that instead of measuring each BS's signal strength at points spaced 1-2 meters apart in order to achieve accuracy, SS measurements were collected for whole offices, treating the entire office as a single position. The system supports both static localization and dynamic tracking at speeds of over 3 m/s.

Regarding static localization (e.g. a static environment and a stationary agent), the method that was performed was Bayesian Framework [6] since it can quantify the uncertain relationship between the observed RSSI values and the position.

Algorithm 2: Bayesian Localization framework

1. During Training phase:
   - Collect RSSI measurements from base station scans in various positions of each cell, so as to cover the entire cell.
   - Create a matrix of conditional probabilities $P(o_j|s_i)$ which encode the probability of observing the RSSI values o_j given that the agent is in state s_i (e.g. the agent being in cell i).
2. During Runtime period:
   - Create $\overrightarrow{\pi_i}$ which is a vector including a prior estimate of the user's state.
   - Perform Bayesian filtering [6] which updates the estimated location $\overrightarrow{\pi_i}'$at a specific cell, e.g. $$\overrightarrow{\pi_i}' = \frac{P(o_j \mid s_i)\overrightarrow{\pi_i}}{\eta}$$ where η is the estimate normaliser or confidence e.g. $\eta = \sum_{i=1}^{n} P(o_j \mid s_i)\overrightarrow{\pi_i}$
   - Repeat the above steps until estimates converge.

This matrix of conditional probabilities is referred to as the *sensor model*. In order to determine the basic localization performance of the system, the Group used two methods for modeling the conditional probabilities:

- The Histogram sensor model. For each s_i, the $P(o_j|s_i)$ are determined by the normalized signal intensity histograms recorded during the training phase.
- The Gaussian fit sensor model, by fitting the ss measurements in a normal distribution for a given observation set $O = B \times V$, where $B = \{b_{j=1,\ldots,k}\}$ is the set of base stations and $V = \{0,\ldots,255\}$ is the set of signal intensity values. The probability of observing $(b_{j,v}) \in O$ at state s_i adds a null hypothesis and normalizes the resulting distribution which is given by the formula:

$$P((b_j, v) \mid s_i) = \frac{G_{i,j}(v) + \beta}{N_{i,j}}$$

Where:

- $G_{i,j}(v)$ *refers to Gaussian distribution determined by mean* $\mu_{i,j}$ *and standard deviation* $\sigma_{i,j}$
- β *is a small constant used to represent the probability of observing an artifact and*
- $N_{i,j}$ *is a normalizer such that* $\sum_{v=0}^{255} P((b_j, v) \mid s_i) = 1$

Performance of the system

Gaussian model proved to be a high-precision topological location technique for positioning a device to one of these 510 cells. Over all experimental trials, the Gaussian method was correct in over 97% of trials, while the histogram method in over 95% of trials. Despite the fact that both models achieved excellent accuracy, in several cells, the Histogram method had fewer than 50% accuracy. Furthermore, Gaussian model requires less training data (95% accuracy is achieved with 84 scans for Histogram and 30 scans for Gaussian), thus, much lower training time (less than 1 min per office or region) than Histogram model. As a result, a user can obtain **high level of accuracy** with only two or three ss measurements (with possibility for only 1 cell error, among 510 cells). Even in the worst case scenario, Gaussian can localize a user in **adjacent offices**.

The Gaussian model is more robust than histogram-based model as the first one can be described by only two parameters for each base station and cell, while keeping the entire histogram requires as much as 30 times more storage. This reduction increases the speed and reduces the memory requirements for localization, making it more suitable for low-power embedded devices. [6]

2.5.3 Device-free passive localization system

The concept of Device-free passive (DfP) localization is that the tracked entity need neither carry devices nor participate actively in the localization algorithm. The idea is to use installed wireless data networks to detect changes in the environment and track the location of entities passively and without requiring any devices to be attached to these entities. The DfP concept relies on the fact that RF signals are affected by changes in the environment due to the movement of the user, especially for the frequency ranges of the common wireless data networks that are currently deployed, such as WiFi, or envisioned, such as WiMax. By placing monitoring stations that continuously record physical quantities, such as signal strength or time-of-flight, DfP can analyze these signals to detect the changes in the environment and correlate them with entities and their locations.

The DfP system consists of signal transmitters (e.g. APs and stations used in traditional WiFi deployment), monitoring points (e.g. standard wireless sniffers), along with an application server for processing and initiating actions as needed (see Figure 4). Figure 5 shows an example of the measured changes of the environment due to the movement of a person. DfP functions include detecting the presence or absence of entities, tracking entities, and identifying entities. [4]

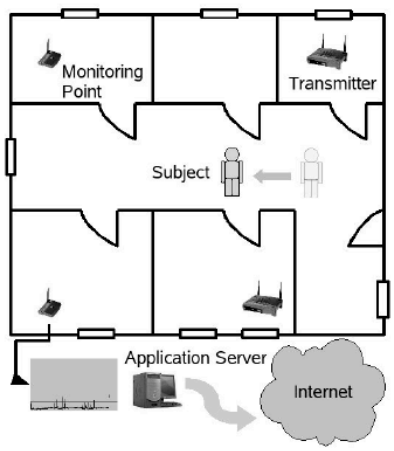


Figure 4: An example of the different components of a DfP system.

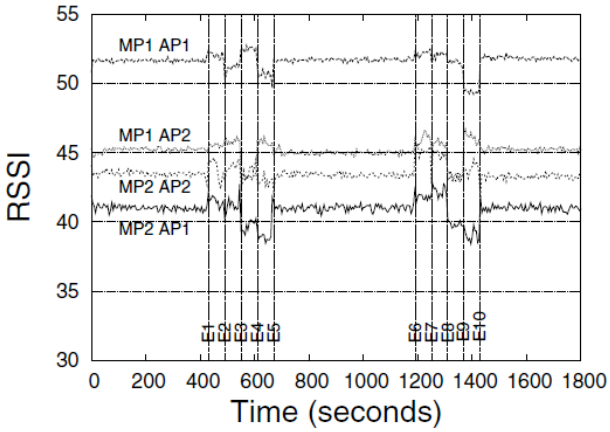


Figure 5: An example of changes in the environment due to the movement of a person (raw data) reported for four different transmitter-receiver pairs. The time of different events, such as movement of a person, is indicated by E1 through E10.

2.5.4 Collaborative Location System (CLS)

CLS is a location-sensing system proposed by the Telecommunications and Networks Laboratory (TNL) of FORTH-ICS that uses the following two features:

- probabilistic-based frameworks for transforming measurements from various sources to position and distance estimates
- the peer-to-peer paradigm.

CLS employs the peer-to-peer paradigm, enabling wireless-enabled devices to adaptively position themselves by using the existing communication infrastructure (WiFi APs) without the need of specialized hardware or training. CLS adopts a ***grid-based representation*** of the physical space; each cell of the grid corresponds to a physical position in the physical space e.g. is associated with a value that indicates the likelihood that the node is in that cell. These values are computed iteratively using a simple voting algorithm to accumulate and evaluate the received positioning information. [3]

Algorithm 3: Voting algorithm approach

A node tries to position itself on its local grid through a *voting process* in which the devices participate by sending position information and casting votes on specific cells. In particular, each local CLS instance runs an algorithm that transforms (maps) signal-strength data from beacons collected from other peers (and act as landmarks) to either distance or position estimates. The transformation algorithm can be based on a radio attenuation model or a pattern matching algorithm. These algorithms relate signal-strength measurements, acquired from messages exchanged between devices, to their position on the terrain or their distance. Based on the position information of the sender and this distance estimation, the receiver estimates its own position on the local grid. An iteration of the voting process at a local CLS instance is described below:

- Gather positioning information from other neighboring peers who act as landmarks and have already positioned themselves
- record SS measurements from the received information
- transform this information to a probability of being at a certain cell of its local grid
- add this probability to the existing value that this cell already has, and
- estimate its distance from its peers by reporting a position that corresponds to the centroid of the set of cells with maximal weight.

