Experimenting with the fingerprinting method using signal-based measurements for providing positioning information to location-based applications

MSc. Thesis

Lito Kriara

Heraklion
July 2009
Experimenting with the fingerprinting method using signal-based measurements for providing positioning information to location-based applications

Submitted to the Department of Computer Science by
Lito Kriara
in partial fulfillment of the requirements for the degree of Master of Science

Author: __________________________
Lito Kriara

Board of enquiry:

Supervisor: __________________________
Dimitris Plexousakis
Professor

Member: __________________________
Grigoris Antoniou
Professor

Member: __________________________
Maria Papadopouli
Assistant Professor

Accepted by: __________________________
Panos Trahanias
Professor
Chairman of the Graduate Studies Committee

July 2009
Heraklion, Greece
Abstract

Fingerprinting is widely used for positioning algorithms (e.g. [3, 30, 44]). Fingerprints can be generated using statistical information of the collected signal-strength values from various wireless devices, such that IEEE 802.11, RFIDs and IrDAs. Positioning systems compare fingerprints obtained at run-time (at an unknown position) with the ones collected during a training phase. In the context of the proposed positioning algorithm, we experimented with confidence intervals and percentiles. We tested various fingerprinting methods at the Cretaquarium and the Institute of Computer Science in Foundation for Research and Technology–Hellas. These methods were also used in a location-sensing system (CLS) for supporting various location-based applications. We experimented with various, as well, technologies (i.e., IEEE 802.11, RFID, IrDA). Accuracy is the property of the smallest distance a system can report. We found that the larger and more detailed the fingerprint the higher the accuracy. For example, the median location error using only IEEE 802.11 in testbed of ICS is 1.1 m. Finally, the incorporation of extra technologies (i.e., IrDA) along with IEEE 802.11 can improve the accuracy of the proposed algorithms. Specifically, using both IEEE 802.11 and IrDA technology the median location error is 0.5 m.
Acknowledgements

I would really like to thank Prof. Maria Papadopouli for her guidance during this work, and Prof. Dimitris Plexousakis, for his support and help. I sincerely thank Giorgos Tzagarakis, Sofia Nikitaki, Antonis Bikakis and Thodoris Patkos with whom I have worked through these years and their help, advices and support were essential for completeness of this work. Moreover, I have to mention that all the members of the Telecommunications & Networks Laboratory and Information Systems Laboratory at the Institute of Computer Science (ICS) of the Foundation for Research and Technology Hellas (FO.R.T.H.), and my non-FORTH friends were always available for my questions, and I want to thank them for their continuous encouragement, support and love. Finally, I am sincerely indebted to my family for always believing in me and being there for me.
Contents

1 Introduction ............................................. 1
   1.1 Challenges ........................................ 1
   1.2 Motivation ........................................ 2
   1.3 Thesis statement .................................. 2
   1.4 Thesis outline .................................... 3

2 Background .............................................. 4
   2.1 Classification of Location-sensing Systems ........... 4
      2.1.1 Hardware requirements ......................... 4
      2.1.2 Description of position ......................... 6
      2.1.3 Training requirements ......................... 6
      2.1.4 Methodology .................................. 7
   2.2 IEEE 802.11 ...................................... 10
   2.3 Related Work ..................................... 11
   2.4 CLS ............................................... 17
      2.4.1 Generation of fingerprints .................... 18
      2.4.2 p2p-based .................................... 21
      2.4.3 Particle filter-based ......................... 21
   2.5 C-NGINE .......................................... 25

3 Experimental results ................................. 32
   3.1 Benchmarks ....................................... 33
   3.2 Testbed description ................................ 33
      3.2.1 Cretaquarium ................................. 34
3.2.2 TNL ................................. 42
3.2.3 ICS .................................. 45
3.3 Impact of number of APs .................................. 47
  3.3.1 Removing most common APs ......................... 47
  3.3.2 Removing least common APs ......................... 48
  3.3.3 Removing random APs ............................... 51
3.4 Statistically improving accuracy ......................... 51
  3.4.1 Removing Outliers ................................. 53
  3.4.2 Incorporating conditional probability .............. 55
  3.4.3 Assigning weights to APs ........................... 57
  3.4.4 Kullback–Leibler divergence ....................... 60
3.5 Conclusions ........................................ 63

4 Adding extra technologies to IEEE 802.11 .................. 68
  4.1 RFID ........................................ 68
    4.1.1 Technology description ........................... 68
    4.1.2 Performance analysis ............................ 75
  4.2 IrDA ......................................... 83
    4.2.1 Technology description ........................... 84
    4.2.2 Performance analysis ............................ 85
  4.3 Conclusions ...................................... 91

5 Conclusions .......................................... 94
List of Tables

3.1 Definition of symbols used in conditional probability formula. . . . . . . . . 55
3.2 Implementations of CLS. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
3.3 Median performance of each algorithm in each testbed. . . . . . . . . . . . 64
3.4 Impact of mean percentage of APs covering a cell of the grid representation
    of each testbed. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65
3.5 Impact of number of peers on the p2p-based method, median location error,
    under extreme conditions. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
4.1 Differences between technologies. . . . . . . . . . . . . . . . . . . . . . . . . . 92
List of Figures

2.1 Location-sensing based on signal-strength. ................................. 8
2.2 Location-sensing based on Angle of Arrival (AoA). ......................... 9
2.3 Location-sensing based on Time of Arrival (ToA). .......................... 10
2.4 Location-sensing based on Time Difference of Arrival (TDoA). .......... 10
2.5 State space model for the proposed location-sensing system. Clear circles indicate hidden state variables, grayed circles indicate observations, horizontal arrows indicate state transition functions and vertical arrows indicate observation functions. ................................................................. 22
2.6 C-NGINE Architecture. ................................................................. 27
3.1 SS coverage from all APs in the testbed of Cretaquarium. ............... 35
3.2 Location error results in the aquarium using the confidence interval-based algorithm with different confidence interval accuracy levels, under normal conditions. (Set 1) ................................................................. 36
3.3 Location error results in the aquarium comparing confidence interval-, deciles- and quartile-based methods, under normal conditions. (Set 1) .................. 37
3.4 Location error results in the aquarium comparing confidence interval-, deciles-, quartile- and particle filter-based methods, under normal conditions. (Set 1) ................................................................. 38
3.5 Location error results for 45 iterations of the particle filters-based algorithm for only one cell of the grid (same input arguments). (Set 1) ................. 39
3.6 Location error results comparing confidence interval- to p2p-based method, under extreme conditions. (Set 2) ................................................................. 39
3.7 Location error results comparing the distance between entries of p2p training set. (Set 2) ......................................................... 40
3.8 Division of Cretaquarium testbed into zones. .......................... 41
3.9 Accuracy results, CLS vs Ekahau in Cretaquarium under extreme conditions. (Set 3) ......................................................... 42
3.10 Grid-based representation of TNL testbed. .............................. 43
3.11 Location error results in the TNL using the confidence interval-, deciles-, quartile- and particle filter-based algorithm. (Set 4) ............... 43
3.12 Location error results for 100 iterations of the particle filters-based algorithm for one cell of the grid (same input arguments). (Set 4) ....... 44
3.13 Location error results comparing confidence interval- to p2p-based. (Set 5) 45
3.14 Grid-based representation of ICS testbed. .............................. 46
3.15 Performance of various implementations in ICS. (Set 6) ............... 47
3.16 Percentage of coverage of the whole testbed, of APs in ICS. (Set 6) .... 48
3.17 Variation of performance of decile-based algorithm while eliminating most common APs in ICS. (Set 6) ......................................................... 49
3.18 Median and 90% percentile of location error of decile-based algorithm while eliminating most common APs in ICS. (Set 6) ............... 49
3.19 Variation of performance of decile-based algorithm while eliminating least common APs in ICS. (Set 6) ......................................................... 50
3.20 Median and 90% percentile of location error of decile-based algorithm while eliminating least common APs in ICS. (Set 6) ............... 50
3.21 Variation of performance of decile-based algorithm while eliminating random APs in ICS. (Set 6) ......................................................... 51
3.22 Median and 90% percentile of location error of decile-based algorithm while eliminating random APs in ICS. (Set 6) ......................... 52
3.23 Location error results comparing removing outlier methods efficiency, under extreme conditions. (Set 1) ......................................................... 54
3.24 Efficiency of confidence interval- and decile-based methods with and without use of conditional probability (normal condition). (Set 7) ................. 56
### List of Figures

3.25 Efficiency of confidence interval- and decile-based methods with and without use of conditional probability (extreme condition). (Set 5) 57

3.26 Location error results comparing the impact of each AP on the accuracy. (Set 8) 58

3.27 Location error results comparing the impact of each AP on the accuracy in the Obstacle area. (Set 8) 58

3.28 Location error results comparing the impact of each AP on the accuracy in the No Obstacle area. (Set 8) 59

3.29 Efficiency of confidence interval-based method with and without use of weights (extreme condition). (Set 8) 60

3.30 Comparing KLD-based approach to the confidence interval-, decile- and quartile-based algorithms. (Set 1) 62

3.31 Comparing KLD- and KLD symmetric-based approaches. (Set 1) 63

4.1 Communication between RFID reader and tag. 71

4.2 RF Code M200 reader device. 72

4.3 Placement of RFID readers to discriminate three labs. 74

4.4 Experiments number 1, 2 and 3 in ICS. 76

4.5 RFID technology deployment in TNL. 77

4.6 Location error results comparing RFID-based in different areas of the TNL. (Set 9) 78

4.7 Location error results comparing RFID- to WiFi- or WiFi&RFID-based in TNL. (Set 9) 80

4.8 Illustration of ICS and the range of three RFID readers. (Set 6) 81

4.9 Location error results comparing WiFi- to WiFi&RFID-enabled on path positions (Scenario 1). (Set 6) 82

4.10 Location error results comparing WiFi- to WiFi&RFID-enabled on path positions (Scenario 2). (Set 10) 83

4.11 One board IrDA beacon with six LEDs, horizontal mounted to the beacon board. 85
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.12</td>
<td>Comparing location error results of WiFi&amp;IrDA- and WiFi-enabled – Scenario 1. (Set 6)</td>
<td>87</td>
</tr>
<tr>
<td>4.13</td>
<td>Comparing location error results of WiFi&amp;IrDA- and WiFi-enabled – Scenario 2. (Set 10)</td>
<td>88</td>
</tr>
<tr>
<td>4.14</td>
<td>Comparing location error results of WiFi&amp;IrDA(7, 12)- and WiFi&amp;IrDA(6.5)-enabled – Scenario 1. (Set 6)</td>
<td>89</td>
</tr>
<tr>
<td>4.15</td>
<td>Comparing location error results of WiFi&amp;IrDA(7, 12)- and WiFi&amp;IrDA(6.5)-enabled – Scenario 2. (Set 10)</td>
<td>89</td>
</tr>
<tr>
<td>4.16</td>
<td>ICS testbed with IrDA transceivers’ coverage</td>
<td>90</td>
</tr>
<tr>
<td>4.17</td>
<td>WiFi&amp;IrDA-enabled CLS real-time results. (Set 11)</td>
<td>91</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Pervasive computing aims to create embedded systems in the environment, as much invisible and constantly available enhancing the information access without distracting the users from their main tasks as possible. Wearable computers, sensors in cars and smart houses are just a few examples of technologies expected to prevail in pervasive computing environments in order to make our lives easier. Deployment of pervasive computing was highly effected and motivated by wireless communication.

The common feature of most of pervasive computing enabled devices is that they need to have a-priori knowledge of their position in space to support different location dependent applications. This determination of the physical position is known as localization or location-sensing.

1.1 Challenges

Early location-sensing algorithms were based on signal-strength measurements and simple triangulation methods exclusively. However, these methods have limited capabilities, especially because of the interference and dynamic characteristics of the radio propagation in various environments. Each environment may have a different behavior, due to the interference, materials used and radio propagation. Therefore, all these aspects impose important challenges for the design of location-sensing methods that will be easily deployable, and computationally inexpensive.

Our main goal is to design and implement algorithms that can overcome all those
problems and have high accuracy at the same time.

1.2 Motivation

Our main motivation is to design and implement algorithms, incorporating various statistical methods, that can achieve high accuracy regardless the physiology of space and the scattering and similar phenomena that IEEE 802.11 technology poses.

We aim to propose a positioning method using wireless infrastructure and a probabilistic framework to estimate the position of wireless-enabled devices in an iterative manner without the need for an extensive infrastructure or time-strenuous training.

1.3 Thesis statement

The main positioning algorithm tested adopts a grid-based representation of the physical space; each cell of the grid corresponds to a physical position of the physical space. The cell size reflects the spatial granularity/scale. Each cell of the grid 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, through which votes are casted on cells of the grid. A vote on a cell indicates the likelihood that the local device is located in the corresponding area of that cell.

In this thesis, we implemented an algorithm in C and C++ in order for a positioning system, that will use it, to be able and collect signal-strength values, either at run-time or not, and then compute an estimated position for the user. We computed the range error (i.e., distance estimation error) based on real-life signal-strength measurements that had taken place in the Institute of Computer Science (ICS) of the Foundation for Research and Technology - Hellas (FORTH), in the Telecommunication and Networks Laboratory (TNL) of ICS-FORTH, as well as in the Cretaquarium [8] in Hersonissos - Heraklion.

We used monitoring tools to collect signal-strength measurements that were transformed in different statistical criteria depending the algorithm used, in the context of this thesis. This was done in order to be able and compare the results of all criteria and algorithms, with the same input data. Thus, the same procedure can be used for run-time
scenarios, as well. We experimented with the impact of several parameters, such as the statistical information used, the range error, the physiology of the testbed and the number and placement of APs on the accuracy of the position estimation, through several scenarios.

On the following we added extra technologies to the main IEEE 802.11-enabled methods. In particular, to enhance its accuracy, we extended the fingerprinting method by incorporating additional information obtained by non IEEE 802.11 sources, such as RFID and IrDA.

To summarize, the main contributions of this thesis are the extensive empirical studies of the proposed fingerprinting methods in real environments. For example, we found that:

1. The larger and more detailed the fingerprint the higher the accuracy.
2. Median location error using only IEEE 802.11 in testbed of ICS is 1.1 m.
3. The incorporation of extra technologies (i.e., IrDA) along with IEEE 802.11 can improve the accuracy of the proposed algorithms.

1.4 Thesis outline

The present thesis is organized as follows: Chapter 2 presents a definition of the location-sensing systems and a basic classification of these systems, a description of the most representative location-sensing systems of the location-sensing computing up to this moment, and an overview of CLS [44], along with its implementations. In Chapter 3 the testbeds are described as well as the main experimenting results of the algorithms presented before and various methods used to improve the systems accuracy. Chapter 4 analyzes the extra technologies added to the system apart from IEEE 802.11 originally used. Finally Chapter 5 presents a comparative study between the proposed algorithms and others already existing, and the conclusions and future work, respectively.
Chapter 2

Background

In this Chapter, we analyze the basic functions of location-sensing systems and we present a classification of these systems based on the location-sensing properties. In particular, we classify the location-sensing systems based on the type of metric, use of hardware, and description of the position. Finally, we describe the technologies usually used in location-sensing systems like IEEE 802.11, RFID and IrDA.

2.1 Classification of Location-sensing Systems

Different dependencies and using special techniques and technologies can classify location-sensing systems depending on the infrastructure and specialized hardware used, signal modalities, training, methodology and models for distances estimation, incorporating orientation, and position, coordination system, location description, localized or remote computation, cost, privacy, and accuracy and precision requirements. The distance can be computed using time of arrival (e.g., PinPoint [49]) if the velocity of the signal is known, or signal-strength measurements if a signal attenuation model for the given environment is known.