When the local CLS estimates its own position, it broadcasts this set of information, i.e. CLS entry, to its neighbors. Each node maintains a table with all the received CLS entries. In case that the peers have new positioning information, the host incorporates this newly received information and iteratively determines its position accordingly.

Votes may have different weights. The larger voting weight a cell has acquired, the more likely it is for the corresponding node to be located in that cell. The set of cells in the grid with maximal value indicates the potential region. Figure 6 shows a snapshot of the grid as three landmarks vote on the location of an unsolved host. Specifically, host u tries to position itself by gathering information from peers A, B and C that have already positioned themselves at the center of the co-centric disks, respectively. After accumulating the votes of the peers, host u is determined to be at the position that corresponds to the centroid of the set of cells in the grid with maximal weight and is depicted as the brightest area in the figure. [3]

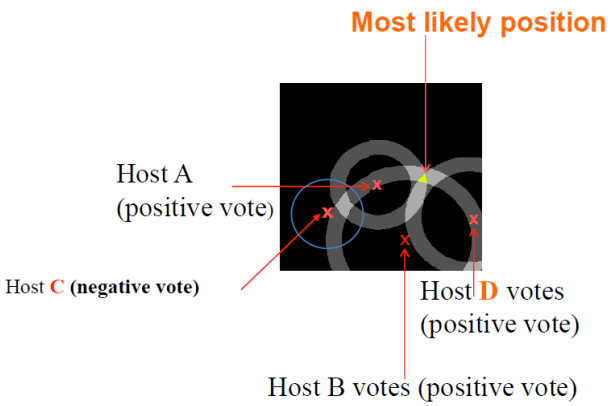


Figure 6: An example of accumulation of votes on grid cells of a host. The brighter an area, the more voting weight has been accumulated on the corresponding grid cells. The brightest area corresponds to a potential solution. The grid cell is too small to be distinguishable.

3 Implementation of already existing algorithms

The scope of this Chapter is to describe the framework for the existing fingerprinting methods (percentiles, confidence intervals, empirical distribution and Multivariate Gaussian model). Afterwards, a comparative performance analysis will be analysed with some simulation results, as it was presented at the respective Paper [8] for the test-bed of Telecommunication and Network Lab (TNL) at FORTH.

3.1 Description of Algorithms

3.1.1 Percentiles

Firstly, each cell is described by a vector of RSSI values of the beacons received from the corresponding APs. In order to construct the fingerprint of a cell, we compute the percentiles (e.g. 90^{th}, 100^{th} percentile) per AP for each cell. In other words, the fingerprint of a cell is not a vector of RSSI values but a vector of the calculated percentiles, each corresponding to an AP.

Algorithm 4: The method of Percentiles

1. During Training phase:
   - Collect RSSI measurements from APs at each cell with known position.
   - Create an $1 \times n$ vector of training RSSI values collected at each cell c from $i=1,\ldots,n$ APs.
   - Compute the $j=1,\ldots,p$ percentile e.g. $PERCENTILE([1,n],[p_1, p_2, \ldots, p_n])$ using the training measurements for the i^{th} AP at the cell c.
   - Create the training fingerprint $Tij(c)$ which is an $1 \times p$ vector containing the j^{th} percentile for each AP at cell c.
2. During Runtime phase:
   - Collect RSSI measurements from each AP at the unknown position.
   - Create an $1xn$ vector of runtime RSSI values collected at the unknown cell from the i^{th} AP.
   - Compute the $j=1,\ldots,p$ percentile e.g. $PERCENTILE([1,n],[p_1, p_2, \ldots, p_n])$ using the runtime measurements from the i^{th} AP at the unknown cell.

- Create the runtime fingerprint *Rij* which is a $1 \times p$ vector containing the j^{th} percentile for each AP at the unknown cell

3. During Runtime phase, compare the runtime with the training fingerprint by assigning a weight at cell *c w(c)* for each $i=1,...,n$ AP

$$w(c) = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{p} (R_j^i - T_j^i(c))^2} \text{ , for } i=1,...,n \text{ and } j=1,...,p$$

Where,

- *n is the number of APs,*
- *p the number of percentiles,*
- R_{ij} *the* j^{th} *percentile of runtime measurements from the* i^{th} *AP and*
- $T_{ij}(c)$ *the* j^{th} *percentile using the training measurements from the* i^{th} *AP at the cell c.*

4. Report as the estimated position the cell *c** with the **minimum** weight.

In the case of the top 5 weighted percentiles approach, the centroid of the top five cells with the minimum weights is reported as the estimated position.

This approach is similar to the confidence-interval one. However, a set of percentiles can capture more detailed information about the signal strength distribution than confidence intervals, and thus, resulting to more accurate fingerprints. [8]

3.1.2 Confidence Intervals

Similarly, each cell is described by a vector of RSSI values per AP for each cell and, then, the system computes the confidence intervals per AP. Thus, the fingerprint of a cell is a vector of the calculated confidence intervals, each corresponding to an AP. The width of the confidence interval gives us some idea about how uncertain we are about the unknown position (see precision). A very wide interval may indicate that more data should be collected before making any estimation about the position of the user.

Algorithm 5: The method of Confidence Intervals

1. During Training phase:
   - Collect RSSI measurements from APs at each cell at known position.
   - Compute the confidence limits of a confidence interval for the i^{th} AP at cell *c* e.g. $[T_i^-(c), Ti^+(c)]$
   - Create the training fingerprint $T_{ij}(c)$ which is a vector of these confidence intervals (for all APs) at cell *c*.

2. During Runtime phase:
   - Collect RSSI measurements from each AP at the unknown position.
   - Compute the confidence limits of a confidence interval for the i^{th} AP at the unknown position/cell e.g. $[R_i^-, R_i^+]$.
   - Create the runtime fingerprint R_{ij} which is a vector containing the calculated confidence intervals from all APs of that cell.
3. During Runtime phase, compare the runtime with the training fingerprint by assigning a weight at cell *c w(c)* for each *i=1,...,n* AP. In the case that:
   - the training confidence interval is included in the runtime confidence interval or
   - the runtime confidence interval is included in the training confidence interval

   the weight of that cell is increased by one. In the case of partial overlap of these two confidence intervals, the value corresponds to the ratio of this overlap.
4. Report as the estimated position the cell *c** with the **maximum** weight. [8]

3.1.3 Empirical Distribution

This method uses the Kullback-Leibler Divergence (KLD) that quantifies the proximity of two probability distributions i.e. how close the training fingerprint is to the runtime fingerprint, in our case. Only APs that appear in both training and runtime are used. Each fingerprint uses all the RSSI measurements collected per AP.

Algorithm 6: The method of Empirical Distribution

1. During Training phase:
   - Collect RSSI measurements from APs at each cell c with known position.
   - Create the training fingerprint q_i, an 1xn vector of training RSSI values collected at each cell from *i=1,...,n* APs.
2. During Runtime phase:
   - Collect RSSI measurements from each AP at the unknown position.
   - Create the runtime fingerprint pi, an 1xn vector of runtime RSSI values collected at the unknown cell from the i^{th} AP.
3. During Runtime phase, compare the runtime with the training fingerprint by assigning a weight at cell *c w(c)* for each *i=1,...,n* AP which corresponds to the average

empirical KLD distance (is always non-negative) between the training and runtime fingerprints p_i and q_i, respectively, for all APs.

$$D_{KL}(p \parallel q) = \sum_{i}^{n} p_i \log_2 \frac{p_i}{q_i}$$

4. Report as the estimated position the cell *c** with the **smallest** KLD distance.

For instance, if the two vectors match exactly, the KLD is equal to zero. Moreover, we must have $p_i = 0$ whenever $q_i = 0$, else KLD is equal to infinity.

D(p||q) is not the true distance because it is assymetric between *p* and *q* i.e. *D(p||q)≠D(q||p)* and it does not satisfy the triangle inequality, as well. [8]

3.1.4 Multivariate Gaussian model

The multivariate Gaussian-based approach takes into consideration not only the ss measurements from each AP but also the interplay or interdependencies among measurements collected from pairs of APs at a certain position. This approach provides important information about the geometry of the environment with a more accurate representation of the RSSI profiles, leading to improved positioning performance. The algorithm presented below is applied in iterations where the selected region is also divided into multiple spatial scales (sub-regions). Then, Multivariate Gaussian Model is applied in multiple spatial scales (regions). The above process is repeated in that region until the region becomes a cell. The signature comparison and position estimation is based on the Kullback-Leibler divergence (KLD).

Algorithm 7: The multivariate Gaussian-based method (spatial scale of a cell)

1. Physical space is divided into overlapping regions and each cell corresponds to a Multivariate Gaussian distribution
2. During Training phase:
   - Collect RSSI measurements from APs at each cell i with known position $c_i \rightarrow S_i = \{\overrightarrow{\mu_i}, \Sigma_i\}$, where
     - $\overrightarrow{\mu_i}$ are the mean values of the received RSSI measurements per AP
     - Σ_i is the covariance matrix (measure of spatial correlation)
   - *trainingAP(i)*: set of APs from which data are collected at cell *i*

3. During Runtime phase:
    - Collect RSSI measurements from each AP at the unknown position $c_R \rightarrow S_R = \{\overrightarrow{\mu_R}, \Sigma_R\}$
    - *runtimeAPs*: set of APs from data are collected
    - *effectiveAP(i): trainingAP(i)* $\cap$ *runtimeAP*
4. During runtime, perform the following steps for each cell i:
    - Generate the training signature for cell i using only training measurements collected from APs ∈ effectiveAP(i)
    - Generate the runtime signature using only runtime measurements collected from APs in *effectiveAP(i)*
    - Apply Multivariate Gaussian model through the density function

$$p(\vec{x} \mid \vec{\mu}, \Sigma) = \frac{1}{(2\pi)^{K/2} \mid \Sigma \mid^{1/2}} \exp\left(-\frac{1}{2} (\vec{x} - \vec{\mu})^{T} \Sigma^{-1} (\vec{x} - \vec{\mu}) \right)$$

    - Estimate the KLD distance between runtime and i^{th} training cell by measuring the similarity of the Multivariate Gaussian distributions (MvGs) i.e.

$$D(p_R \parallel p_{i,T}) = \frac{1}{2} \left(\left(\vec{\mu}^{s}_{i,T} - \vec{\mu}^{s}_{R} \right)^{T} \left(\Sigma^{s}_{i,T} \right)^{-1} \left(\vec{\mu}^{s}_{i,T} - \vec{\mu}^{s}_{R} \right) + tr\left(\sum\nolimits_{R}^{s} \left(\Sigma^{s}_{i,T} \right)^{-1} - I \right) - \ln \mid \sum\nolimits_{R}^{s} \left(\Sigma^{s}_{i,T} \right)^{-1} \mid \right)$$

5. Report as the estimated position the cell i^* with the minimum KLD distance. [8]

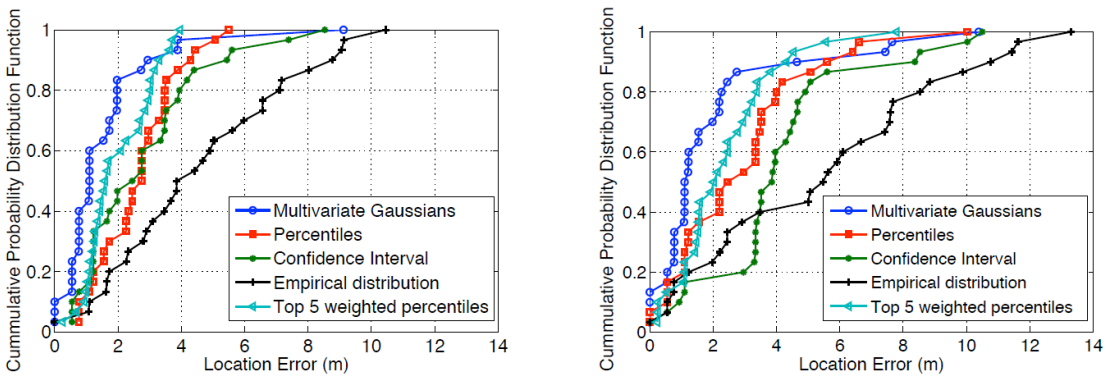
3.2 Simulation Results

The empirical-based evaluation of the different signature-based approaches took place in the TNL at FORTH, an area of 7m×12m, which was represented as a grid of cells of 55cm×55cm. During the Training phase and after collecting signal-strength values at various cells of the grid, two datasets were collected:

- One during a relatively *busy period* (around 15:00 on weekdays) with at least five people in the laboratory and several others walking in the hallways outside and
- Another one during a *quiet period* (around 23:00 on the same weekdays as the busy period dataset) with only one person in the laboratory.

The busy (quiet) period dataset included measurements from 108 (104) different cells and 13 (12) APs, respectively. On average 6 APs were detected at a given cell and more than 300 RSSI values were collected at each cell per AP. The trainer remained still for approximately 90s (30s) to collect beacons at each position during training (runtime) period, respectively. In order to capture SS values, *iwlist*, which polls each channel and acquires the MAC address and RSSI measurements from each AP (in dBm), and the tcpdump, a passive scanner relying on libpcap, for the retrieval of each packet were used.

In order to evaluate the performance of the various fingerprinting methods, the *localization error* was computed, which was measured as the Euclidean distance between the centers of the reported cell and the cell at which the mobile user was actually located at runtime (as we saw also in RADAR). 30 measurements were run at different positions (run time cells). Figures 7(a) and 7(b) illustrate the localization error of the different approaches during busy and quiet period, respectively.



(a) Busy period dataset *(b) Quiet period dataset*

Figure 7: The performance of various fingerprint positioning methods at FORTH.

After calculating the median error for the signature-based approaches, the multivariate Gaussian model (MvGs) performs much better than percentiles, confidence interval (90%) and empirical distribution approaches, as it is shown in Table 1.

Signature-based approaches	**Dataset**	
	Quiet period	**Busy period**
Multivariate Gaussian model (MvGs)	1.72m	1.60m
Percentiles	2.91m	2.65m
90^{th} confidence interval	4.16m	2.82m

Table 1: Median localization error for different signature-approaches

However, transient phenomena or radio propagation characteristics in the given environment should be carefully considered, since it might be difficult to distinguish the correct cell from other further-away cells that have similar training fingerprint with the runtime one due to aliasing phenomenon (as it was also described previously for RADAR [6]). To measure the impact of the number of APs on the localization accuracy, each i^{th} AP is associated with a *popularity index* $|\{c|APi \in effectiveAP(c)\}$ that indicates the number of cells at which there were measurements (from that AP) at both training and at runtime. The APs were sorted in a decreasing order based on their popularity index. The analysis was repeated using the top x most popular APs for the busy period and the quiet period datasets.