2.1.1 Hardware requirements

Location-sensing systems can be classified in several categories based on their properties. Firstly, we classify the location-sensing systems, regarding the use of hardware.
There are various systems depending on infrastructure and other that do not. In the first case they take advantage of their environmental infrastructure. For instance, a system can receive positioning information from an AP which is wired connected with the Internet, Bluetooth or IrDA devices, that either already exist in the environment or they are specifically placed for that reason. On the other hand, a dense deployment of a wireless infrastructure for communication and location-sensing may not be feasible due to environmental, cost, and regulatory barriers. Ad hoc networks exploit cooperation by enabling devices to share positioning estimates [39]. They are infrastructureless, self-configuring networks of mobile hosts connected by wireless links, forming an arbitrary topology. The hosts are free to move randomly and organize themselves arbitrarily. In a purely ad-hoc location-sensing system, all of the entities estimate their locations by cooperating with other nearby objects. In this way, objects of a cluster of the ad-hoc network are located relative to one another or absolutely if some objects in the cluster occupy known locations.

To infer the position, location-sensing systems devices may employ different modalities, such as radio [3, 42, 9], infrared [47], ultrasonic [34, 35], Bluetooth [17, 15, 36, 5, 1], 4G, or vision [26], while others physical contact with pressure, touch sensors or capacitive detectors. The wide popularity of the IEEE 802.11 network, low deployment cost, and advantages of using it for both communication and positioning, make it an attractive choice.

This makes the first category a more realistic phenomenon since companies and institutes are based on infrastructure, usually IEEE 802.11 (APs). On the other hand, the ad-hoc approach is still under research but it has not been applied in a real-life testbed.

Moreover in the first case mentioned above, there are location-sensing systems that depend on specialized hardware (i.e., tags, cameras, ultrasound receivers) to locate a wireless device. However, there are others that have no need of specialized hardware, but they are based on existing infrastructure, such as the IEEE 802.11 infrastructure.

The location of the infrastructure used is another classification of location-sensing systems. It can be either terrestrial or satellite. In the former case APs and sensors placed on the ground or on ceilings are included, whereas in the latter one satellites are used like

A final classification concerning the hardware 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 or RADAR), whereas remotely means that a device (i.e., tag) does the collection and then sends the measurements to the actual device (i.e., laptop, PDA) that is going to compute the position of the user.

2.1.2 Description of position

Location-sensing systems can be classified regarding the description of the position. Some systems can be characterized as either physical, if the returned position corresponds to a physical coordinate system (e.g., global coordinates), or symbolic, if they provide information that employs textual description of a location, and place or geographic symbolic coordinates (i.e., coordinates of a grid-representation of the physical space).

In both cases, though, the information provided can be either absolute or relative. Absolute position implies a location system that employs a grid for all located objects, for example latitude, longitude and altitude. On the contrary, a location system that uses relative positions can have a distinct frame of reference, for example 2 m from a specific AP.

Another very important aspect of location-sensing systems is the distinction between accuracy and precision. Accuracy is the property of the smallest distance a system can report. Precision of a system is the percentage of the times the prescribed accuracy is reported. The main goal of all location-sensing systems is to achieve both high accuracy and high precision, in order for them to be considered robust.

2.1.3 Training requirements

Most of the signal-strength based localization systems can be classified into the following two categories, namely the signature or map-based and the distance-prediction based.

The first type creates a signal-strength signature or map of the physical space during a training phase and compares it with analogous run-time measurements [3, 30, 48]. To
build such maps, signal-strength data is gathered from beacons received from APs at various predefined checkpoints during a training phase. Thus, each checkpoint in the map associates the corresponding position of the physical space with statistical measurements based on signal-strength values acquired at those positions. Such maps can be extended with data from different sources or signal modalities, such as ultrasound from deployed sensors to improve location-sensing [17].

In other situations, a dense deployment of a wireless infrastructure for communication and location-sensing may not be feasible due to environmental, cost, and regulatory barriers. Ad hoc networks are used in that case, because they exploit cooperation by enabling devices to share positioning estimates [14, 7, 49].

2.1.4 Methodology

The methodology used by each positioning system can divide them certain categories based on which of the following methods they use:

1. Received signal-strength indication (RSSI)

2. Direction resolve (Angle of Arrival)

3. Distance resolve (Time of Arrival, Time Difference of Arrival)

The signals used in the localization process in each of the above systems, could be either radio frequencies (RF), infrared (IR) or ultrasound.

The Received signal-strength indication (RSSI) is a measurement of the strength (not necessarily the quality) of the received signal in a wireless network, in arbitrary units, depending on the hardware (i.e., wireless card) used. Location-sensing using the received signal-strength, also uses a known mathematical model which describes the path loss attenuation of the signal with distance. The measurement of signal-strength 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 distance can be measured by calculating the path loss. The path loss can be found if the mobile object knows the power transmitted from the base station and
the received power. If we use three base stations or more, the position of the mobile object can be determined using the technique of triangulation. The errors due to shadow fading can be eliminated with the use of pre-measured signal-strength (training) maps each time centered at a different base station. This describes the changes of the signal-strength for a given space. However, this presupposes a stable topological environment and enough time-strenuous effort. Even in this case, though, the results can be different at another instance due to the variability of the environment.

Other systems are based on direction, as mentioned before. The systems based on the direction of a signal estimate the mobile object’s location by measuring the Angle of Arrival (AoA) of a signal from a mobile object at several base stations with the use of antenna arrays. The intersection of the lines that define the direction of the signal is the position of the mobile object.

Scattering between the mobile and base station will alter the measured AoA. If there is no line-of-sight (LOS), the antenna array will take a reflected signal that may not be coming from the direction of the mobile object. Even if a LOS component is present, multipath effects may still interfere with the angle measurement. The accuracy of the
Figure 2.2: Location-sensing based on Angle of Arrival (AoA).

AoA method diminishes with increasing distance between the mobile object and the base station due to fundamental limitations of the devices used to measure the arrival angles.

Finally, there are systems based on distance that estimate the position by measuring the absolute distance, or the difference of the distances, of the mobile object from the base stations. The distance between a mobile object and a base station is measured by finding the Time of Arrival (ToA) which is the one-way propagation time between them, assuming that the transmission time is known. Geometrically, this is depicted with a circle, with its center at the base station, on which the mobile object must lie.

By using at least three base stations to resolve ambiguities, the position of the mobile object is at the intersection of the circles. A basic precondition for the system to function is the absolute synchronization of the mobile object and the base station.

When this is not true, instead of the absolute times, the Time Difference of Arrival (TDoA) is used, that is, the differences of the times of arrival in every base station, since it is much easier for the base stations to be synchronized. Since the hyperbola is a curve of constant time difference of arrival for two base stations, the time differences define hyperbolae, on which the mobile objects must lie. Hence, the location of the mobile object
Figure 2.3: Location-sensing based on Time of Arrival (ToA).

Figure 2.4: Location-sensing based on Time Difference of Arrival (TDoA).

is at the intersection of the hyperbolae as shown below.

2.2 IEEE 802.11

IEEE 802.11 is a set of standards implementing wireless local area network (WLAN) computer communication in the 2.4, 3.6 and 5 GHz spectrum bands. The 802.11 family includes over-the-air modulation techniques that use the same basic protocol. The most popular are those defined by the 802.11b and 802.11g protocols, and are amendments to
the original standard. 802.11b and 802.11g use the 2.4 GHz ISM band, and because of this choice of frequency band, 802.11b and g equipment may occasionally suffer interference from microwave ovens and cordless telephones. Bluetooth devices, while operating in the same band, in theory do not interfere with 802.11b/g because they use a frequency hopping spread spectrum signaling method (FHSS) while 802.11b/g uses a direct sequence spread spectrum signaling method (DSSS).

The segment of the radio frequency spectrum used varies between countries. Frequencies used by channels one through six (802.11b) fall within the 2.4 GHz amateur radio band. 802.11g, that we use, works in the 2.4 GHz band (like 802.11b), but uses the same OFDM based transmission scheme as 802.11a. It operates at a maximum physical layer bit rate of 54 Mbit/s exclusive of forward error correction codes, or about 19 Mbit/s average throughput. 802.11g hardware is fully backwards compatible with 802.11b hardware and therefore is encumbered with legacy issues that reduce throughput when compared to 802.11a by 21%.

Like 802.11b, 802.11g devices suffer interference from other products operating in the 2.4 GHz band. Devices operating in the 2.4 GHz range include: microwave ovens, Bluetooth devices, baby monitors and cordless telephones.

2.3 Related Work

Over the last few years, significant research has been done in the area of location-sensing using signal-strength measurements. Bahl et al. [3] employ integrated signal-strength measurements into signal-strength maps. These measurements are acquired through training phase from APs at various different positions with specific coordinates. Each collected signal-strength vector is compared to the map and the coordinates of the best match will be reported as the estimated position of the user. The 90-th percentile of the location error is 6 m using one sample per 13.9 $m^2$ and 19.1 $m^2$ of the testbed and 3 and 5 APs, respectively.

Bahl et al. extended Radar to incorporate the dynamic changes of signal-strength nature, such as aliasing and multipath [2]. This extended version of Radar resulted in a
mean location error of 2.37 m and a 90% percentile of 5.97 m.

Unicycles and Baddy Nata [32] introduced a co-operative location-sensing system that propagates position information of landmarks towards distance hosts, while closer hosts enrich this information by determining their own location. Various methods were evaluated (i.e., “DV-hop”, “DV-distance”, “Euclidian”). Afterwards, the authors [33] incorporated specialized hardware to their algorithm, in order to estimate the angle between two hosts in an ad hoc network. They used antenna arrays or ultrasound receivers in order to implement this idea. Hosts gather data, compute their estimated positions, and expand them throughout the network.

Ladd et al. [30] proposed an algorithm that uses the IEEE 802.11 infrastructure. Firstly, a host compute the conditional probability of its location for a number of different locations by employing a probabilistic, based on the collecting signal-strength measurements from 9 APs. Secondly, the system exploits the mobility of users and its speed in order to refine the results and eliminate positions faraway of the mobile host. In the case that the second step is used, 83% of the time hosts can predict their location within 1.5 m, whereas this accuracy is achieved in 77% of the times in the case that the second step is not performed.

[6] is another location-sensing system using only ad hoc network and discusses the tradeoffs among parameters inside the system.

Zaruba et al. [40] incorporated particle filters along with the Received Signal-Strength Indication (RSSI) of signal measurements collected from 2 APs at various locations and orientations of an indoor environment–creating a grid-based signal-strength map. They reported a mean location error of at most 2.1 m, by deploying 3000 particles in an area of 88 cells. However, their training phase methodology has a large overhead, due to the fact that they collect signal-strength measurements at each cell of the map and for various different orientations.

Evennou and Marx [12] also employed Kalman and particle filters with the Motley-Keenan propagation model. Their signal-strength map was built by acquiring one measurement per room and one measurement in every two meters in the case of a corridor, in a 35x35 \( m^2 \) area with an AP placed at each corner. Kalman and particle filters had a mean location error of 2.29 m and 1.86 m, respectively. In [13], they even incorporated
information from Inertial Navigation Systems (INS), increasing the accuracy with a mean location error of 1.53 m using 10,000 particles.

Hightower and Borriello [18] created a location-sensing system for indoor environments that also applied particle filters using the Sequential Importance Sample with Resampling (SISR) algorithm. A procedure called KLD adaptation determined the appropriate number of samples at each step. A robot walking with a speed of 0 to 2 m/s was used for collecting measurements from a IEEE 802.11 client device, an ultrasound badge, two types of infrared badges and RFID tags. They tested the system in a 900 m² testbed. The 80% location error was at most 1.8 m.

Howard et al. [19], used the standard Monte Carlo Localization algorithm and IEEE 802.11-enabled mobile robots exploiting only a signal-strength map and odometry in order to localize themselves.

Gwon et al. proposed two location-estimation algorithms for indoor environments, based on RF technology combining information from other positioning techniques to improve accuracy, in [17]. The first algorithm takes into consideration various information sources so as to estimate the location of non-mobile users. The second one defines and analyzes the problem of two different locations having similar RF characteristics. They performed an evaluation of these algorithms in an environment incorporating 4 IEEE 802.11 APs, 3 Bluetooth APs and a laptop. These algorithms resulted in improving the location accuracy by at least 24%.

Horus WLAN [48], proposed by Youssef et al., operates in training phase and the run-time phase. As Horus WLAN computes the correlation between consecutive signal-strength samples using an autoregressive model. A high number of samples is required for creating a valid model. Thus, extensive results on the number of samples and the impact of the sample size to the location estimation, was not provided. Moreover, high number of samples increased the time needed for the system to both be trained and compute the position of the user. The evaluation of the system in a 11.8 x 35.9 m² area with coverage of totally 5 APs and an average of 4 APs per cell, showed that Horus WLAN has 90% of 1.32 m location error.

Furthermore, ARIADNE is another novel and automated location determination method
proposed by Yiming Ji et al. [20]. Using a two dimensional construction floor plan and only a single actual signal-strength measurement, ARIADNE generates an estimated signal-strength map comparable to those generated manually by actual measurements. Given the signal measurements for a mobile, a proposed clustering algorithm, it searches the signal-strength map to determine the current mobile’s location.

In order to create the map, ARIADNE uses a radio propagation model. In the search phase, ARIADNE selects a set of candidate locations according to a predetermined mean square error (MSE) threshold. Because of the imprecise nature of the estimated SS-MAP, some of the selected candidate locations may be scattered around the floor plan. Before clustering, the scattered positions must be detected and omitted from the set of candidate locations. The main purpose of the clustering phase is to determine the intrinsic grouping of the set for these positions, and to select the right cluster for the estimates. The largest cluster has the highest probability to contain the right position.

The basic idea of COMPASS positioning algorithm, as presented by King et al. in [23], is to sample the signal-strength for selected orientations at each reference point during the offline phase and combine a subset of these values to histograms in the online phase, so that an orientation-specific signal-strength distribution can be computed and utilized to increase the accuracy of position estimates. At each reference point they collect the AP’s signal-strength distribution for eight orientations, during the offline phase. In the online determination phase, the user takes samples containing the signal-strength measurements and orientation. The algorithm determines the user’s position, based on these data, using Bayes’ Rule.

Elnahrawy et al. make a comparative study on the limits of localization using signal-strength in [10] and they characterized the limits of a wide variety of approaches to localization in indoor environments using signal-strength and IEEE 802.11 technology. According to this survey when only IEEE 802.11 technology is used, the best expected median location error is 10 ft (≈3 m) and 97-th percentile 30 ft (≈9.14 m) for a good algorithm and much sampling. However, a median error of 15 ft with a 40 ft 97-th percentile is obtainable with much less sampling effort.

Comparisons and examinations of uncertainty PDFs suggest that algorithms based on
matching and signal-to-distance functions are unable to capture the myriad of effects on signal propagation in an indoor environment. While many of the algorithms can explore the space of this uncertainty in useful ways, e.g., by returning likely areas and rooms, they cannot reduce it. Given the large training sets, it is unlikely that additional sampling will increase accuracy. Adding additional hardware and altering the model are the only alternatives. It is unclear if building models at the level of detail where one must model all items impacting signal propagation (walls, large bookshelves, etc.) would be worth the improvements in localization accuracy.

Ekahau Positioning Engine (EPE) [9] is a commercial tracking system used both within indoor and outdoor applications, using IEEE 802.11 technology.

Active Badge uses diffuse infrared technology, and requires each person to wear a small infrared badge that emits a globally unique identifier every ten seconds or on demand. A central server collects this data from fixed infrared sensors around the building, aggregates it and provides an application programming interface for using the data. The system suffers in the case of fluorescent lightning and direct sunlight, because of the spurious infrared emissions these light sources generate.