Figure 8 shows the impact of the number of APs on location error. *"The higher the number of APs, the lower the location error"*. However, after a certain threshold, the impact of the number of APs diminishes regarding busy period primarily.

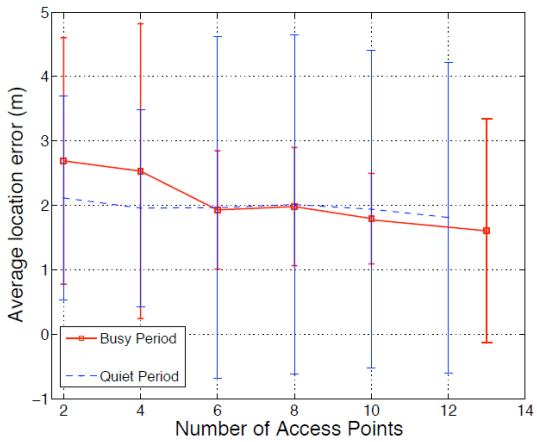
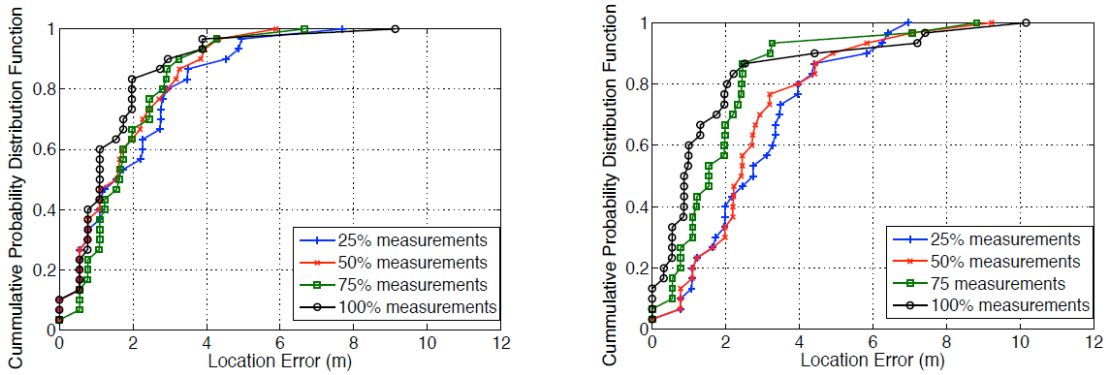


Figure 8: Impact of the number of APs on location error.

The x-axis indicates the number of the top x APs considered in both training and runtime datasets.

This happens due to the fact that the busy period dataset is more vulnerable to transient phenomena than the quiet period one. Thus, "the impact of the number of APs is more prominent in the busy period dataset than in the quiet period one". For example, in the busy period dataset, the improvement in the location error when the number of APs becomes six is about 80cm, while when the number of APs increases from six to 13, the location error is reduced by only 20cm.

Figures 9(a) and 9(b) illustrate the impact of the measurement size on the accuracy of the multivariate Gaussian-based method. The percentage (%) indicates the percentage of measurements considered in both training and runtime datasets out of the corresponding original datasets (used in the other plots). In general, *"the larger the measurement set, the more accurate the position estimation"*.



(a) Busy period dataset *(b) Quiet period dataset*

Figure 9: Impact of the number of signal-strength measurements on location error.

3.3 Conclusions

The empirical-based evaluation at FORTH revealed that the multivariate Gaussian method outperforms other signal-strength fingerprint approaches. The presence of people, the density and placement of APs as well as the size of the training set have a prominent impact on positioning. [1, 8]

4 Validation of two simple Algorithms and Testing

4.1 Development of Algorithms

In this Chapter, there will be a description of the development of algorithms for a fingerprinting location-sensing system in order to infer position, using the methods of RADAR and Percentiles. The fingerprinting position system will read RSSI values per AP from training and runtime cells and will compute the Euclidian distance and the weight (after calculating the 10 percentiles) that corresponds to each cell respectively for each method. Finally, the cell with the minimum Euclidian Distance/weight will be reported as the estimated position.

The figure below shows the floorplan of the Telecommunications and Networks Laboratory (TNL) at FORTH, an area of 7m×12m. The area is divided into cells of equal size, which was represented as a grid of cells of 0.55cm×0.55cm. The cells are represented with the gray squares. For better understanding, you may find represented the cell with coordinates (0, 0) with the red square below.

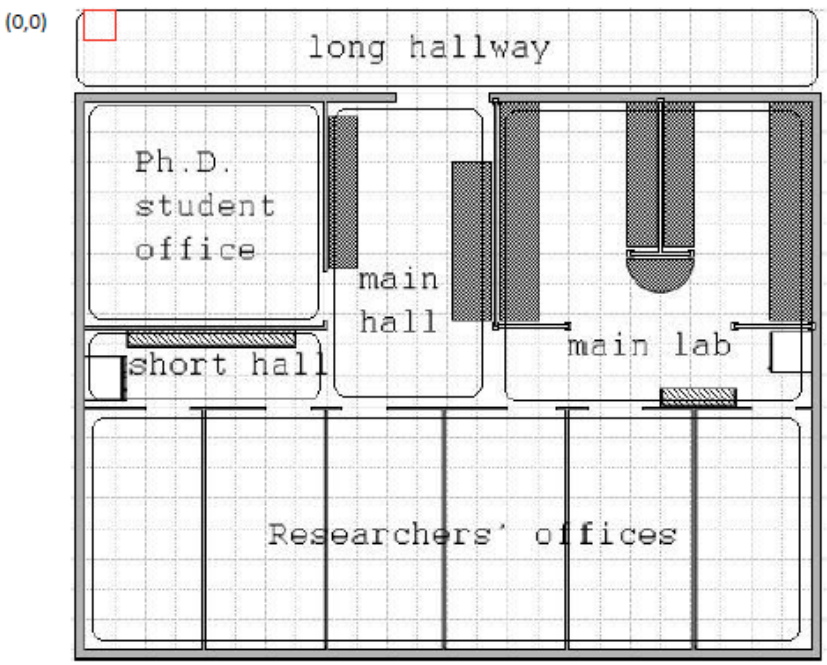


Figure 10: Grid Representation of the Telecommunications and Networks Laboratory (TNL) at FORTH

4.2 Defining Euclidean Distance and Percentiles

In mathematics, "*the Euclidean distance is the "ordinary" distance between two points that one would measure with a ruler and is given by the Pythagorean formula. By using this formula as distance, Euclidean space (or even any inner product space) becomes a metric space*".

In Cartesian coordinates, if $p=(p_1, p_2,...,p_n)$ and $q=(q_1, q_2, ...,q_n)$ are two points in Euclidean *n*-space, then the distance from *p* to *q* or from *q* to *p* is given by:

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + ... + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

The position of a point in a Euclidean *n*-space is a Euclidean vector. So, p and q are Euclidean vectors, starting from the origin of the space, and their tips indicate two points. [17]

In statistics, a percentile can be defined as "*the value of a variable below which a certain percent of observations fall"*. For example, the 80th percentile is the smallest value that is greater than or equal to 80% of the values. In our case, a signal strength measurement would be the 80th percentile if 80% of the observations were below than this. In our project, in order to calculate the percentiles of a dataset, we used the MATLAB function:

```
Y = prctile(X,p)
```

This MATLAB function returns percentiles of the values in a data vector or matrix X for the percentages p in the interval [0,100]. If X is a vector, then *Y* is a scalar or a vector with the same length as the number of percentiles required *(length(p))*. *Y(i)* contains the *p(i)* percentile.