SmartFloor employs a pressure sensor grid installed in all floors to determine presence information. It can accurately determine positions in a building without requiring from users to wear tags or carry devices. However, it is not able to specifically identify individuals. UbiSense provides an accuracy of 15 cm using a network of ultra wide band (UWB) sensors (17 cm x 12 cm x 5 cm) installed and connected into a building’s existing network (four sensors in a typical office environment of 625 $m^2$). The UWB sensors use Ethernet for timing and synchronization. They detect and react to the position of tags based on time difference of arrival and angle of arrival. An RF tag is a silicon chip that emits an electronic signal in the presence of the energy field created by a reader device in proximity. Location can be deduced by considering the last reader to see the card. RFID proximity cards are in widespread use, especially in access control systems.

The Active Bats system consists of a controller that sends a radio signal and a synchronized reset signal simultaneously to the ceiling sensors using a wired serial network. Bats respond to the radio request with an ultrasonic beacon. Ceiling sensors measure
time-of-flight from reset to ultrasonic pulse. ActiveBat applies statistical pruning to eliminate erroneous sensor measurements caused by a sensor hearing a reflected pulse instead of one that traveled along the direct path from the Bat to the sensor. A relatively dense deployment of ultrasound sensors in the ceiling can provide within 9 cm of true position 95% of the measurements.

Positioning using RFID technology is another way to locate users. In [46], Wang et al., present a system where a number of RFID tags and/or readers with known locations are deployed as reference nodes. They propose two positioning schemes, namely, the active scheme and the passive scheme. The former scheme locates an RFID reader. For example, it may be employed to locate a mobile person who is equipped with an RFID reader or an object that is approached by an RFID reader. The passive scheme locates an RFID tag, which is attached to the target object. Both approaches are based on a Nelder-Mead nonlinear optimization method that minimizes the error objective functions. Moreover, sphere geometry is used to 3-D localize the user in the testbed. The proposed methodology is very interesting but only simulations were run and real life experiments were not conducted at all, in order to be able to compare it with our own extended system.

Moreover, Anonymous Tracking using RFID tags [24] is another aspect that has a main goal of preserving the user’s privacy. In short, the authors propose a privacy-preserving scheme that enables anonymous estimation of the cardinality of a dynamic set of RFID tags, while allowing the set membership to vary in both the spatial and temporal domains. In addition, the proposed scheme can identify the dynamics of the changes in the tag set population. The main idea of the scheme is to avoid explicit identification of tags, by using Gaussian models to describe the distribution of the signal between the tags and readers, as well as the set theory, in order to estimate multiple overlapping RFID tag sets. This implementation that identifies a whole set of tags, has a completely different philosophy than our own system that only identifies one user at a time. However, it depicts some very serious subjects like privacy on which researcher have paid great attention over the last few years.
2.4 CLS

The Cooperative Location-sensing System (CLS) employs the p2p paradigm and a probabilistic framework to estimate the position of wireless-enabled devices in an iterative manner without the need for an extensive infrastructure or time-strenuous training. Was designed and partly implemented by C. Fretzagias, A. Katranidou and C. Vandikas with the help of the mobile group at ICS-FORTH, under the guidance of Prof. M. Papadopouli [16, 21, 44]. CLS can incorporate signal-strength maps of the environment to improve the position estimates. Such maps have been built using measurements that were acquired from APs and peers during a training phase.

CLS uses the following features:

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

2. the p2p paradigm

CLS applies the p2p paradigm by enabling devices to gather positioning information from other neighboring peers, estimate their distance from their peers based on signal-strength measurements, and position themselves accordingly [16]. Periodically, CLS can refine its positioning estimations by incorporating newly received information from other devices.

CLS adopts a grid-based representation of the physical space; each cell of the grid corresponds to a physical position of the physical space. The cell size reflects the spatial granularity/scale. Each cell of the grid is associated with a value that indicates the likelihood that the node is in that cell.

These values are computed iteratively using one of the following approaches [44]:

1. A simple voting algorithm, through which a local CLS instance casts votes on cells of the grid. A vote on a cell indicates the likelihood that the local device is located in the corresponding area of that cell.

2. A particle filter-based model.
2.4.1 Generation of fingerprints

A node tries to position itself on its local grid through a voting process in which devices participate by sending position information and casting votes on specific cells. In the implementations using IEEE 802.11 both infrastructure (i.e., APs) and other peers (incorporating the p2p paradigm) can participate.

The process that transforms the acquired measurements into probability of being at a certain location can be implemented in different ways, described later on, and depends on the CLS implementation and the statistical criteria used.

Each local CLS instance employs an algorithm that transforms (maps) these signal-strength values to either distance or position estimates. The transformation algorithm can be based on a radio attenuation model or a pattern matching algorithm. Such 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. 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.

When a training phase prior to voting is feasible, CLS can build a map or signature of a physical space, which is a grid-based structure of the space augmented with measurements from peers. Two such signature types are explored, namely, position-level and distance-level signal-strength based signatures.

At run-time, the local CLS instance acquires signal-strength measurements from APs/peers, constructs a run-time signature, and compares this run-time signature with the ones that have been generated during the training phase.

To explain, in all cells of the grid-based representation of the physical space we collect multiple measurements from all APs in proximity. Afterwards we transform them in the statistical information we want to use (i.e., confidence intervals, percentiles), and sort them in a vector per AP. For example:

\[
\langle i, j, ss_1, ss_2, ..., ss_k \rangle, \forall i, j \text{ representing the coordinates of a specific cell of the training map and } ss_k \text{ the statistical information used for the } k\text{-th AP.}
\]
When we have this kind of vector for all cells of the grid-based representation of the 
physical space, we merge them and create a 2-D array that composes our training map.

During run-time phase, the user collects multiple signal-strength measurements, and 
then transforms them like before in the same statistical information as the training map 
used, depending on the implementation of CLS used. A new vector is created this way, 
like the following
\[\langle ss^R_1, ss^R_2, \ldots, ss^R_k \rangle,\] 
where \(ss^R_k\) represents the run-time statistical information used for the \(k\)-th AP. This vector is different for each cell and each time measured and therefore 
we call it the cell’s signature.

Then, a simple voting algorithm finds the cell that is most likely for the user to be there, by comparing the signature of the run-time cell with all entries of the training map, 
and finding the best match.

We explore various different criteria for the comparison, namely, confidence interval-, 
quartile-, percentile-, particle filter- and p2p-based criteria.

**Confidence interval-based**

In the case of confidence interval-based implementation of CLS, where \(ss_k\) and \(ss^R_k\) are 
replaced by \(ss_k = [ss^l_k, ss^u_k]\) and \(ss^R_k = [ss^R^l_k, ss^R^u_k]\), respectively. Confidence intervals 
in general indicate the reliability of an estimate to a percentage that we define. This is 
usually 95%, and shows through an interval \([lower, upper]\) the range in which 95% of the 
measurements taken, for example, belong (\(lower\) is the lower and \(upper\) the upper bound 
of the interval).

During training, a position-level signature based on confidence intervals associates each 
position of the terrain (cell of the grid) with a vector of confidence intervals. Each entry 
of the vector corresponds to an AP and its associated confidence interval of the RSSI 
values that were recorded from beacons received from that AP during the training phase. 
Beacons are messages broadcast by APs periodically. In this case of measurements, the 
confidence interval would be \([ss^l_k, ss^u_k]\), where \(ss^l_k\) illustrates the lower, and \(ss^u_k\) the upper 
bound of the confidence interval of the measurements collected from the \(k\)-th AP in the 
specific training cell. This means that the training map now has the following form,
\langle i, j, [ss^1_i, ss^u_i], [ss^1_j, ss^u_j], \ldots, [ss^1_k, ss^u_k] \rangle, \forall i, j \text{ training cell.}

At run time, a local CLS instance acquires a number of beacons from APs and computes a confidence interval for each AP, \([ss^{R,l}_k, ss^{R,u}_k]\) illustrating the lower and upper bound of the confidence interval of the measurements collected from the \(k\)-th AP in the specific run-time cell. The run-time signature is like this, \(\langle [ss^{R,l}_1, ss^{R,u}_1], [ss^{R,l}_2, ss^{R,u}_2], \ldots, [ss^{R,l}_k, ss^{R,u}_k] \rangle\).

The algorithm assigns a weight at cell \(c\), \(w(c)\) according to the percentage of overlapping of the corresponding confidence intervals. There are two different cases of overlapping confidence intervals. Firstly, the case of training measurements being lower than run-time measurements (1) and secondly the case of run-time measurements being lower than the training measurements (2). The assignment of vote of weight in each case follows:

1. \(ss^{l}_k < ss^{R,l}_k < ss^{u}_k < ss^{R,u}_k \Rightarrow w(c) = \frac{(ss^{u}_k - ss^{R,l}_k)}{(ss^{R,u}_k - ss^{R,l}_k)}\)
2. \(ss^{R,l}_k < ss^{l}_k < ss^{R,u}_k < ss^{u}_k \Rightarrow w(c) = \frac{(ss^{R,u}_k - ss^{l}_k)}{(ss^{u}_k - ss^{R,l}_k)}\)

In the case of no overlapping the vote is \(w(c) = 0\), and if the two confidence interval match \(w(c) = 1\).

**Percentile-based**

Like the confidence interval, the percentile-based criteria uses also signal-strength measurements. However, it captures more detailed information about their distribution, and thus, allows for more accurate comparisons. The weight of a cell \(c\), \(w(c)\), is computed as follows:

\[
w(c) = \sum_{i=1}^{n} \sqrt{\frac{1}{p} \sum_{j=1}^{p} (ss^{R,j}_i - ss^{l}_j)^2} \tag{2.1}\]

where \(n\) is the number of APs, \(p\) is the number of percentiles, \(ss^{R,j}_i\) is the \(j\)-th run-time percentile and \(ss^{l}_j\) the \(j\)-th percentile of the \(i\)-th AP at training cell \(c\).

There are two implementations of the percentile-based algorithm of CLS, the decile and the quartile one. In the former one CLS uses ten percentiles \((p = 10)\) per AP (i.e., 10%, 20%, \ldots, 100%), while in the latter one it needs only four percentiles \((p = 4)\) per AP (i.e., 25%, 50%, 75%, 100%).
2.4.2 p2p-based

Distance-based signatures are employed only when non-stationary peers participate in voting and training is possible. They relate distances with signal-strength measurements recorded by the local CLS instance at the reception of messages from other non-stationary peers at the respective distances during training phase. During training, devices located at various positions participate in CLS by sending messages and recording the signal-strength value with which each of these messages was captured and the distance between the two devices. Specifically, a training set is composed of entries, each including a distance and a confidence interval of the signal-strength values recorded from messages exchanged between peers at that distance.

At run-time, the distance $D$ between two peers is estimated using the following formula:

$$D = \frac{\sum_{i=1}^{n} D_i \cdot \sqrt{(ss_{R,l} - ss_l)^2 + (ss_{R,u} - ss_u)^2}}{\sum_{j=1}^{n} \sqrt{(ss_{R,l} - ss_l)^2 + (ss_{R,u} - ss_u)^2}}$$

(2.2)

where $n$ is the number of entries in the training set, $D_i$ is the distance from the $i$-th entry of the training set, $[ss_{R,l}, ss_{R,u}]$ is the run-time confidence interval and $[ss_{l,k}, ss_{u,k}]$ is the confidence interval from the $k$-th entry in the training set.

2.4.3 Particle filter-based

In probabilistic terms, CLS can be formulated as the problem of determining the probability of a node being at a certain location given a sequence of signal-strengths. Assuming first-order Markov dynamics, the above problem can be expressed using the network graph depicted in Figure 2.5, where $x_k$ is the node location (system state) at time instant $k = 1, \ldots, T$. $x_k$ cannot be observed directly (it is “hidden”). Yet, for each location $x_k$, a measurement vector $y_k$ (signal-strength) is available that depends on the hidden variable according to a known observation function.

Due to the Markov assumption, each node location, given its immediately previous location, is conditionally independent of all earlier locations, that is

$$P(x_k | x_0, x_1, \ldots, x_{k-1}) = P(x_k | x_{k-1})$$
Similarly, the observation at the $k$-th time instant, given the current state, is conditionally independent of all other states

$$P(y_k | x_0, x_1, ..., x_k) = P(y_k | x_k)$$

Based on this model, location-sensing can be formulated as the problem of computing the location $x_k$ of a node at time $k$, given the sequence of observations $y_1, y_2, ..., y_k$, up to time $k$, that is, determining the a posteriori distribution $P(x_k | y_1, y_2, ..., y_k)$. 

To estimate the above a posteriori, which is actually a density over the whole state space, we use particle filter. Particle filter is a technique for implementing a recursive Bayesian filter by placing Monte Carlo sampling. According to this technique, the a posteriori $P(x_k | y_1, y_2, ..., y_k)$ is expressed as a set of samples

$$x^{(L)} = (x, y)^{(L)}, L = 1, 2, ..., N$$

distributed among the whole state-space. The denser the samples in a certain region of the state-space, the higher the probability that the node is located in that region.

Unlike Kalman-filter, particle filter does not impose any constraints on the format of the involved distributions and noise models, or on linearity of the involved functions. This makes them particularly well-suited to location-sensing.

**Sampling/Importance Resampling algorithm:** To generate and maintain the samples (particles), we utilize the Sampling/Importance Resampling (SIR) algorithm introduced by Rubin [38]. According to SIR, instead of sampling the true a posteriori distribution (which is not possible because this distribution is not available in closed form),
samples are drawn from the so-called proposal distribution $\pi(x_k | y_1, y_2, \ldots, y_k)$. To compensate for this difference, each sample $s^{(L)}$ is also assigned a weight $w^{(L)}$, which is computed, according to the Importance Sampling Principle:

$$w^{(L)}_t = \frac{P(x_k | y_1, y_2, \ldots, y_k)}{\pi(x_k | y_1, y_2, \ldots, y_k)}$$

By choosing the proposal distribution to be the transition prior $P(x_k | x_{k-1})$, the weights can be computed as

$$w^{(L)}_t = \frac{P(x_k | y_1, y_2, \ldots, y_k)}{\pi(x_k | y_1, y_2, \ldots, y_k)} \approx \frac{P(y_k | x_k) \cdot P(x_k | x_{k-1})}{P(x_k | x_{k-1})} \cdot w^{(L)}_{t-1} = P(y_k | x_k) \cdot w^{(L)}_{t-1}$$

To avoid degenerate situations in which large numbers of samples have weights close to zero, after a few iterations, SIR also includes a resampling step which ensures that unlikely samples are replaced with more likely ones. The following pseudocode describes our implementation of SIR.

**SIR algorithm:**

1. For $L = 1, \ldots, P$

2. Transition step: Draw a new sample $x^{(L)}_k$ from the transition prior of sample $x^{(L)}_{k-1}$ according to $P(x^{(L)}_k | x^{(L)}_{k-1})$

3. Observation step: Calculate the weight $w^{(L)}_k$ of sample $x^{(L)}_k$, according to the importance sampling principle. That is, $w^{(L)}_k = w^{(L)}_{k-1} \cdot P(y_k | x^{(L)}_k)$

4. End for loop

5. Normalize weights

6. Resample

7. Goto Step 1

Initially, all particles are uniformly distributed among the state space. Particles whose their validity is confirmed by observations tend to get larger weights, so after a few resam-
pling steps, particles are expected to gather around certain regions. If sufficient input is available to resolve all ambiguities, all particles will eventually gather in one region only.

The density of particles in a specific region of the state space indicates the probability of the node to be in that region. The expected node location is defined as the location within the state-space with the highest particle density and is computed iteratively using the highest particle density algorithm. In this algorithm, the particle with the highest number of other particles within a certain radius is initially chosen as the circle center. The centroid of these supporting particles is then calculated and the process is repeated with this calculated centroid being the circle center until convergence is achieved.

**Computation of transition prior:** Motivated by the observation that nodes located in offices tend to remain mostly stationary while nodes in corridors tend to be in motion, we segment the state space into two different types of area: corridor and office areas.

To compute the transition prior $P(x_k^{(L)} \mid x_{k-1}^{(L)})$ for the Step 2 of SIR, we consider the following cases:

1. the particle’s current position estimate $x_k^{(L)}$ being inside an office area
2. the particle’s current position estimate $x_k^{(L)}$ being in a corridor area

The transition prior for office areas is assumed to be a gaussian distribution, centered at the previous position estimate $x_{k-1}^{(L)}$ and having a standard deviation of $\sigma_{off}$. For corridor areas, the transition prior was assumed to be uniform within a radius $\delta_{cor}$ around the previous position estimate defined as follows:

$$P(x_k^{(L)} \mid x_{k-1}^{(L)}) = \begin{cases} \frac{1}{\pi \sigma_{cor}^2}, & \text{if } \| x_k^{(L)} - x_{k-1}^{(L)} \| \leq \delta_{cor} \\ 0, & \text{otherwise} \end{cases}$$

The standard deviation $\sigma_{off}$ of the “office” transition prior and the radius $\delta_{cor}$ for the “corridor” transition prior were found experimentally and were set to be 0.5 m and 1 m, respectively.

**Computation of observation probability:** To compute the observation probability $P(y_k \mid x_k^{(L)})$, required for Step 3 of the SIR algorithm, we utilize observations from both APs and other
peers.

For APs, observation models were extracted and stored offline during training, as described before. That is, the state-space is discretized into a finite number of cells. For each state-cell, the mean signal-strength of each AP and its corresponding standard deviation is computed and stored. For cells with no measurements, we apply interpolation and store the reported values.

For peers, signal-strength values measured during training are stored as a function of distance. As in the case of APs, the distance-space is discretized, and for each distance, the mean and standard deviation of the observed signal-strengths are stored. It is important that the gathered training data captures all the variation in different positions within the state space.

Although peer training data exhibits large variation, it can improve the accuracy by considering that the distance of two peers is likely to exceed a certain threshold, when the corresponding signal-strength measurements drop to zero.

Based on the assumption that signal-strength measurements are independent, at runtime, all available signal-strength measurements are combined to compute the joint observation probability as:

\[
P(y_k | x_k) = P((y_{k1}, y_{k2}, \ldots, y_{kn}) | x_k) = P(y_{k1} | x_k) \cdot P(y_{k2} | x_k) \cdot \ldots \cdot P(y_{kn} | x_k)
\]

where \(n\) is the number of measurements.

### 2.5 C-NGINE

C-NGINE [25] is a framework for structuring advanced location-based and context-aware services that integrates up-to-date technologies and develops novel mechanisms and interactive interfaces by applying techniques and formalisms from the Semantic Web and Ambient Intelligence domains. The objective is to explore the intelligent pedestrian navigation by implementing a contextual guide for users in indoor environments. C-NGINE focuses on modeling and representing context using Semantic Web technologies for efficient
processing and dissemination of context-based knowledge in order to develop services for mobile users. C-NGINE enriches indoor navigation with many capabilities which realize the Ambient Intelligence paradigm:

1. Modeling of users’ profiles (capturing various dimensions of user characteristics) and their context.

2. Development of semantic services customized to users’ needs and preferences, such as path-finding, navigation and visualization according to users’ impairments and preferences and generation of semantic calendars and events based on dynamically created user groups.

3. Adaptation of the spatial domain to dynamic changes (e.g. obstacles, operational malfunctions etc.)

4. User’s movement tracking.

5. Dynamic path re-planning, in case of user deviation.

6. Generation of dynamic tours based on parameters such as personal user interests and preferences, relative contextual information and available time.

7. Preserving user’s privacy by applying user-defined preferences in terms of rules.

C-NGINE consists of five fundamental components; a User Interface, a Knowledge Base (KB), a Services Layer, a Reasoner and a Rule Engine. These components form the system’s architecture as depicted in Figure 2.6.

Besides User Interface, which is intuitive enough, KB contains the schema and instances of our modeled classes and properties; it stores all information in the form of semantic web rules and OWL-DL [45] ontologies. Integrating web ontology languages, such as OWL, with context-aware applications has manifold benefits. These languages offer enough representational capabilities to develop a formal context model that can be shared, reused, extended for the needs of specific domains, but also combined with data originating from other sources, such as the Web or other applications. Moreover, the development of the logic layer of the Semantic Web is resulting in rule languages that enable reasoning about
the user’s needs and preferences and exploiting the available ontology knowledge. For usability and clarification purposes, we split $KB$ into three ontologies; the User Profile Ontology (UPO), the Spatial Ontology for Path-Finding (SOP) and the User Tracking Ontology (UTO). SOP supports both path search and the presentation tasks of a navigation system. UTO captures useful information about user location and motion state. For the purposes of this thesis, only UPO is thoroughly analyzed, which is the essential building block for the development of personalized context-based services. C-NGINE’s main functionality is represented by the Services Layer, which is decomposed into three mutually dependant components; the User Modeling Component, the Contextual Path-Finding Component, and the User Tracking Component. The information flow between the three components in Services Layer is analyzed as follows: The first important task that the system confronts, during execution is the creation of an appropriate user profile. During this procedure, User Modeling component receives information from SOP ontology in order to associate users with space elements (e.g., associating a user with his/her office). Respectively, Contextual Path-Finding component communicates with User Modeling component in order to provide the optimal navigation path and the guideline information according to user’s profile and preferences. The User Tracking component when triggers dynamically-
planning process, calls Contextual Path-Finding to calculate a new optimal path from user’s current position to the same desired destination.

Finally, Reasoner and Rule Engine are mechanisms that are responsible for reasoning tasks such as checking inconsistencies, classifying instances under various classes, computing inferred types and reasoning on the available context using knowledge supplied in the form of rules.

The aim of User Modeling component, designed by M. Michou, is to capture all user-related information. As a result, we model not only the users’ profiles, but also their context. UPO is developed as an extension of OntoNav’s User Navigation Ontology (UNO) [22] to give emphasis on context-awareness and modeling of users’ dimensions and provide a flexible model by limiting context to the conditions that are relevant for the purpose of C-NGINE framework. Such characteristics are essential for the development of group-oriented services, such as semantic notes, semantic alerts, and single-user/group calendars.

User Tracking Component, which is designed by E. Nikoloudakis, is responsible for tracking user’s movement on the suggested path and providing with useful information about user location and motion state. For example, when a user enters an elevator in order to move from the ground floor to the first floor of a building, the information provided by the system will be: “User is inside elevator and moves upwards”. Furthermore, the component is responsible for checking whether a user is following the proposed path and for detecting any deviations. In case of a path deviation, the User Tracking component dynamically re-plans the user’s suggested path from his/her current position. Another case, in which the tracking mechanism executes the dynamic re-planning process, is upon receiving user feedback. For example, in case a user finds an obstacle blocking the way, he/she sends feedback to the application about the collision point and the system proposes a new, collision-free, path to user’s desired destination.

The Contextual Path-Finding component, designed by M. Kritsotakis, is responsible for providing the “best traversable” path, according to the parameters analyzed in the previous section. Moreover one of the tasks accomplished by this component is the population of SOP and the creation of a spatial graph by using floor maps. The user options concerning path-finding are:
1. Typical path-finding: We become aware of the desired user destination in one of the following ways:

   (a) The user clicks on the desired destination directly on the map

   (b) The user chooses to go to a spatial element that is closest to him/her from a selection list (e.g. printer, bathroom)

2. Time-based tour: The user denotes that he/she has a particular amount of time available and requests the system to provide him/her with a tour. This tour must include elements that may be of interest to him/her. Also the destination point is considered to be the same as the starting one and the requirement is that the user is back there in time provided as input. The time needed to traverse the requested route, is actually affected by the user’s profile and in particular by his physical capabilities/impairments. (e.g. an impaired user with a wheelchair is considered to move with lower speed when compared to a user with no impairments).

   All these components need the exact position of the user in order to function. The participation of CLS in this system is to provide this information (i.e., current position of the user of C-NGINE) to the User Tracking and Contextual Path-Finding components.

   CLS runs in parallel with the rest of the C-NGINE components, without adding any overhead to the system. Each time the current position of the user is computed, it is saved in a pre-defined file that the other components of the system have access to. In this way, every time any of the two location-based components need to retrieve the users’ latest position, they can do it by accessing the specific file. This file is renewed with a new estimated position every 3-4 sec, which is how long CLS needs to perform an estimation.

   In this way C-NGINE can run real-time scenarios with mobile users in the ICS testbed.

   Below, a typical scenario is presented, which takes place in the premises of a research institute, like ICS-FORTH, and takes advantage of the capabilities of C-NGINE. Professor Smith has just arrived in town for a business meeting. After the meeting he realizes that he has a little time to spare and decides to visit the nearby research center, where he has many professional connections. Upon entering the main gate of the building he is provided with a PDA through which he can have access to C-NGINE. After logging
in, Prof. Smith fills in a form that describes his profile. He chooses one of the default profiles, and in particular one describing him as a male non impaired visitor, and then fills in some more specific details. He chooses escalator or elevator as his routing preferences. He selects from a list “machine learning” and “computer vision” as his scientific fields of interest and goes on to using the navigation system. The first screen Prof. Smith views provides him with a description of the events taking place during that day, along with the time and maybe a list of people participating in. He notices that there is a series of lectures on machine learning algorithms all day long, and considers paying a visit if he has time later on. But before proceeding, he decides that he would better go to the restroom, to freshen up. He asks for a path to the closest restroom. The navigation system highlights a path on the navigation map, from his current location to the closest restroom for men, along with a simple textual description of the path he must follow. While moving towards the desired destination, the system visualizes his current location and the motion direction using both textual description (e.g. located in Corridor X, moving northeast) and landmark highlighting (e.g. showing an arrow depicting direction on the spatial element he is currently located). While moving towards the restroom, he realizes that the suggested path is blocked, as a crew of electricians performs maintenance operations and has blocked the whole corridor. Prof. Smith uses the navigation map, to denote that this particular corridor is blocked, and moves out of the suggested path. The system senses that the user has deviated from the selected path and tries to figure out another way to the restroom. However, the path leading to the same restroom and avoiding that corridor is now too long; therefore, the system suggests a new restroom and Prof. Smith finally reaches his destination.

Prof. Smith feels tired of wandering around and decides he should catch up with some of his friends in the institute. Since he has not visited the Institute for a long time, he is not sure where the office of his friend Dr Anderson is. So, he asks the system for a list of people who have a public profile. Dr Anderson is indeed, in the list, but through his public calendar, Prof. Smith realizes that his friend must be in a meeting and probably will not be able to meet him at that particular time. But one of the options in the list of public profiles is Prof. Heinz, who is also a friend and seems to be in his office. So, Prof.
Smith asks for directions to his office, which is located on the second floor. The system, being aware that Prof. Smith prefers only elevator or escalator, suggests a path that leads him to the second floor, through an elevator. Prof. Smith follows the path and meets with his friend. After catching up with Prof. Heinz, Prof. Smith realizes that time has passed and he really should go. He makes a phone call, and the taxi agency informs him that the taxi in closest proximity will be there in at least 30 minutes, so Prof. Smith finds the perfect chance to wander around the research center. He denotes to the system that he has 30 minutes to spend and asks for a route that starts from the building entrance, where he is currently located, and returns him back there in time. The system, using information about Prof. Smith’s interests, suggests a route that walks through a corridor with posters showing the latest achievements of the computer vision laboratory. Another point of interest is also the conference room where the talking session on machine learning algorithms is now taking place. Prof. Smith follows the suggested route, and after about 30 minutes is located in the main entrance, where the taxi waits for him.
Chapter 3

Experimental results

In this Chapter an extensive evaluation of the algorithms described in Chapter 2.4 is presented. Firstly, we are going to mention the benchmarks we used during the evaluation. Secondly, a description of the various testbeds is going to follow, along with the evaluation results for each one of them and a study concerning the impact of the number of APs in the accuracy of the system is going to end this Chapter. Finally, some techniques for improving the accuracy of the previous methods are presented.

There are two ways to evaluate the proposed positioning algorithms.

1. Run-time execution of CLS means that signal-strength measurements are collected via `iwlist` [43], by a CLS instance, and then transformed into the statistical information needed and then compared to the training phase data. In this case the program runs the iwlist command as many times as the exact predefined and same number of measurements needed (60 in the current version of CLS for training phase and 30 for run-time phase) and collects signal-strength by all APs in the area. Furthermore, before measurements from an AP are used again the system verifies that there are at least 30, for example, measurements per AP. If fewer measurements are collected from an AP, then the system adds the missing measurements considering the latest existing measurements. If no measurements are collected from a certain AP, then it does not participate in the voting process and is regarded as an AP not covering the specific position of the physical space.

2. Off-line execution means that signal-strength measurements are collected by the user
manually via `iwlist`. They are, afterwards, transformed into the statistical information needed using certain scripts (in bash scripting language and Matlab). Then they are given as input to an off-line-version of the positioning algorithms and they are compared to the training phase data. In this case, if more than 30 measurements are collected by an AP, the program only keeps the first 30 of them, ignoring the rest. If the opposite happens, which is due to human error (the person manually collecting the measurements terminated the iwlist process too soon), we have chosen to make an interpolation of the measurements collected to add the ones missing. In other words, we add the missing measurements considering the latest existing measurements.

### 3.1 Benchmarks

The benchmarks we are going to examine in this Chapter have to do with both accuracy and precision as described in Chapter 2.1. Different statistical methods are evaluated, as well as the impact of outliers existing in the signal-strength timeseries collected, and other improving techniques. The number of APs and it’s impact on the accuracy of our positioning methods is another parameter we examined.

### 3.2 Testbed description

The training and run-time experiments took place in the Telecommunication and Networks Lab (TNL) at the Foundation for Research and Technology-Hellas (FORTH) in Greece, an area of $84m^2$ (7m x 12m), and in Institute of Computer Science (ICS) of the Foundation for Research and Technology - Hellas (FORTH), area of $1462m^2$ (43m x 34m), both with a grid representation considered with cells of size 0.5 cm x 0.5 cm. Another testbed was the Cretaquarium [8] in Hersonissos – Heraklion which was an area of $1760m^2$ (32m x 55m) and consisting of a grid of 1 m x 1 m.

To generate the signal-strength signatures for training map, we acquire 60 samples of signal-strength values for each cell of the grid in which signatures were generated. In these cells, the trainer waited for approximately one minute to acquire the training signal-strength measurements. To capture signal-strength values, the following monitoring tools
were used:

1. iwlist [43], which polls each channel and acquires the MAC address and signal-strength measurements from each AP (in dBm).

2. tcpdump [41], a passive scanner relying on libpcap for the retrieval of each packet.

The experiments were conducted around midday on several weekdays in order to have a typical interference from people working or visiting throughout 2008 and the first months of 2009. Sony Vaio and Toshiba laptops with the same wireless adapter (ipw2200) were used for both the training and run-time experiments. On the following all testbeds and results concerning them are going to be described.

All of our algorithms were implemented in C, C++, and Matlab and evaluated with real-life measurements.

During experimenting with these algorithms, we also decided that we should also examine the impact of the physiology of the testbed on the accuracy results of the system. This is why we conducted experiments in three different areas.

3.2.1 Cretaquarium

Cretaquarium is the biggest and most visited aquarium in Greece, covering an area of 1760m². It currently consists of about 30 tanks, while another 25 are now being installed, and 8 APs in total cover the whole testbed, out of which 5.7 APs, on average, can be detected at a given cell. In Figure 3.1 the signal-strength coverage from all APs is illustrated. We divided the physical space into a grid representation with cell sized 1 m x 1 m.