4.3 Description of Scenario

4.3.1 Reading the data

As we mentioned, the development of the algorithms includes reading the RSSI values from both training and runtime phase. During *training*, signal strength measurements from all the active APs are collected for each cell. During *runtime*, signal strength measurements are collected from the unknown location of the user. In the programming exercise in MATLAB, we have been provided with two datasets; the "training" and "runtime_quiet", containing the mentioned signal strength measurements per AP, for some of the cells in the region, captured through *iwlist* (see Chapter 3.2). Each file in the dataset is on the form "cellx_y", indicating the cell with coordinates (x,y). Opening each file, we will notice that it contains 3 fields:

<timestamp> <AP mac address> <signal strength in dBm>

indicating the time (timestamp) at which a ss measurement from an AP in a cell was recorded by the mobile device, not the time of the event itself. Each cell with coordinates (x,y) has an id and is described by several rows and each of them corresponds to:

- the *AP_id (j=1,...,n)* that is described by its MAC address, followed by
- the set of signal strength measurements collected from the specific AP at that cell.

AP_1	$SS_{1,1}$	$SS_{1,2}$	…
AP_2	$SS_{2,1}$	$SS_{2,2}$	…
…	…	…	…
AP_n	$SS_{n,1}$	$SS_{n,2}$	…

Table 2: Representation of each Cellx_y in the datasets

We should mention that the number of the RSSI values is *not fixed* but differs per AP, e.g. in the training cell0_0 there are 83 RSSI values measured from AP with MAC Address "00_03_e2_03_d3_bf", 912 RSSI values from AP "00_11_21_61_84_a0" etc. The next that we noticed for all the cells separately was the fact that each cell is described by *different number of APs*; however, there have been 10 specific APs (see "valid" data structure in Matlab) in the area of TNL used both in training and runtime datasets. In order to achieve correct results we took into consideration only the common APs and

we included them in the "valid" list, avoiding in this way the problem of having APs that add noise to our system.

00:11:21:61:84:a0
00:11:93:03:1e:30
00:11:93:03:22:f0
00:11:93:03:29:50
00:15:62:51:f0:90
00:15:62:51:f1:70
00:15:62:51:f3:d0
00:15:62:51:f4:90
00:15:62:51:f6:00
00:15:62:51:f7:30

Table 3: MAC addresses of the 10 APs used both in training and runtime

Reading each file/cell involves creating the appropriate data structures where each field is the AP_id. Regarding the implementation in MATLAB for:

- RADAR, the function "*load_cell(set,x,y)*" takes into consideration the three arguments e.g. the dataset that this cell belongs to (training or runtime) as well as its coordinates (x, y) and returns as output the data structure "*avgSS*" where each value will be a number that denotes the average of SS measurements per AP for the cell. In order to achieve the avgSS data structure we had to create other two more structures e.g.
  - the "*sums*" structure where each value is the sum of SS and
  - the "*cnts*" structure where each value will be the counter of the current AP.

  The process involves:
  - reading each line of the file/cell
  - checking if the current MAC Address is included in the valid list; if yes, then
  - storing the current MAC Address as AP_id (in case that it is not already included in the "*sums*") in the respective field of the structure as well as its RSSI value
  - adding its RSSI value to the "*sums*" of measurements that have been already read from this specific AP and increasing "*cnts*" by 1
  - repeating the above steps until the whole cell to be read and, finally,
  - calculating and returning as output the "*avgSS*" structure e.g. the average of SS measurements for the i^{th} AP for this cell according to the formula:

$$avgSS_i = \frac{sums_i}{cnts_i}$$

- Percentiles, the function "percentiles(set,x,y)" returns as output the data structure "*percentiles*" where each value will be an array of the 10 percentiles [10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%] per AP for each cell. In this case, firstly, we had to create the "*maclist*" structure where each value is the "list" array of the SS values per AP. The process involves:
  - reading each line of the file/cell
  - checking if the current MAC Address is included in the valid list; if yes, then
  - storing in the field the current MAC Address as AP_id (in case that it is not already included in the "maclist") structure as well as its RSSI value
  - storing its RSSI value to the array of the "maclist" structure e.g. "list" after the measurements that have been already read from this specific AP
  - calculating and returning as output the "*percentiles*" structure per AP for this cell according to the command:

$$Percentile_i = prctile(list, [10,20,30,40,50,60,70,80,90,100])$$

In the table below, you may see the results after running both functions that we mentioned previously e.g. "*load_cell(set,x,y)*" – for returning the average of all the RSSI values that were observed per AP – and "*percentiles(set,x,y)*" – for returning the 10 percentiles that were calculated after reading all the RSSI values per AP – for the *cell0_0*. It is obvious that the APs are the same in both cases.

```
>> load_cell('training',0,0)

ans =

 mac00_11_21_61_84_a0: -62.2862
 mac00_11_93_03_1e_30: -51.0143
 mac00_11_93_03_22_f0: -55.6960
 mac00_11_93_03_29_50: -73.6678
 mac00_15_62_51_f0_90: -57.7430
 mac00_15_62_51_f3_d0: -78.6745
 mac00_15_62_51_f4_90: -79.2055
```

```
>> percentiles('training',0,0)

ans =

mac00_11_21_61_84_a0: [-66 -64 -63 -62 -62
                       -61 -61 -60 -59 -57]
mac00_11_93_03_1e_30: [-55 -54 -52 -51 -51
                       -49 -49 -49 -48 -38]
mac00_11_93_03_22_f0: [-59 -59 -59 -54 -54
                       -54 -54 -54 -54 -54]
mac00_11_93_03_29_50: [-78 -77 -76 -74 -73
                       -72 -71 -71 -70 -67]
mac00_15_62_51_f0_90: [-58 -58 -58 -58 -58
                       -58 -58 -57 -57 -57]
mac00_15_62_51_f3_d0: [-82 -80 -80 -80 -79
                       -79 -77 -76 -75 -73]
mac00_15_62_51_f4_90: [-84 -83 -81 -80 -79
                       -78 -78 -77 -77 -70]
```

Table 4: Example with results after uploading the Training cell(0_0) in Command window in Matlab

4.3.2 Uploading Training and Runtime data

In order to upload the training set and store it to an array (training fingerprint), we took into consideration the fact that the cells are in the form "cellx_y" with "x=0:2:12 and y=0:2:22". Thus, in the MATLAB script "radar.m" for RADAR ("main.m" for Percentiles method respectively) each cell from training dataset is stored in "trainingfn" array which includes the data structures, each of them corresponding to one cell. As we saw previously, each data structure comes as output from the function "load_cell" for RADAR (and "percentiles" for Percentiles method respectively) for each cell through the following for-loops:

```
for x = 0:2:12
    for y = 0:2:22
        trainingfn(x+1,y+1) = {load_cell('training', x, y)};
    end
end
```

Table 5: Uploading Training data and creating Training fingerprint (trainingfn array)

It is worth mentioning that in order to avoid the delay after running the main algorithm and import the data in the training fingerprint a lot of times for testing purposes, we created for both algorithms a .mat file (cells.mat and cells2.mat for RADAR and Percentiles respectively).