Training signal-strength measurements were collected in the whole testbed and they were transformed in all statistical information described above in order to evaluate all implementations in this testbed as well. The first set of measurements were collected on the 15-th of November 2007 for creating the training set, and on the 16-th of November 2007 for runtime measurements, when about 100 people were in the testbed (Set 1).

The first decision we had to make as far as the confidence interval-based implementation is concerned was the percentage of confidence level. Usually the 95% confidence interval is used. Hence, it is up to the algorithm designer what confidence interval accuracy is
Figure 3.1: SS coverage from all APs in the testbed of Cretaquarium.

used. Therefore we thought that it would be a good idea to test the algorithm for different confidence interval accuracies, and then choose the most efficient in our case, because the higher the accuracy, the higher the margin in the range of values of the interval.

We expect that the lower the accuracy the location error of this method will be decreased as the bounds of the interval become tighter, the rarer it will be to have cases of overlapping confidence intervals between run-time and training phase. Because of the high margin in the range of values it is easier to find overlapping areas with other cells’ confidence intervals. This is the reason why we tried to see the results that the same algorithm would return with 99% and 75% confidence intervals too.

As expected, Figure 3.2 shows that the higher the accuracy of the confidence interval, the higher location error the algorithm returns. For 99%, 95% and 75% accuracy of the confidence intervals, the median location error is 4.3 m, 4.3 m and 3.6 m, respectively.

There is discrimination in the experiments conducted regarding the conditions of the
Chapter 3. Experimental results

Figure 3.2: Location error results in the aquarium using the confidence interval-based algorithm with different confidence interval accuracy levels, under normal conditions. (Set 1)

testbed as far as the density of users and visitors in general are concerned. There are normal conditions, where there are about 100 people, maximum, in the testbed, and extreme conditions where there are about 250 people in the testbed. This happens because during the summer months there is an extreme increase in the number of visitors, due to tourist period, in the Cretaquarium testbed, compared to the winter months.

This phenomenon happens because higher accuracy of the confidence intervals equals to higher margin in the range of values of the interval. This means that when margins are high many confidence intervals would contain the same values, so there would be overlapping area in cells that there should not be any.

This makes us come to the conclusion that the tighter the bounds of the confidence interval, the more accurate the system becomes. Therefore from this point on when we refer to confidence interval-based implementation, 75% of confidence level is always assumed.

After choosing the level of accuracy as far as the confidence interval-based implementation is concerned, we have to compare this one to the rest of the methods.

In the confidence interval-based algorithm only two values representing the data are used. This is too few and cannot describe in detail a set of values (per AP, per cell). If
each set was represented by more than two values, it is a firm belief that the error would
decrease dramatically as the number of values, participating in the voting process grew.
We expect the percentile-based algorithm to have a higher accuracy, due to the fact that
more than two values describe the signal-strength measurements from a specific AP and
in a certain cell. In the deciles’ case ten values per AP and per cell represent the input set
of data, whereas in the quartiles’ case only four of them. Thus, quartiles should be more
accurate than the confidence intervals, and deciles more accurate than quartiles.

Our original hypothesis is validated by Figure 3.3, where the deciles-, quartile- and
confidence interval-based approach for the same testing positions, have a median of 2 m,
2.2 m and 3.6 m location error respectively.

On the following, there is the particle filter-based approach. All the methods mentioned
above compare statistical data to come to an estimated position for the user. A different
approach would be to use a probabilistic model (i.e., particle filters).

Our original intuition is proven wrong, as it is obvious that the particle filter-based
method is the least efficient one with median location error 4.5 m (Figure 3.4). Due to
the fact that this is a probabilistic model, as described before, the system is randomly
initialized. This means that for the same input data the result differs, because of the
random initialization of the grid in the beginning of each iteration. As Figure 3.5 illustrates
for the same input data (same cell of the grid) the particle filters-based algorithm returns these 45 different results for 45 iterations, where the median location error is 15 m.

Considering the results above, we can say that a system using the particle filter-based method cannot be consider as robust as using confidence interval- and percentile-based methods.

In many cases there are outliers that decrease the accuracy of the system whereas in the two other methods the outcome of the algorithm for the same input is the same no matter the number of iterations and it cannot be considered as an outlier.

Finally, the p2p-based method, incorporating one or more extra peers to the confidence interval-based (with 95% accuracy level in this case) system is believed to improve the accuracy of CLS. This is because the more information taken into consideration during the voting process the lower the location error achieved, as proven so far.

We created a new training map containing p2p measurements between 2 peers, both with the same wireless cards (i.e., ipw2200), for different distances between them. Afterwards we run the system in real-time in the Cretaquarium in the same cells we used in the single user confidence interval-based algorithm. 30 experiments took place in open areas of the testbed on the 24th of July 2008 at 12:30 - 14:00. There were heavy conditions,
Figure 3.5: Location error results for 45 iterations of the particle filters-based algorithm for only one cell of the grid (same input arguments). (Set 1)

which means that around 250 people were in the whole testbed at most times (Set 2). The results are described in Figure 3.6.

Figure 3.6: Location error results comparing confidence interval- to p2p-based method, under extreme conditions. (Set 2)

We notice that the median location error using single user, confidence interval-based algorithm is about 4.3 m, whereas it improves to 3.9 m using 1 and 3.6 m using 2 extra peers and falls down to 2.5 m, when 3 extra peers take part in the voting process. As expected, high location errors are the ones improved the most using the p2p paradigm,
whereas in lower location errors (> 2 m) no great improvement is noticed.

As mentioned before, in p2p-based approach, we have an extra training map apart from the one we have in single user method as well. This map has to do with the confidence interval of signal-strength values between two peers per distance. In each distance we collect 60 RSSI measurements between the two peers. Afterwards, we transform these values into confidence intervals in the same way we do with the single-user training set RSSI measurements between the user and APs. Both peers have the same wireless cards (i.e., ipw2200). The positions of the two peers were always in range of each other, but not necessarily in line of sight.

The optimal distance frequency is in question here. We want to find out if the distance is best to be per one meter (0 m, 1 m, 2 m...), per two meters (0 m, 2 m, 4 m ...) or per three meters (0 m, 3 m, 6 m ...). As the Figure 3.7 shows the smaller the distance between peers the higher the accuracy.

![Figure 3.7: Location error results comparing the distance between entries of p2p training set. (Set 2)](image)

The important contribution of the p2p paradigm is that it improves the very high location errors (outliers). For example in this series of testing, there was a cell, where according to the single user confidence interval-based algorithm the location error is 18.1 m, whereas with the use of the p2p paradigm the accuracy is 3 m. In other words, in this way the maximum error is minimized.
This happened because the second peer was in a 6 m distance from the peer trying to locate itself, and the ring computed according to this distance was far away from the result given by the plain confidence interval-based method. In this way the ring excluded the other cells from possible positions of the user and the final result was much closer to the real position of the user.

*CLS vs Ekahau*

A location based information system application was designed for the Cretaquarium visitors that needed to know the position of the user in order to provide information about the tanks around him/her. Therefore, both CLS (using the confidence interval-based method) and Ekahau were deployed in the Cretaquarium.

Ekahau [9] is another commercial positioning system also collecting signal-strength measurements and then performing a centralized voting process calculates the position of
the user. Like CLS, Ekahau also uses the IEEE 802.11 infrastructure, creates a map with calibration data, and compares the training and run-time measurements to estimate the position.

The testbed was divided into zones as depicted by Figure 3.8 (the boundaries of each zone are depicted by the orange lines), according to the needs of this application. This is way this system did not need an exact position as an input from CLS, but only the ID of the zone in which the user is estimated to be in.

We tested CLS and Ekahau at the same time and at the same positions on the 24rd of July at about 12.00 - 12.30, when approximately 200 to 250 people were apparent, which is considered to be extreme conditions, with high interference (Set 3). In each zone we tested the system in about 3 different positions, depending on the size of each zone. As shown in the Figure 3.9 the accuracy of CLS is 91.3%, whereas Ekahau has 80.4% accuracy. The difference is more than 10% in favor of CLS.

![Accuracy results, CLS vs Ekahau in Cretaquarium under extreme conditions. (Set 3)](image)

3.2.2 TNL

The Telecommunication and Networks Lab (TNL) is one of the six laboratories of which ICS consists, and is an area of $84m^2$, and the cells of this grid are sized 0.5 m x 0.5 m. There are 12 APs in total, out of which 8 APs, on average, can be detected at a given cell. Figure 3.10 depicts the floor plan of TNL.

Again the same sequence of steps was followed like in the Cretaquarium testbed, on the 12-th of December 2007, at about 17:00 to 19:00 (Set 4). Figure 3.11 shows once again
the difference on the location error between each algorithm. The minimum location error is achieved by using the decile-based approach, which median error is 1.6 m, and then quartile-, confidence interval- and particle filter-based follow with 2.2 m, 2.3 m and 2.9 m median location error, respectively.

The same problem with particle filters remain as described in the Cretaquarium testbed results. Due to the fact that this is a probabilistic model, the system is randomly initialized in the beginning of each iteration. This means that for the same input data the result differs
In other words the system using this method cannot be considered as robust as using the confidence interval- and percentile-based methods. In many cases there are outliers that decrease the accuracy of the system whereas in the two other methods the outcome of the algorithm for the same input is the same no matter the number of iterations and it cannot be considered as outlier.

![Figure 3.12: Location error results for 100 iterations of the particle filters-based algorithm for one cell of the grid (same input arguments). (Set 4)](image)

In the case of one of the other methods is used, apart from the particle filter-based one, even if the result of the algorithm is one with high location error this is due to the input data and the alteration of the signal-strength values in the specific cell and at that particular moment in comparison to the data obtained in training phase. On the other hand in the case of particle filters this low accuracy result would be either due to the random initialization of the grid, or due to a combination of this and the variation of the signal-strength. In order to minimize the impact of this random grid initialization we perform additional checks. They ensure that the result the algorithm returns is the right one and that if we try to find the user’s position again with the same data, the same outcome will be returned.

To evaluate the impact of the number of peers on the performance, we ran experiments
varying the number of participating peers (at run-time). Figure 3.13 shows the impact of one extra node – peer in the accuracy of single user method. The measurements took place on the 2-nd of July 2008 (Set 5). Once again we can see that the median location error is decreased to 2.1 m from 2.3 m when one user takes part in the voting process. There is no substantial improvement on the accuracy due to the fact that the original error, when no peers are apparent, is not as big as in the case of the Cretaquarium testbed. Therefore, there is not much improvement to be made using extra peers.

Figure 3.13: Location error results comparing confidence interval- to p2p-based. (Set 5)

The reason why the location error in general is lower than in the previous testbed is that the size of the actual testbed is smaller.

3.2.3 ICS

The Institute of Computer Science (ICS) is one the seven institutes of the Foundation for Research and Technology - Hellas (FORTH), a major national research center partly funded by the General Secretariat for Research and Technology of the Hellenic Ministry of Development.

This testbed is sized 1462 $m^2$ and 21 APs where on during training phase in the whole testbed, out of which 11.2 APs, on average, can be detected at a given cell. The
measurements took place on the 11-th of April 2009 (Set 6).

Due to privacy issues we created a training map only in the main Hallway of FORTH and in the four smaller ones, inside ICS, and tested our algorithms only in these areas (Figure 3.14).

Again the same sequence of steps was followed like in the previous testbeds. The particle filter-based algorithm was not evaluated in this testbed, because we have already proved twice that it is not a robust algorithm. The performance of the rest of the algorithms in ICS is shown in Figure 3.15.

The confidence interval-based algorithm is the least efficient one as expected with 2 m median location error, while the decile- and quartile-based ones have a median error of 1.1 m. What is more, the maximum location error from 6.3 m, using confidence intervals, is decreased to 3 m, using deciles, which is more that 50% improvement on the accuracy of the system.

The decile- and quartile-based implementations results match in this testbed due to physical space distinctiveness (hallways minimize the range of possible error).

Despite the fact that the ICS testbed is almost the same size as the Cretaquarium one, the location error is almost half the one of the second testbed. ICS accuracy using deciles
is 1.1 m, while Cretaquarium using deciles as well is 2 m. This is obviously due to the fact that in the former case there are 21 APs in the area, while in the later one there are only 8 of them. This clearly shows the impact of the number of APs in the accuracy of the decile-based algorithm, and any system relaying on signal-strength measurements from APs.

### 3.3 Impact of number of APs

Our original hypothesis concerning the impact of the number of APs participating in the localization process is that the more APs participating the higher the accuracy of the system. In order to validate this idea, we run again the experiments in the ICS testbed, with the decile-based algorithm, while removing multiple APs.

#### 3.3.1 Removing most common APs

The first way to choose the APs removed from the voting process was the following. Firstly, we listed the APs in an descending order from the most common AP (appeared in the most training cells of the ICS testbed) to the most rare AP (appeared the least times in the...
same training cells). We know that the total number of APs in the ICS testbed is 21 (Figure 3.16), but we wanted to see how the accuracy would vary when one AP at a time would be eliminated, in the order mentioned, until only 12 APs participated in the voting process.

Figure 3.16: Percentage of coverage of the whole testbed, of APs in ICS. (Set 6)

The results are illustrated in Figure 3.17. As expected fewer APs result in lower accuracy.

Another important notice is that certain eliminated APs have a stronger impact on the accuracy than others, either improving it or worsening it (Figure 3.18). An outstanding example is that when 16 APs participate the 90% percentile of location error is 6.5 m, while in the case of 14 participated APs it is only 3.5 m. One would expect that the error would rise by eliminating this two APs, but their appearance discriminated a certain–wrong–position from another–correct–one, and it’s elimination resulted in the system being able to better match the run-time signature with the correct training one.

3.3.2 Removing least common APs

The above conclusion drove us to re-examine wether most or least common APs have the highest impact on the accuracy of a location-sensing system. A hypothesis formed was
that APs only appeared in few training position would clearly identify and discriminate them from the rest, in case a run-time scenario took place in one of them.

In this approach we listed the APs in an ascending order from the most rare AP
(appeared the least times in the same training cells) to the most common AP (appeared in the most training cells of the ICS testbed). Once again we wanted to see how the accuracy would vary when one AP at a time would be eliminated, in the order mentioned, until only 12 APs participated in the voting process.

The results of the same run-time scenario as before, are illustrated in Figure 3.19. The results are even more intuitive than before.

![Graph showing variation of performance of decile-based algorithm while eliminating least common APs in ICS.](image)

Figure 3.19: Variation of performance of decile-based algorithm while eliminating least common APs in ICS. (Set 6)

The first five APs (each one of them covering maximum 27% of the training area as Fig. reffig:im76 shows) we notice that have no impact on the accuracy, while the next two of them (covering 27–31%) result in decreasing the accuracy. The elimination of the following AP (covering 36.5%) and finally the last eliminated AP (covering 38% of the area), further increase the location error.

This approach shows no phenomena like in the previous one when APs with 62–99% covering area were removed from the voting process, due to the reasons mentioned above (Figure 3.20).
Chapter 3. Experimental results

3.3.3 Removing random APs

A final approach was to see the impact that randomly chosen APs were removed one at a time, and compare it to the previous results.

In order to randomly choose 9 APs to be eliminated a script in Matlab was ran and generated 9 random integers from 1 to 21, each one of them corresponding to an AP.

As Figure 3.21 shows, the location error rises accordingly to the number of APs removed from the voting process. However, there are still cases as the ones mentioned in the first approach. When 20 and 19 APs are used, we notice that the 90% percentile of location error is over 6 m and less than 6 m, respectively, whereas one would expect to see the opposite phenomenon (Figure 3.22).