However, the cells for runtime dataset are not similarly increasing, for this reason we used the variables "xrun" and "yrun" to store the coordinates x and y for each of the 35 runtime cells. Here, we should notice that the runtime fingerprint is not an array of data structures but since it corresponds to one specific cell, it is a data structure.

```
xrun=[0 0 0 0 1 1 2 2 3 4 4 4 5 5 6 6 6 6 6 7 7 8 8 8 8 9 9 10 10 11
11 12 12 13 13];


yrun=[0 8 16 20 2 11 4 14 11 4 14 18 10 21 0 3 14 15 23 5 9 6 12 16 20
1 11 18 22 0 15 4 12 9 17];

for z = 1:length(xrun)
[…]
    runtimefn = load_cell('runtime_quiet', xrun(z),  yrun(z));
[…]
end
```

Table 6: Uploading Runtime data and creating Runtime fingerprint (runtimefn array)

4.3.3 Comparing the data

Afterwards, the algorithm compares the runtime and training fingerprint in order to infer users' location based on Euclidean distance and Percentiles. The runtime values are compared with these of training dataset and the cell containing the most "similar" values is reported as the estimated position.

The concept was the same for both cases; for each of the 35 runtime cells with coordinates (*xrun,yrun*):

- Initiate *mindist*, *minx* and *miny* variables, indicating the estimated cell that will result with minimum Euclidean distance or minimum weight for Percentiles.
- Upload the specific runtime cell in the respective structure "runtimefn"
- Compare the two fingerprints e.g. *trainingfn* and *runtimefn* that are the arguments of another function "*my_distance*" with output a number "dist" that indicates the Euclidian distance or the weight of the cell (see next paragraph for more information) between the runtime cell and the current training cell
- Check if the calculated distance is minimum from the current minimum distance (e.g. variable *mindist*); if yes replace the current value of *mindist*
- Store as *minx* and *miny* the coordinates that correspond to the training cell with minimum distance
- Repeat the above steps until the runtime cell will be compared with all the cells from training dataset and result to an estimated position
- Calculate the location error and store it to the array *error_i*

Regarding the function "*my_distance*", as we mentioned, it gets the two structures as arguments for their comparison. The first we had to notice is the fact that the runtime cell should have the same APs with all the training cells and vice versa. For this reason, we assigned very small (-100) signal strength values to the APs that are not present in the current cell from training fingerprint and was not included in the runtime fingerprint (and vice versa).

Afterwards, since the two structures have the same APs, we calculate for:

- RADAR the Euclidian distance of the average RSSI values for the i^{th} AP according to the formula:

$$d = \sqrt{\sum_{i=1}^{n} (R_i - T_i)^2}$$

- Percentiles, the weight of the cell with the difference that now we do not have as value in the structure a number (avgSS) but an array of j=10 percentiles, e.g.

$$w(c) = \sum_{i} \sqrt{\sum_{j} \left(R_{i,j} - T_{i,j}\right)^2}$$

General Instructions for running the algorithms:

In the MATLAB code, you may run the script:

- radar.m for RADAR algorithm and
- main.m for Percentiles method

which calls the other two functions that we mentioned e.g. “load_cell/percentiles” and “my-distance” for observing the implementation results.

4.4 Simulation Results and Comparisons

4.4.1 Estimated position and location error

In the table below, you may see the estimated position after running both algorithms at MATLAB and the location error as well.

	Real cell (xrun, yrun)	Est. Position RADAR (minx, miny)	Est. Position Percentiles (minx, miny)	Location Error RADAR	Location Error Per-centiles	Difference
1	[0,0]	[0,2]	[0,2]	2	2	0
2	[0,8]	[2,6]	[2,6]	2,83	2,83	0
3	[0,16]	[0,18]	[0,16]	2	0	-2
4	[0,20]	[0,22]	[2,20]	2	2	0
5	[1,2]	[2,0]	[2,4]	2,24	2,24	0
6	[1,11]	[2,10]	[2,10]	1,41	1,41	0
7	[2,4]	[0,2]	[0,2]	2,83	2,83	0
8	[2,14]	[2,14]	[2,14]	0	0	0
9	[3,11]	[2,10]	[2,10]	1,41	1,41	0
10	[4,4]	[2,2]	[2,2]	2,83	2,83	0
11	[4,14]	[4,14]	[4,14]	0	0	0
12	**[4,18]**	**[12,18]**	**[12,18]**	**8**	**8**	**0**
13	[5,10]	[12,10]	[12,10]	7	7	0
14	[5,21]	[4,20]	[4,20]	1,41	1,41	0
15	[6,0]	[8,2]	[8,2]	2,83	2,83	0
16	[6,3]	[4,4]	[4,4]	2,24	2,24	0
17	[6,14]	[4,14]	[4,14]	2	2	0
18	[6,15]	[12,14]	[12,8]	6,08	9,22	3,14
19	[6,23]	[4,22]	[4,22]	2,24	2,24	0
20	[7,5]	[8,4]	[8,4]	1,41	1,41	0
21	[7,9]	[8,8]	[10,12]	1,41	4,24	2,83
22	[8,6]	[10,4]	[8,6]	2,83	0	-2,83
23	[8,12]	[6,12]	[6,12]	2	2	0
24	**[8,16]**	**[8,12]**	**[6,8]**	**4**	**8,25**	**4,25**
25	[8,20]	[10,22]	[10,22]	2,83	2,83	0
26	[9,1]	[8,4]	[8,4]	3,16	3,16	0
27	[9,11]	[8,12]	[8,12]	1,41	1,41	0
28	[10,18]	[6,18]	[6,18]	4	4	0
29	[10,22]	[10,22]	[12,18]	0	4,47	4,47
30	[11,0]	[12,0]	[12,0]	1	1	0
31	[11,15]	[6,14]	[10,16]	5,1	1,41	-3,69
32	[12,4]	[12,4]	[12,4]	0	0	0
33	[12,12]	[6,8]	[8,12]	7,21	4	-3,21
34	[13,9]	[12,10]	[12,10]	1,41	1,41	0
35	[13,17]	[12,14]	[10,16]	3,16	3,16	0

Table 7: Results for RADAR and Percentiles

In the table above, there is an analysis of the estimations and location errors for both algorithms. In general, 69% of the estimated results are the same for both methods regardless if they are accurate enough or not, which means that RADAR and Percentiles are equivalent enough.

At a first glance, these total results indicate that despite the fact that the algorithm for Percentiles seems more efficient to identify the real cell, the RADAR algorithm tends to provide better estimations when there is a remarkable deviation between real and estimated positions (see Table 7).

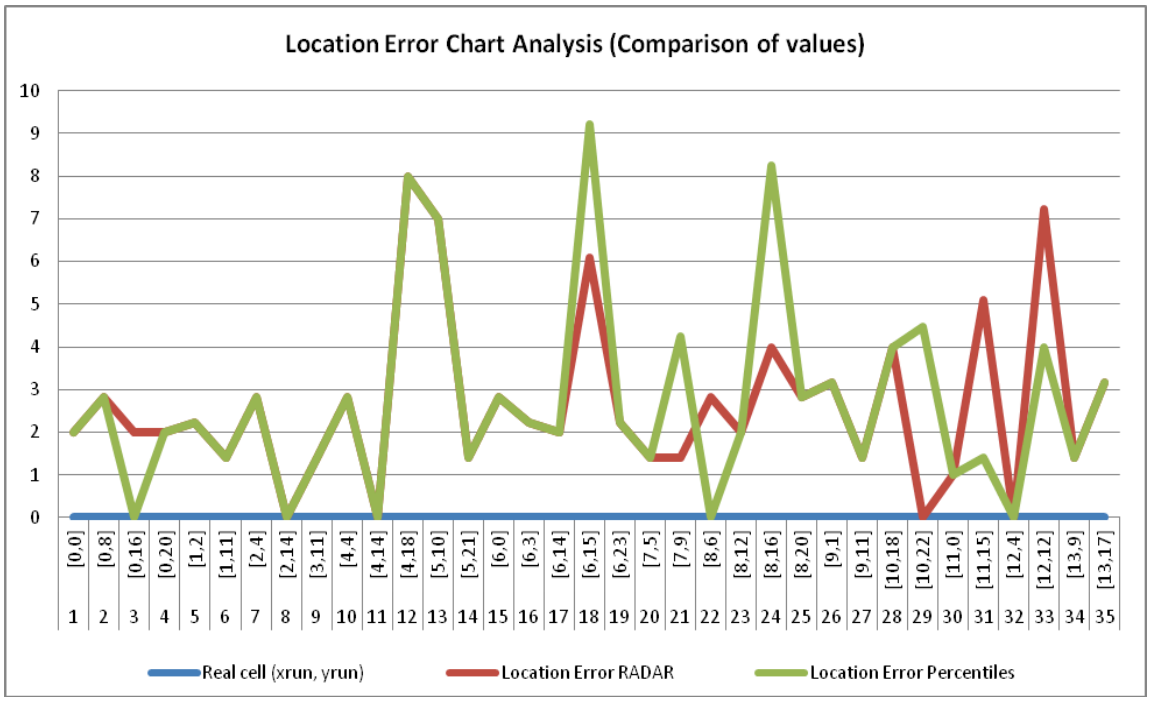


Figure 11: Graphical Representation of Table 8

From the table above, the 12th case is the biggest failure for both algorithms with location error 8 (and 13 as well with 7 location error). [4,18] real position, RADAR and Percentiles [12,18]

The 24th case is the worst for Percentiles comparing with RADAR with 4.25 difference in their location error. 24 case [8,16] is the real position, RADAR, [8,12], Percentiles [6,8]

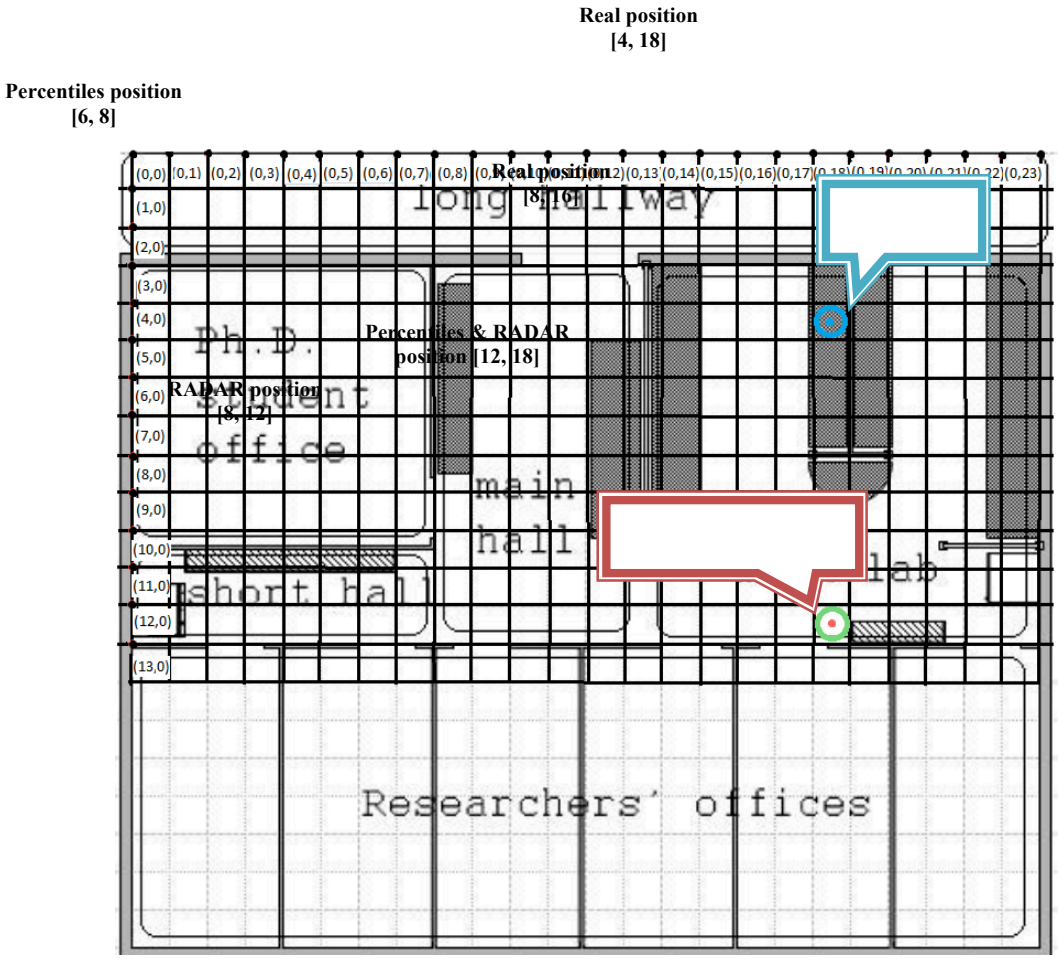


Figure 12: Grid Representation of 12th case, [4,18] real position, RADAR and Percentiles [12,18]

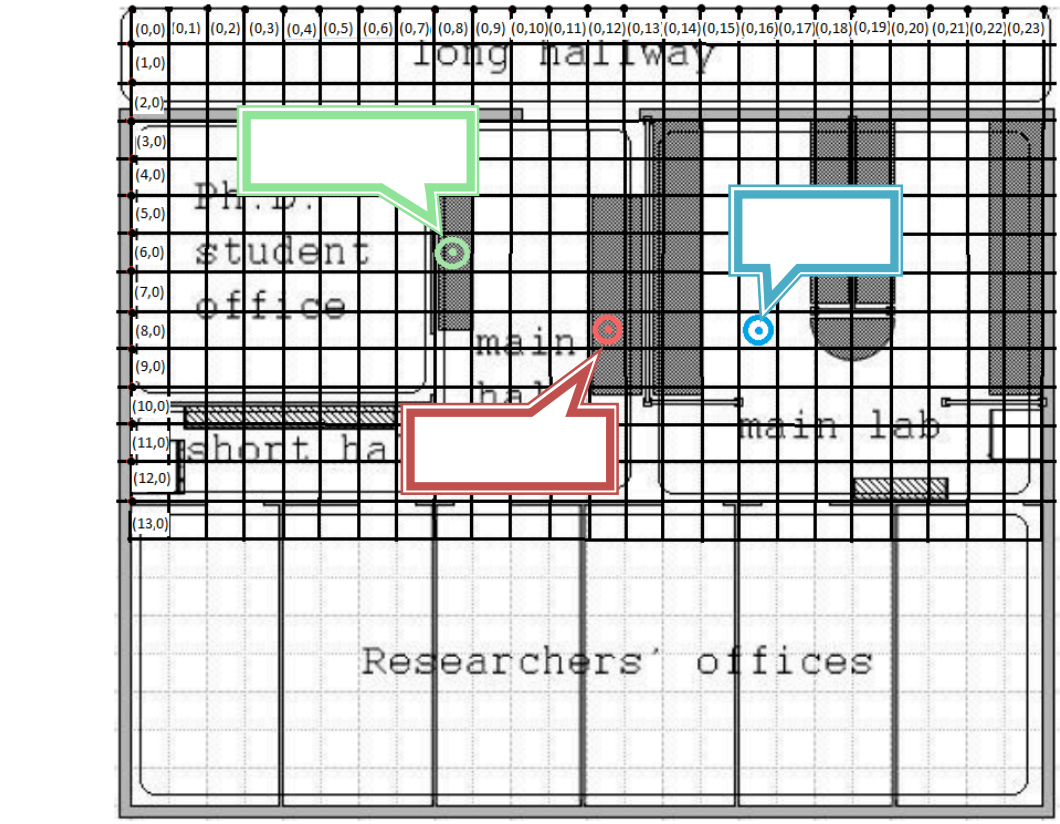
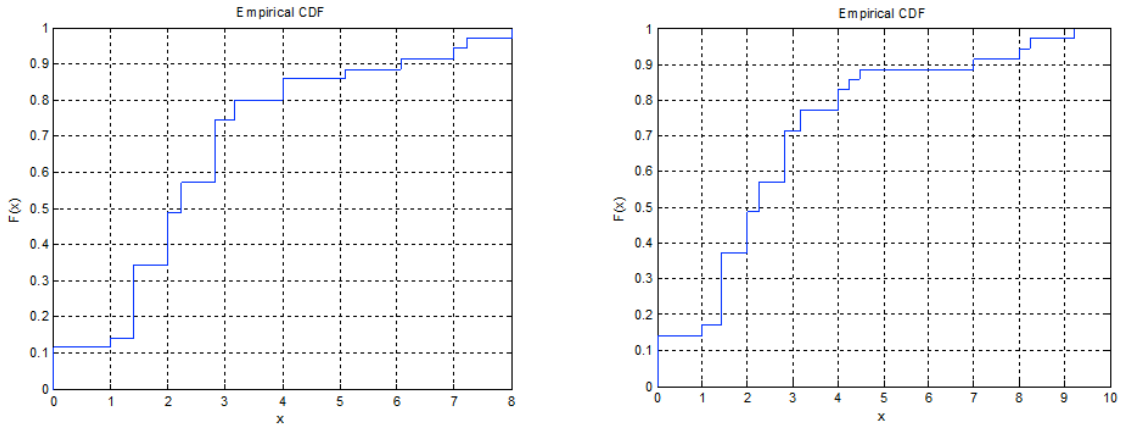