3.4 Statistically improving accuracy

The Cretaquarium testbed is the biggest and the most challenging one, because it is an actual area that provides location based services to its visitors. Therefore, we attempted to optimize the accuracy of the methods described before by applying the methods that will
Figure 3.21: Variation of performance of decile-based algorithm while eliminating random APs in ICS. (Set 6)

Figure 3.22: Median and 90% percentile of location error of decile-based algorithm while eliminating random APs in ICS. (Set 6)

be described later on, only on this testbed. However, the same algorithms and procedures can be applied on the rest of the testbeds, too.
3.4.1 Removing Outliers

The signal-strength collected has some outlier values that possibly affect the accuracy of the methods. This is because those outlier values alter the statistical information representing the set of signal-strength collected. If those values are removed then those that are most likely to be the most common ones for this AP and the specific cell will remain and the compare between training and run-time values will be more accurate and trustworthy, because values created due to external unpredicted factors (outliers) will be removed from the signal timeseries.

In order to validate the original hypothesis we used two different methods of removing outliers. Firstly, the Quantitative Approach to Outliers [4] and secondly, the Boxplot method [37]. We used those two methods because they use different statistical data and the both are commonly used in bibliography as removing outliers methods.

Quantitative Approach to Outliers method considers as outliers the values not contained in the interval

\[ [mean - 2 \cdot std, mean + 2 \cdot std] \]

where \( mean \) is the mean value and \( std \) the standard deviation of the vector of signal-strength values.

There are two different Boxplot methods. Firstly there is the mild outliers removing Boxplot method that considers as outliers the values not belonging in the interval

\[ [x_{25} - 1.5 \cdot IQR, x_{75} + 1.5 \cdot IQR] \]

where \( x_{25} \) and \( x_{75} \) are the first and third quartiles, and \( IQR = x_{75} - x_{25} \).

Secondly, there is the extreme outliers removing Boxplot method where the formula changes to

\[ [x_{25} - 3 \cdot IQR, x_{75} + 3 \cdot IQR] \]

We decided to compare the methods described above in the decile-based algorithm
because as Figure 3.4 proved, it is the most efficient one, and we would like to see how much more improvement on the accuracy can be achieved.

Figure 3.23 shows that the maximum location error is decreased by 2.5 m using outlier removal methods. However, the median value in cases of no outlier removal, mild-Boxplot and quantitative approach to outliers (QAO) methods is 2 m, but it is decreased to 1.8 m with the use of extreme-Boxplot.

![Figure 3.23: Location error results comparing removing outlier methods efficiency, under extreme conditions. (Set 1)](image)

The mild-Boxplot method is more accurate than QAO as far as higher location errors are concerned. In the most cases the mild-Boxplot method creates tighter intervals, but the size of the interval containing the statistical information used (deciles) for each cell and each AP, depends on each vector of values. This is because both approaches to remove the outliers use different statistical data, both depending on the values of each vector. Mild-Boxplot uses the first and third quartiles in order to compute the interval of values that contain the signal-strength used to compute the deciles used in the decile-based method. On the other hand, the QAO method uses the mean and the standard deviation values.
3.4.2 Incorporating conditional probability

If we could minimize the searching area of the algorithm, then the accuracy would rise, but in order to do that we should divide the testbed into smaller sub-areas. Moreover, our algorithm should have knowledge of the sub-area the user is located in, so as to search only in this area for the user’s exact position. This could be implemented with the use of RFID or IrDA technology (described later on), or any other method that could identify a user in a certain area.

In other words, suppose we know the sub-area (Z) the user is in, we can compute a conditional probability using the decile-based algorithm in order to find the cell the user is, given the id of the sub-area (or zone in the case of Cretaquarium, as shown in Figure 3.8) he/she is in. The following formula describes exactly this idea and Table 3.1 explains the symbols used,

\[ P(C|Z) = \frac{P(C \cap Z)}{P(Z)} = \frac{P(Z|C) \cdot P(C)}{P(Z)} \]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(C/Z)$</td>
<td>user’s probability to be in cell $C$ given zone $Z$</td>
</tr>
<tr>
<td>$P(C)$</td>
<td>user’s probability to be in cell $C$ from training set</td>
</tr>
<tr>
<td>$P(Z)$</td>
<td>user’s probability to be in zone $Z$</td>
</tr>
<tr>
<td>$P(Z/C)$</td>
<td>user’s probability to be in zone $Z$ given cell $C$</td>
</tr>
</tbody>
</table>

We run two series of experiments for this case. Firstly, the case where there are not too many visitors in the testbed of the Cretaquarium (normal conditions, up to 100 visitors simultaneously) and secondly the case where there are far too many visitors (extreme conditions, over 250 visitors at the same time).

The reason why we did this was to explore the impact of human interference in the performance of the positioning methods. We expect that in normal conditions the accuracy
is going to be higher than in extreme ones.

Figure 3.24 shows the accuracy in the testbed of Cretaquarium under normal conditions, using measurements collected on the 18-th of February and 15-th of May 2008 (Set 7). As we can see the percentile approach (deciles) with conditional probability results in highest accuracy with median location error 2 m.

![Figure 3.24: Efficiency of confidence interval- and decile-based methods with and without use of conditional probability (normal condition). (Set 7)](image)

On the other hand, Figure 3.25 shows the accuracy in the testbed of Cretaquarium under extreme conditions. As we expected the percentile approach with conditional probability gives the best results with median location error 2.2 m.

Conditional probabilities have a major impact on the accuracy, but an even greater one on confidence interval-based approach. This is because the confidence interval-based method can be improved using multiple methods and it is not a robust method. One the other hand, it has less impact on percentile-based approach (deciles), due to the fact that it is a more robust method, as we have already mentioned before.
3.4.3 Assigning weights to APs

It has been noticed that each AP may have a different impact on the estimation of the user’s position. This is proven by Figure 3.26, where we decided to exclude one AP at a time out of the 8, in order to see the impact each one of them has in the final location error of the positioning algorithm (using the confidence interval-based implementation). We conducted new experiments on the 24\textsuperscript{th} of July 2008 at 12:30 - 14:00 (Set 8) in the Cretaquarium for that purpose, under extreme conditions. Moreover, we use the positions with higher location error in order to see more clearly the impact of each AP as well as the impact of the weights on the accuracy of the system.

The higher the location error by excluding one AP, means that this AP has a higher impact on the location error.

This means that adding weights to each AP’s vote is believed to result in higher accuracy of our methods. The weights we are going to compute will result in each AP having a different impact on the final result of the algorithm. Our goal is each time that the vote of each cell is computed, to take into higher consideration the AP we believe to be the most accurate and trustworthy one.

The next step we took was to divide the testbed in two areas, one with \textit{Obstacles} in it and another one with \textit{NoObstacles}. We wanted to test if the impact of each AP would
Figure 3.26: Location error results comparing the impact of each AP on the accuracy. (Set 8)

be the same in both areas, and same as in the whole testbed as in Figure 3.26.

Figure 3.27: Location error results comparing the impact of each AP on the accuracy in the Obstacle area. (Set 8)

In Figure 3.26, we can see that the AP with the highest impact on the system’s accuracy is AP 4, while the AP that if taken into consideration worsens the accuracy is AP 1. On the other hand, from Figures 3.27 and 3.28 we can see that in each area of the overall testbed the APs do not have the same behavior. It is obvious that in the NoObstacles area the AP with the higher impact is AP 4, whereas in Obstacles area is AP 8.

These figures show that each AP has a different impact on each area of the testbed which means that it is not a good idea to have weight per AP regardless of the location
Weights per AP per cell  We formed an optimization problem taking all the factors mentioned above into consideration in order to assign weights per AP per every training cell. We need to find a vector $< w >$, such that a distance $d$ is minimized. This distance is defined below.

\[
\text{Find } < w > \\
\text{Minimize } d = \sum_{i=1}^{k} \sum_{j=1}^{n} \| T_{i,j} - w_{i,j} \cdot R_{i,j} \|
\]

Where

$k$ : number of samples in the training set

$n$ : number of APs

$T_{i,j}$ : statistical value for training set T, i-th cell and j-th AP

$R_{i,j}$ : statistical value for training set R, i-th cell and j-th AP

$w_{i,j}$ : weight for i-th cell and j-th AP

Two training sets under same conditions $\{ T, R \}$ are used, that came out by splitting the original training set in half, and one run-time set $\{ U \}$ is used.

As we can see the distance $d$ is a sum of positive numbers which means that it is minimized when
\[ d = 0 \Rightarrow \sum_{i=1}^{k} \sum_{j=1}^{n} \| T_{i,j} - w_{i,j} \cdot R_{i,j} \| = 0 \Rightarrow T_{i,j} - w_{i,j} \cdot R_{i,j} = 0, \forall i, j \Rightarrow w_{i,j} = \frac{T_{i,j}}{R_{i,j}}, \forall i, j \]

Figure 3.29: Efficiency of confidence interval-based method with and without use of weights (extreme condition). (Set 8)

The training time fingerprints used were the same as in extreme conditions, as well as the run-time ones.

Accuracy improved from 5.14 m to 2.2 m, as Figure 3.29, and hypothesis of weight per AP per position improving accuracy is validated.

We can see that the performance of this algorithm is the same as if conditional probability was used. The only difference is that in conditional probability-based method there are no outliers in addition to the weights per AP per cell-based. In the first case the maximum location error is 6 m, whereas in the second one is almost 15 m.

### 3.4.4 Kullback–Leibler divergence

In probability theory and information theory, the Kullback–Leibler divergence (KLD) [29, 27, 28] (also information divergence, information gain, or relative entropy) is a non-
commutative measure of the difference between two probability distributions \( P \) and \( Q \). Typically \( P \) represents the “true” distribution of data, observations, or a precise calculated theoretical distribution. The measure \( Q \) typically represents a theory, model, description, or approximation of \( P \).

In our scenario, we have our training phase measurements and the run-time ones. The run-time timeseries per AP is transformed into a probability distribution \( P \) that is compared iteratively to all probability distributions \( Q \) composed by the training time measurements of AP per cell. These probability distributions use bin size equal to 1 and number of bins equal to \( \text{maximumValue} - \text{minimumValue} \) in vector, due to the fact that the values of signal-strength we use are integer values and \( \text{bin size} = 1 \).

In the case where all measurements of a set are equal to the same value, no probability distribution can be formed. Then a new Gaussian set of values is generated with mean value (mean) the one same value of the one repeated in the original set, and standard deviation value (std) as small as possible.

The std alters the original set. The higher the std, the higher the alternation of the original measurements. A typical example that can prove that is the following. The range of the new distribution to be generated is \([\text{mean-std}, \text{mean+std}]\). If the mean value of the original set is, for example, 80 the following is going to happen:

1. for \( \text{std}=1 \) the range of values in the new set is about \([79, 81]\)
2. for \( \text{std}=0.1 \) the range of values in the new set is about \([79.9, 80.1]\)
3. for \( \text{std}=0.01 \) the range of values in the new set is about \([79.99, 80.01]\)

The nature of our data is that they are integer values. This means that for \( \text{std} \) equal to 1, 0.1 and 0.01 the range is \([79, 81]\), \([79, 80]\) and \([79, 80]\), respectively. The tighter the distribution, the closer we are to the original data collected. In other words, the higher the value of \( \text{std} \), the wider the distribution becomes and the more the original data is altered.

We need to choose the value of \( \text{std} \) that returns the tightest and closest to the original data. Due to the fact that our data are integer values, we see that for all values of \( \text{std} \) smaller than 0.1 (\( \text{std} < 0.1 \)), the range of values returned equals the returned values of
when std=0.1, and the generated data cannot become more like the original measurements set.

![Figure 3.30: Comparing KLD-based approach to the confidence interval-, decile- and quartile-based algorithms. (Set 1)](image)

From Figure 3.30 one can see that KLD-based approach is much more efficient than the confidence interval one, because much more information is taken into consideration before reaching to a result. The median location error of KLD-based is 2 m same as in the decile-based approach, compared to 3.6 m median location error in the case of confidence interval- and 2.2 m in quartile-based methods. Despite the fact that this method has the same median location error as decile-based one, in general (apart from the 50% percentile) the location is always much higher than both the decile- and the quartile-based methods.

One would expect the KLD-based approach to be the most accurate one due to the fact that the more the information used, the higher the accuracy, because KLD uses all data collected separately. However, the outliers have a negative contribution to the estimation of the real position of a user, as shown in previous sections. The fact that KLD uses all data collected means that it also uses the outliers, and therefore the accuracy is lower than the decile-based approach and the maximum error is about 13 m opposed to 7 m in the
decile-based method.

A different version of this approach is to add symmetry to the original KLD method. This would mean that not only set P (representing the probability function of run-time data collected) is compared to Q (representing the probability function of training phase data collected) as mentioned in the beginning of this section, but sets Q are also compared to P.

Figure 3.31 shows that the two methods do not vary significantly. However, it is noted that extra information decreases the accuracy instead of increasing it, due to the fact that still outliers are taken into consideration and affect the accuracy of the algorithm. The median location error in the KLD-based approach is 2 m, as mentioned before, compared to 2.2 m in the KLD symmetric-based method.

![Figure 3.31: Comparing KLD- and KLD symmetric-based approaches. (Set 1)](image)

### 3.5 Conclusions

There are various implementations of CLS [44], described in Table 3.2 below,

In the sections above we have shown, in our preliminary research, that the particle filter-based algorithm is not a robust method, due to the fact that it is based on a random initialization that for the same input arguments, different positions are reported from the system. Extra measurements and in depth analysis of the algorithm would provide more
Table 3.2: Implementations of CLS.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Only APs vote</th>
<th>Peers &amp; APs vote</th>
<th>Distance computed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidence interval-based</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>p2p-based</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Decile-based</td>
<td>Deciles</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Quartile-based</td>
<td>Quartiles</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Particle filter-based</td>
<td>Particle filters</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

trustworthy and secure results as far as the particle filters are concerned.

Therefore on the following table (Table 3.3) we compare the remaining three algorithms (i.e., confidence interval-, quartile- and decile-based).

Table 3.3: Median performance of each algorithm in each testbed.

<table>
<thead>
<tr>
<th></th>
<th>Confidence intervals</th>
<th>Quartiles</th>
<th>Deciles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cretaquarium</td>
<td>3.6 m</td>
<td>2.2 m</td>
<td>2 m</td>
</tr>
<tr>
<td>TNL</td>
<td>2.3 m</td>
<td>2.2 m</td>
<td>1.6 m</td>
</tr>
<tr>
<td>ICS</td>
<td>2 m</td>
<td>1.1 m</td>
<td>1.1 m</td>
</tr>
</tbody>
</table>

It is obvious that in all cases, the confidence interval-based method is the least accurate while the quartile-based one always follows and the highest accuracy is at all times achieved by the decile-based algorithm. As we have already explained, this is because of the number of statistical data describing the signal-strength collected from each AP in each position and used to compute the users’ positions. In the first case only two values (lower and upper bound of a confidence interval) are used, in the second one four (25%, 50%, 75% and 100% percentiles), whereas in the third one ten values (10%, 20%, ..., 100% percentiles) are used, and therefore the highest accuracy is achieved.

Moreover, the impact of APs is explored and, as shown above, the more the APs in a testbed, the lower the location error. This is because when more APs are used there is even more information used to describe the signature of a specific cell in detail. In this
Another important issue is the percentage of APs covering a specific cell of the testbed, and the impact it has on the overall accuracy of the system. When we mention the percentage of APs coverage we mean the mean number of APs in any cell divided by the total number of APs in the testbed. We can see that in the following table (Table 3.4). We notice that the higher the percentage of APs, out of the total number of APs, covering any area of the testbed, the lower the accuracy. For example in the Cretaquarium the percentage of APs covering any given cell is 5.7/8=71% (mean number of APs in any cell divided by the total number of APs in the testbed) and results in 2m median location error using the decile-based method, whereas in the ICS testbed, where the percentage is 11.2/21=53% the median location error is 1.1 m.

Table 3.4: Impact of mean percentage of APs covering a cell of the grid representation of each testbed.

<table>
<thead>
<tr>
<th>Testbed</th>
<th>Decile median location error</th>
<th>Percentage of APs covering a cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cretaquarium</td>
<td>2 m</td>
<td>71%</td>
</tr>
<tr>
<td>TNL</td>
<td>1.6 m</td>
<td>66%</td>
</tr>
<tr>
<td>ICS</td>
<td>1.1 m</td>
<td>53%</td>
</tr>
</tbody>
</table>

This is because of the fact that if just a small percentage of the total number of APs is covering an area, then it would be easier discriminated from other cells that would be covered again by a small percentage of APs, which would be composed by another set of APs. In this way, CLS voting algorithm does not add votes in cells that would be more similar, if the same APs are apparent in two or more cells, and the estimated position is most likely to be the one, or very close to the one the user is really in.

Furthermore, as far as the p2p-based approach is concerned, it is true to say that the more the number of extra peers, the higher the accuracy. This would be expected if we realize that peers are like APs, adding extra information. Based on the previous conclusion we see that apart from the number of APs, the number of extra peers has the same great impact on the accuracy of the confidence interval-based algorithm, as Table 3.5 shows.

Another conclusion we came to is the fact that the physiology of the testbed is a critical factor as far as the performance of IEEE 802.11 localization systems are concerned. The
material that the testbed is made of can cause reflection, scattering etc of the signal and therefore “confuse” the system. This happens because we rely on the signal-strength measurements of APs, and whether an AP covers a specific area or not in order to accurately compute the position of a mobile device. When scattering or other familiar phenomena happen, an AP may cover an area that shouldn’t, or should cover with low signal-strength and the signal is higher than expected. This would confuse the system in the sense that a nearby cell that is covered by the same APs, but was discriminated by the other cell only by the signal-strength measurement of the specific AP, would now have no difference, and the system would assign the same number of votes in both cells, which means that the more far away the two cells are, the higher the location error the system is going to report.

Moreover, the more the APs participating in the localization procedure the higher the accuracy achieved.

As mentioned before, the number of APs has a great impact on the accuracy, but the placement of APs in the testbed has a great impact as well. We can understand this by taking a look at Table 3.3, where Cretaquarium (1760 $m^2$) and TNL (84 $m^2$) have about the same location error (even exactly the same in the case of quartile-based method used) despite the great difference in size. In both cases there are about the same small number of APs covering the testbeds, in addition to ICS, but in the case of TNL the APs are randomly placed with the only objective the high satisfaction of those who use the WiFi. On the other hand, in the case of Cretaquarium testbed, the APs were placed with the objective of how their placement would increase the performance of a IEEE 802.11-based location-sensing system, along with the satisfaction of the users. This means that in this scenario both the interference between APs and the scattering effects were eliminated, resulting in higher accuracy regarding the size of the testbed, as we explained before.

Furthermore, after analyzing various ways we tested to improve the accuracy, we can say that all methods had a significant impact on minimizing the location error. We used run-time positions in the Cretaquarium testbed that had high location error in order to
explore, how high errors can be improved using each of the methods above.

In the case of removing outliers, the best result was outcome of the extreme outliers removing Boxplot method, which decreased the median location error using decile-based method from 2 m to 1.8 m.

In the case of conditional probability-based method, a major impact on the accuracy was imposed, but an even greater one on confidence interval-based approach (about 1m decrease of location error) than on decile-based one (no more than 0.2 m improvement of accuracy).

As far as weights are concerned, we have shown that it is not efficient to use weights per AP in a certain testbed, but weights per AP per position. We formed a minimizing optimization problem in order to compute these weights per AP per position, and the result was to improve the error from 5.14 m to 2.2 m.

Finally, we have shown that the Kullback – Leibler Divergence, in its original form, cannot improve the location error in the Cretaquarium.
Chapter 4

Adding extra technologies to IEEE 802.11

4.1 RFID

After exploring various ways to improve the accuracy of the system, we decided to move on incorporating extra technologies in the positioning methods, apart from IEEE 802.11. Therefore, we tried to use the RFID technology in order to be able to implement the conditional probability-based approach that was described in Chapter 3.4. This means that if we have divided our testbed into smaller sub-areas (like the zones in the Cretaquarium testbed – Figure 3.8), we can use a single RFID reader for each area that will identify it. For example if a user is located in a certain sub-area that is identified by RFID reader with a specific ID number, and this information reaches the user, then algorithm will not need to search in the whole testbed in order to locate the user, but will automatically eliminate all positions outside this identified sub-area.

As proven before, using conditional probability-based approach avoids high location error – outliers.

4.1.1 Technology description

Radio-frequency identification (RFID) is an automatic identification method, relying on storing and remotely retrieving data using devices called RFID tags or transponders. The
technology requires some extent of cooperation of an RFID reader and an RFID tag.

**RFID tag**

An RFID tag is an object that can be applied to or incorporated into a product, animal, or person for the purpose of identification and tracking using radio waves. Some tags can be read from several meters away and beyond the line of sight of the reader.

RFID tags come in three general varieties: passive, active, or semi-passive (also known as battery-assisted or semi-active) and beacon types. Passive tags require no internal power source, thus being pure passive devices (they are only active when a reader is nearby to power them by wireless illumination), whereas semi-passive and active tags require a power source, usually a small battery. Beacon tags transmit autonomously with a certain blink pattern and do not respond to interrogation.

To communicate, tags respond to queries generating signals that must not create interference with the readers, as arriving signals can be very weak and must be differentiated. Besides backscattering, load modulation techniques can be used to manipulate the reader’s field. Typically, backscatter is used in the far field, whereas load modulation applies in the nearfield, within a few wavelengths from the reader.

Passive RFID tags have no internal power supply. The minute electrical current induced in the antenna by the incoming radio frequency signal provides just enough power for the CMOS integrated circuit in the tag to power up and transmit a response. Most passive tags signal by backscattering the carrier wave from the reader. This means that the antenna has to be designed both to collect power from the incoming signal and also to transmit the outbound backscatter signal. The response of a passive RFID tag is not necessarily just an ID number; the tag chip can contain non-volatile data, possibly writable EEPROM for storing data.

Passive tags have practical read distances ranging from about 11 cm (4 in) with near-field (ISO 14443), up to approximately 10 meters (33 feet) with far-field (ISO 18000-6) and can reach up to 183 meters (600 feet) when combined with a phased array. Basically, the reading and writing depend on the chosen radio frequency and the antenna design/size. The lack of an onboard power supply means that the device can be quite small: commercially
available products exist that can be embedded in a sticker, or under the skin in the case of low frequency (LowFID) RFID tags.

Unlike passive RFID tags, active RFID tags have their own internal power source, which is used to power the integrated circuits and to broadcast the response signal to the reader. Communications from active tags to readers is typically much more reliable (i.e. fewer errors) than those from passive tags due to the ability for active tags to conduct a "session" with a reader.

Active tags, due to their onboard power supply, also may transmit at higher power levels than passive tags, allowing them to be more robust in "RF challenged" environments with humidity and spray or with RF-dampening targets (including humans and cattle, which contain mostly water), reflective targets from metal (shipping containers, vehicles), or at longer distances: Generating strong responses from weak reception is a sound approach to success. In turn, active tags are generally bigger (due to battery size) and more expensive to manufacture (due to price of the battery). However, their potential shelf life is comparable, as self-discharge of batteries competes with corrosion of aluminated printed circuits.

Many active tags today have operational ranges of hundreds of meters, and a battery life from several months to 10 years. Active tags may include larger memories than passive tags, and may include the ability to store additional information received from the reader.

Semi-passive tags are similar to active tags in that they have their own power source, but the battery only powers the microchip and does not power the broadcasting of a signal. The response is usually powered by means of backscattering the RF energy from the reader, where energy is reflected back to the reader as with passive tags. An additional application for the battery is to power data storage.

If energy from the reader is collected and stored to emit a response in the future, the tag is operating active. Whereas in passive tags the power level to power up the circuitry must be 100 times stronger than with active or semi-active tags, also the time consumption for collecting the energy is omitted and the response comes with shorter latency time. The battery-assisted reception circuitry of semi-passive tags leads to greater sensitivity than passive tags, typically 100 times more. The enhanced sensitivity can be leveraged as increased range (by one magnitude) and/or as enhanced read reliability (by reducing bit
error rate at least one magnitude).

![Diagram of RFID communication](image)

Figure 4.1: Communication between RFID reader and tag.

**RFID reader**

An RFID reader typically contains a module (transmitter and receiver), a control unit and a coupling element (antenna). The reader has three main functions: energizing, demodulating and decoding. In addition, readers can be fitted with an additional interface that converts the radio waves returned from the RFID tag into a form that can then be passed on to another system, like a computer or any programmable logic controller. Anti-Collision algorithms permit the simultaneous reading of large numbers of tagged objects, while ensuring that each tag is read only once. RFID readers can connect to a server either wirelessly or through a wire.

A problem as far as RFID readers are concerned is their size. In case of passive tags used the size of a reader can be very big in order to achieve high range. For example, in case of 2 m reader range and passive tags, a typical solution is the size of a door, and costs at least 900, at the time being. On the other hand when active tags are used, the reader can be of size 15 cm x 15 cm, its range can reach a distance of 100 m, and the minimum cost is about 500.

Readers of small distances (about 10 - 20 cm) are much cheaper (about 150 – 200) and may use passive tags. Such small distance could not be of use in our project, due to the fact that this would require the interaction of the user (to pass the tag by the reader), whereas we want to create an intelligent testbed that would require the minimum intervention of the user.
Use of RFID

We decided to use the M200 Reader (12x12 cm$^2$) from RF Code company (Figure 4.2), that is equipped with two antennas each, using wired connection to the server, and active tags (5x3 cm$^2$). We chose this device because it is small, flexible as far as its range is concerned (we can set the desired range from a couple of meters to hundreds) and has a reasonable price regarding its attributes.

![RF Code M200 reader device.](image)

The approach that we decided to follow was to have stable RFID readers (one in each sub-area), and mobile RFID tags for the users to carry with them. In this way, each user is associated to only one tag, and when the user passes by a specific reader, the tag will be identified and the user will be let know immediately, in order for the proposed algorithms to be executed, taking this information into consideration.

The company of the devices used provides an API of functions that can be used in order to retrieve information from the readers. In this way, a Java program was created that monitors all the tags identified from each RFID reader.

The main algorithm of the program running on the server is the follows. The server,

1. connects to all readers
2. sets the desired range of each one of them
3. resets the events in cache of the readers
4. notices an event (new identification of a tag) from a reader
5. retrieves the IP of the owner of the tag identified, from a database associating each tag ID to the IP of the user owning it
6. creates a TCP socket that connects wirelessly the server to the mobile device (laptop or PDA of the user)

7. sends a message to each user containing the ID of the reader that identified their tag

The steps that the user – client follows after the procedure above has finished are:

1. the proposed algorithm can reduce the searching area in testbed, to the range of the reader that identified the specific user

2. the algorithm returns the estimated position of the user

This program in implemented in Java, apart from the client-server model, that is implemented in C. The main Java program uses the exec() command to call a C executable file, with two arguments (IP of user, ID of reader identified user’s tag), in order to inform the local CLS instance, using any of the proposed positioning methods mentioned before, as far as the RFID reader ID that identified a specific tag is concerned.

The selection of number of RFID readers needed is another important issue that arises. Generally, and in an ideal, and rectangle shaped environment, the following would apply.

If we had one reader per room with quite small range (5-6 m), we would be able to define the room – lab the user carrying the tag is, depending on the ID of the reader that has identified him/her. This is a neither intelligent nor economic enough solution. Therefore, we want to achieve the same result with the minimum number of readers.

We have learned from Combinatory Theory that if we want to distinguish up to 2 different conditions we need 1 devices \( (2 = 2^1) \), up to 4 conditions 2 devices \( (4 = 2^2) \), up to 8 conditions 3 devices \( (8 = 2^3) \), and so on. In other words, in the following formula \( k \) gives the optimal number of devices needed, depending on \( n \) number of rooms to be defined. If \( k \) is not an integer according to the following method, it should be rounded up to the next integer.

\[
\begin{align*}
    n & \leq 2^k \\
    \log \left( n \right) & \leq k \cdot \log \left( 2 \right) \\
    k & \geq \frac{\log\left( n \right)}{\log\left( 2 \right)}
\end{align*}
\]
In a simple scenario where we should discriminate 3 rooms (labs) one next to the other, Telecommunication and Networks Lab (TNL), and 2 areas of the Information Systems Lab (ISL), as in Figure 4.3. In this case, we would need \( n = 3 \), because

\[
k \geq \frac{\log(3)}{\log(2)} = 1.59
\]

This means that the minimum number of devices needed is \( k = 2 \). The different condition – combinations are the following depending on the operation or not of each device: \{ R1\_on.R2\_off, R1\_off.R2\_on, R1\_on.R2\_on, R1\_off.R2\_off \}.

The only problem left is to define the position of each of the \( k \) devices, and each condition – combination to a room. This highly depends on the physiology of the space, too. Due to the fact that we need both readers on in order to identify a room, and the rooms in our case are next to each other, the only way to place the readers is to have R1 in TNL and R2 in ISL\_2, as the following figure illustrates. In this way, the placement of the devices has resulted in this correspondence:

- \( \text{R1\_on.R2\_off} : \text{TNL} \)
- \( \text{R1\_off.R2\_on} : \text{ISL}_2 \)
- \( \text{R1\_on.R2\_on} : \text{ISL}_1 \)
- \( \text{R1\_off.R2\_off} : \text{all other places} \)

![Figure 4.3: Placement of RFID readers to discriminate three labs.](image)

In other words, the range of each reader should cover the size of just 2 rooms, so that the third will only be covered by the other reader. The same procedure can be followed for more rooms, theoretically.
4.1.2 Performance analysis

Having taken all the above into consideration we conducted a series of experiments in a hallway at ICS-FORTH. The pink dots, in the following Figure 4.4, reflect position of the RFID reader (with horizontal antennas) and the blue ones the position of each tag used. The physiology of the specific testbed is challenging due to the fact that a lot of metal, glass and thin walls are apparent, that maximize the diffraction, reflection and scattering phenomena in it.

The RFID readers used are RF Code M200 readers and active tags of the same company. The reader used had an option for the user to set its range manually. The maximum range is 300 ft and the reader’s range can be set from 1 to 8 range. These values are the default that the manufacturer has divided the maximum range into. We want to test the efficiency of each range value, in order to find the optimal one for our scenario. The original idea was that in one hallway only one reader should be active. In this way, the system would be able to distinguish the hallway the user walks by, by the RFID reader that identifies him/her.

In Figure 4.4 experiment 1 is illustrated and the desired results would be to find the optimal reader’s range that the reader identifies all tags apart from 9 - 12 (that belong to another hallway).

The results from this experiment, as well as the rest of them, validated the original hypothesis that the reflections, scattering and similar phenomena, make it impossible to distinguish a hallway from the reader identifying a tag.

This is because even in the scenario of the first experiment the reader identifies tag number 10 even with the minimum reader’s range (=1). In the case of the remaining scenarios, tags numbered 1 and 9 where identified with minimum reader’s range 7 and 4 in the second experiment and with 8 and 3 in the third one, respectively.