Figure 13: Grid Representation of 24th case, [8,12], Percentiles [6,8]

4.4.2 Error Analysis

The Error Analysis was conducted by computing the location error as the Euclidian distance between the real cell and the estimated cell. The CDF and the median error are reported, too. For both cases, after running 35 measurements at the different positions (runtime cells), the following figures illustrate the location error of the RADAR approach during runtime period. As it is distinguished from the Cumulative probability Distribution Functions (which are almost the same) and confirmed with the calculation of the median error, the median error is 2.24m (same for both cases).

"The cumulative probability functions (CDF) of the distance error is used for measuring the precision of a system. Since these two positioning techniques are compared, the system with CDF graph that reaches high probability values faster is considered more precise because its distance error is concentrated in small values. In practice, CDF is described by the percentile format. For example, in both cases, the system has a location precision of 50% within 2.24 m (the CDF of distance error of 2.4 m is 0.5), and 95% within 7 m." [19]



(a): Cumulative Distribution Function for RADAR *(b): Cumulative Distribution Function for Percentiles*

Figure 14: The performance of fingerprint positioning methods at FORTH.

5 Conclusions

In this project, we have presented the taxonomized location system properties, the basic techniques used for location sensing as well as a surveyed research/related work in the specific field. Afterwards, we analyzed some already existing algorithms and described their framework. We discussed their simulation results and evaluated their performance (according to [8]) after calculating their median error. From all signature-based approaches - confidence intervals, percentiles, empirical distribution and multivariate Gaussian model (MvGs) – the last one performs much better than percentiles, confidence interval (90%) and empirical distribution approaches. Furthermore, the number of APs has an impact on location error since "*the higher the number of APs, the lower the location error"*. Also, *"the larger the measurement set, the more accurate the position estimation"*.

Finally, we applied the method of Radar algorithm and percentiles in order to upload both training and runtime fingerprint based on common APs, calculate the Euclidian distance/formula for each cell and return the estimated cell with the minimum coordinates. For each of the 35 runtime cells, we calculated the location error for each of the 35 comparisons between Runtime and Training. Two comments that we could summarize are that percentiles and RADAR perform almost the same, although RADAR algorithm tends to provide better estimations when there is a remarkable deviation between real and estimated positions. It is worth mentioning that the fact that we had only 10 APs, is an ideal situation that does not happen in reality because of the existence of noise.

Bibliography

[1] Papadopouli M.: *Lectures on Positioning, Wireless Networks and Mobile Computin.* Department of Computer Science, University of Crete

[2] Hightower J., Borriello G.: *A Survey and Taxonomy of Location Sensing Systems for Ubiquitous Computing*. Technical Report, University of Washington, Department of Computer Science and Engineering UW CSE 01-08-03, Seattle, WA, Aug. 2001.

[3] Papadopouli M., Henning Schulzrinne: *Peer-to-Peer Computing for Mobile Networks: Information Discovery and Dissemination*. Springer, July 14, 2008

[4] Moustafa Youssef, Matthew Mah, Ashok Agrawala: *Challenges: Device-free Passive Localization for Wireless Environments.* Dept. of Computer Science, University of Maryland

[5] Paramvir Bahl, Venkata N. Padmanabhan, Anand Balachandran: *Enhancements to the RADAR User Location and Tracking System.* Microsoft Research, University of California at San Diego

[6] Andreas Haeberlen, Eliot Flannery, Andrew M. Ladd: *Practical Robust Localization over Large-Scale 802.11 Wireless Networks*. Rice University

[7] Mobile Innovation Center: *Lectures on Mobile Telecommunications*

[8] Milioris D., Kriara L., Papakonstantinou A., Tzagkarakis G., Tsakalides P., Papadopouli M.: *Empirical Evaluation of Signal-Strength Fingerprint Positioning in Wireless LANs*. Computer Science Department, University of Crete and Institute of Computer Science, Foundation for Research and Technology-Hellas (FORTH)

[9] Steve McDonnell: How to Calculate the Average RSSI for an 802.11. http://www.ehow.com/how_7823781_calculate-average-rssi-80211.html

[10] Lito Kriara: *Experimenting with the fingerprinting method using signal-based measurements for providing positioning information to location-based applications.* MSc. Thesis. Department of Computer Science, University of Crete, July 2009

[11] Standards, IEEE 802.11: *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*.

[12] Nissanka B. Priyantha, Anit Chakraborty, Hari Balakrishnan: *The Cricket location-support system*. In ACM International Conference on Mobile Computing and Networking (MobiCom), pages 32–43, Boston, MA, USA, August 2000.

[13] Nissanka B. Priyantha, Allen K. L. Miu, Hari Balakrishnan, Seth Teller: *The Cricket compass for context-aware mobile applications*. In ACM International Conference on Mobile Computing and Networking (MobiCom), pages 1–14, Rome, Italy, July 2001.

[14] 307. R. Harle, A. Ward, A. Hopper: *Single reflection spatial voting*. In Proceedings of the First International Conference on Mobile Systems, Applications and Services, San Francisco, May 2003.

[15] Session 1: *Introduction to GPS and GMT resources*.
http://mygeologypage.ucdavis.edu/gps/EDUCATION/5_SESSION/SESSION1/session1_intro.html

[16] Roy Want, Andy Hopper, Veronica Falcao, Jon Gibbons: *The Active Badge Location System.* ACM Transactions on Information Systems, 10(1):91–102, January 1992.

[17] Elena Deza, Michel Marie Deza: *Encyclopedia of Distances*, page 94, Springer 2009.

[18] http://www.mathworks.com/help/stats/prctile.html

[19] Hui Liu, Houshang Darabi, Pat Banerjee, Jing Liu: *Survey of Wireless Indoor Positioning Techniques and Systems*. IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews 2007. 37(6):1067