These results clearly indicate the strong existence of the phenomena mentioned above. This makes us reconsider the original idea of each RFID reader “representing” one hallway. Another implementation closer to the conditions of the testbed, in the same direction as the previous one, would be to empirically find the cells of the grid that each range of the specific reader covers. In this way, when a reader is identified from a reader with known range,
the WiFi-enabled positioning method search will be limited in the set of cells belonging in the area covered by this reader, instead of just a hallway that the original idea suggested.

The first step in order to extend IEEE 802.11 with RFID technology is for us to use it in the same way we used WiFi, in the TNL testbed. The RFID readers were used like the APs, and the tags like the mobile wireless card of the user. This means that again signal-strength was collected just by the RFID readers, transformed in the preferred statistical value (i.e. confidence interval), and created a new training map that is composed by RFID readers’ signal-strength. Now an entry of the training map would look like this:
where \((x,y)\) are the coordinates of the specific training cell, and \([rfSS_l^K, rfSS_u^K]\) is the confidence interval of the \(K\)-th RFID antenna. The algorithm of the original CLS is not altered. The same is for run-time signature as well.

Figure 4.5 shows the deployment of RFID technology in TNL. At first the same methodology as in WiFi-based algorithm was used, as far as training and run-time phases are concerned. Red dots represent the cells where run-time measurements were collected, the black ones the training-phase cells and finally the yellow ones are the position of the two RF-readers. Each of the readers has two antennas, as mentioned before.

![Figure 4.5: RFID technology deployment in TNL.](image)

In Figure 4.6, we see the location error of RFID-based approach. The measurements took place on the 25-th of July 2008 in the TNL testbed (Set 9). It is smaller in the area on which the readers are placed at the corners, as seen on the cyan curve. The other curves show the location error on the whole testbed. The red curve is for the experiment where measurements from 2 of the 4 antennas were taken into account. The blue curve shows the location error when measurements from 4 antennas are used. For both situations, the location error is almost the same.

This means that the system performs better if the readers are not placed close by,
and in the center of the testbed, when the location error is about 1.5 m, but when they surround the testbed.

As mentioned before, TNL is an area of $85m^2$ and the median location error given by the positioning method as mentioned in earlier research, when just WiFi APs and the confidence interval-based approach are used is 2.25 m, whereas when just RFID technology is used the median location error reaches 3 m, as Figure 4.6 shows.

![Figure 4.6: Location error results comparing RFID-based in different areas of the TNL. (Set 9)](image)

The next step was to integrate WiFi technology with RFID. There were two different aspects as far as the type of integration that should be done:

1. **WiFi extended with RFID**: In this scenario we use the RFID technology in the same way we used WiFi. The RFID readers were used like the APs, and the tags like the mobile wireless card of the user. This means that again signal-strength was collected by both the APs and the RFID readers, transformed in the preferred statistical value (i.e. confidence interval, percentiles), and created a joint training map that is composed by both AP and RFID reader signal-strength. Now an entry of the
training map would look like this:

\[ x, y, SS^1_l, SS^1_u, SS^2_l, SS^2_u, ..., SS^N_l, SS^N_u, rfSS^1_l, rfSS^1_u, ..., rfSS^K_l, rfSS^K_u \]

where \((x,y)\) are the coordinates of the specific training cell, the confidence interval of the \(N\)-th AP is represented by \([SS^N_l, SS^N_u]\) and \([rfSS^K_l, rfSS^K_u]\) is the confidence interval of the \(K\)-th RFID antenna. The algorithm of the original algorithm is not altered. This means that in this method the RFID technology reports the run-time signal-strength values between each antenna and the tag.

2. WiFi using RFID as conditional probability: In this case the RFID technology is used in order to give extra information to the user regarding the subarea of the testbed that he/she is in. In other words, when the user is identified by a RFID reader, it means that he/she is inside the specific reader’s range. The original positioning algorithm takes this as an input in order to minimize its searching area in the reader’s range. So, instead of the system searching the whole testbed to find the user’s exact position, it only searches in the boundaries of the reader’s cover area. This means that in this method the RFID technology only reports the subarea of the testbed the user is in.

The Figure 4.7 shows how the efficiency of each method differs with each other. Higher accuracy is given by WiFi extended with RFID with a median location error of 1.8 m, decreasing the error of the original method by about 0.5 m. We expect the decrease to be much higher in bigger testbeds (i.e. the whole ICS, or the Cretaquarium). WiFi using RFID as conditional probability follows with median location error of 2 m and the same maximum error as the previous method. This method performs slightly worse but there is no need for extra training if the RFID technology as the first method requires, which is very time-consuming. To continue, WiFi- (original method) and RFID-based methods follow as expected.

Later on, we wanted to test the performance of the algorithm in a bigger and more challenging testbed like the whole ICS (1020\(m^2\)), excluding the main Hallway of FORTH, outside ICS. Due to the fact that the testbed is much bigger than the one of TNL (85\(m^2\))
it was not considered a good idea to do again a training map of RFID signal-strength measurements, because it would be very time consuming. What is more collecting signal-strength measurements during run-time is a great overhead and adds great delay to the system.

Measurements have shown that while the positioning algorithm, using just WiFi or bluetooth, needs about 3 seconds to report an estimated position for the user, RFID adds extra delay. In the case of WiFi along with RFID as conditional probability the extra overhead is due to the client server model, where a central PC reports to the mobile user the ID of the RFID readers that have identified him/her. On the following the user has to read the file and then proceed with the estimation of the user’s position. This process can make the system take up to 4-5 seconds per estimation. On the other hand WiFi extended with RFID has to communicate with all readers and retrieve the signal-strength measurements between each reader and the specific user’s tag. This takes extra time and can result in the system needing about 7-8 seconds per estimation.

It is obvious that 8 sec are too much for a real-time location based system, that this method is going to be used, that needs a user’s position in order to provide information.
to him/her or take any other action depending on his/her position. It is most likely that
the user will have changed position in less than 8 sec and the position that the system
will report will then be an old one at that point. In other words, despite the fact that the
results may be slightly better, the time needed makes WiFi extended with RFID not an
ideal solution for a real-time application.

Therefore, we decided to only test WiFi along with RFID as conditional probability in
the testbed of ICS at FORTH. Due to the fact that the ICS is of much bigger size that
the TNL, where 2 readers were used, we chose to use 3 readers, as Figure 4.8 illustrates,
and the main goal we tried to achieve was to identify in which corner of the hallways is
the user at each time.

The red reader is R1, the green one R2 and the blue one is R3, and the colored lines in
Figure 4.8 intent to show the boundaries of the range area of each RFID reader. In this
way a user is supposed to be in the up right corner by not being identified by reader R1,
or in the up left corner by not being identified by reader R2 and so on.

Figure 4.8: Illustration of ICS and the range of three RFID readers. (Set 6)

If we would like to know the exact position the system returns in the testbed described
above, and compare it with the real position of the user, we would get the following results
for 30 random positions close to boundaries of the range of all three RFID reader’s (white
area in the Figure 4.8). The reason why we chose these positions was to see the impact of
the RFID technology in extreme situations, like in positions that could confuse the system.
Positions closer to the “borders” of the range of each RFID reader are such cases that may impose problems to the implementation of the algorithm in case a reader does not identify correctly a tag (user), because they will be classified to belong in another zone rather than the correct one.

We collected some measurements in order to be able to compare all methods by importing the same input signal-strength arguments, throughout the whole testbed, like in a normal path that any user would follow in each hallway (Scenario 1).

Figure 4.9 shows that in a normal route of a walking user he/she is unlikely to be in one of the previously tested positions. This means that the user will be in a position that belongs to a bigger zone covered by one or a set of RFID readers. This zone is very big compared to the average error of the algorithms which makes it impossible for the RFID technology (with the specific range of readers) to minimize the average error. Only if the range of the readers was lower than the average error, a high decrease of the location error would be noticed.

![Figure 4.9: Location error results comparing WiFi- to WiFi&RFID-enabled on path positions (Scenario 1). (Set 6)](image)

We conducted extra measurements in a new use case scenario (Scenario 2), where the user walked through corners, close to walls, steel or glass, on the 11-th of April 2009 (Set 10). We expect that a positioning system will be challenged in such positions. Moreover
the period of time that the experiment took place was 7 months since the last update of the training set. Various changes have taken place in the actual physical space and even new APs were active, imposing the interference factor on the signal-strength. Taking into consideration these conditions we expect that the results would not be satisfying. The results illustrated in Figure 4.10, show that RFID technology with such range and placement of the RFID-readers cannot improve the accuracy of the algorithm, since their range is still higher than the location error, even in this Scenario.

The conclusion of this section is that the RFID technology has no impact in the accuracy of the method. Only if more RFID readers with lower range were placed in the testbed, higher accuracy would be achieved. This means that the lower the range of the reader the higher the accuracy a positioning system may achieve. Because placing more RFID readers would be much more expensive, we decided to use another technology providing lower covering range with less cost, the IrDA technology.

![Figure 4.10: Location error results comparing WiFi- to WiFi&RFID-enabled on path positions (Scenario 2). (Set 10)](image)

### 4.2 IrDA

The Infrared Data Association (IrDA) defines physical specifications communications protocol standards for the short-range exchange of data over infrared light, for uses such as
personal area networks (PANs).

IrDA is a communication system based on infrared light. It is commonly used in mobile devices for cheap point-to-point communication. Digital cameras, mobile phones and laptops are just a few examples of devices that often use IrDA.

The IrDA Beacon family offer high flexibility and portability, is used as inconspicuous and reliable location-aware solution, and has extremely low power consumption as base for very long, maintenance-free mobile service

### 4.2.1 Technology description

1. IrDA is a very short-range example of free space optical communication.

2. IrDA interfaces are used in palmtop computers, mobile phones, and laptop computers (most laptops and phones also offer Bluetooth, but it is now becoming more common for Bluetooth to simply replace IrDA in new versions of products).

For the devices to communicate via IrDA they must have a direct line of sight.

More specifically, IrDA defines two suites of protocols aimed at supporting wireless communication using infrared light. The two standards are IrDA Data standard and IrDA Control standard. IrDA Data is recommended for high speed short range, line of sight, point-to-point cordless data transfer - suitable for digital cameras, handheld data collection devices, etc. If IrDA is supported, it must be targeted at the 4 Mb/s components. IrDA Control is recommended in-room cordless peripherals to host PC for lower speed, full cross range, point-to-point or to-multipoint cordless controller - suitable for keyboards (1 way), joysticks (2 way and low latency) etc.

IrDA was popular on laptops and some desktops during the late 90s through the early 2000s. However, it has been displaced by other wireless technologies such as WiFi and Bluetooth, favored because they don’t need a direct line of sight, and can therefore support hardware such as mice and keyboards. It is still used in some environments where interference makes radio-based wireless technologies unusable.
4.2.2 Performance analysis

In our case, we used IrDA again in order to implement the conditional probability-based approach, just like with RFID. The difference of this technology is that it minimizes the searching area of the original algorithm even more than RFIDs. This is because IrDA requires line of sight between the IrDA beacon transceiver and the mobile device in addition to RFID. In other words, when an IrDA beacon transceiver is placed in a hallway, only devices with direct line of sight can be identified.

The difference with RFID technology is that in IrDA the mobile device interacts with the beacon transceiver and there is no need for a server, like in RFID technology to communicate firstly with the reader and then with the mobile device, which saves time for the application in run-time. Moreover, the need of line of sight attribute of IrDA makes the system more robust and secure, opposed to RFID where due to scattering phenomena of the signal, it may identify devices in areas of the testbed where it should not.

It is expected that the accuracy of the algorithm will be improved using this feature even more than in the case of RFID technology used, as described above.

Each IrDA beacon transceiver was placed on the center of the ceiling of each hallway. Due to the line of sight characteristic of IrDA, each hallway is identified by at least one transceiver. The transceivers (Figure 4.11) used (six horizontal LEDs each bought from lesswire AG) have a range of 7 m or 12 m and the angle the six LEDs cover is 165°. This means that one transceiver can cover a hallway of 13 m or 23.5 m long, regarding the fact that the hallways in ICS are 2.5 m high.

![Figure 4.11: One board IrDA beacon with six LEDs, horizontal mounted to the beacon board.](image)

In the case of the 35 m long hallway, two transceivers (one with range 7 m and another
one with 12 m) are needed, whereas for the 11 m long hallway, only one transceiver (with range 7 m) is needed.

**WiFi using IrDA as conditional probability:** The IrDA technology is used in order to give extra information to the user regarding the subarea of the testbed that he/she is in. In other words, when the user is identified to be in a specific area covered by an IrDA beacon transceiver, it means that he/she is inside the specific transceiver’s range. The original algorithm takes this as an input in order to minimize its searching area in the transceiver’s range. So, instead of the system searching the whole testbed to find the users’ exact position, it only searches in the boundaries of the transceivers’ cover area. This means that in this method the IrDA technology only reports the subarea of the testbed the user is in.

The main algorithm using IrDA technology is,

1. Mobile device is in range of a beacon transceiver and identifies it

2. Retrieve Beacon ID (continuously broadcasted by devices)

3. Forward received data to the voting algorithm

4. The voting algorithm can reduce the searching area in testbed, to the range of the reader that identified the specific user

5. The voting algorithm returns the estimated position of the user

The performance of this method is illustrated in Figure 4.12, using the same use case scenario (Scenario 1) as in previous results. We can see that the decile- and quartile-based methods match, again in the WiFi&IrDA-like in the WiFi-enabled version, without improving the final results at all as Figure 4.12 shows. The use of IrDA technology has not changed the median location error of decile- and quartile-based methods from 1.1 m, whereas it has increased the median error of confidence interval-based method from 2 m to 2.4 m.

Thus, IrDA technology can improve high location errors, which is more obvious in the confidence interval-based algorithm where the maximum location error is decreased to 4.5 m with the IrDA use from originally 6 m, when only WiFi is used. However, this error
Figure 4.12: Comparing location error results of WiFi&IrDA- and WiFi-enabled – Scenario 1. (Set 6)

could have been further minimized if the range of IrDA transceivers was lower than this error, but in this case more devices would have been needed.

Again, we run Scenario 2 (as described in the previous section) for WiFi&IrDA-enabled. We validated this intuition by Figure 4.13, where we see that for the confidence interval-, quartile- and decile-based algorithms, the maximum error is 20.5 m, 16 m and 16 m, respectively, and the median location error is 3 m in all cases, when only WiFi is enabled. On the other hand there is a great improvement when IrDA technology is used, in which case the median location error is decreased to 2.4 m in confidence interval-based and to 1.5 m in quartile and decile-based method. What is more, there is a significant impact on the maximum location error of all methods. Namely, in the confidence interval-based method the maximum error decreased from 20.5 to 8.5 m, in the quartile-based from 16 to 7.5 m, and in the decile-based from 16 to 5 m.

As far as the time needed for a user to find out the ID of the IrDA beacon transceiver’s range he/she is in, the original algorithm needs about 2 sec in order to pass this information – argument to the voting algorithm. This is not a long time period if this technology was the only one used, but as mentioned before, the WiFi-enabled need about 3 sec to estimate a position, which means that the final time needed in the WiFi&IrDA-enabled is 5 sec,
which is not a very satisfying time interval for a run-time location-based application.

Scenario 2 shows that the IrDA technology can decrease high errors (higher than the IrDA transceivers’ range), whereas Scenario 1 proved that it cannot decrease errors lower than the transceivers’ range. This means that IrDA technology can be helpful in such a location-sensing system in order to eliminate the outliers and extreme results of the algorithm that can be due to instantaneous signal-strength variation, since signal-strength changes dynamically for various reasons.

Later on we wanted to examine the impact of the IrDA beacon transceiver’s range. Therefore we conducted the exact same experiments with the only difference that the range of each beacon transceiver is only 6.5 m for both Scenario 1 and 2 (WiFi&IrDA(6.5)), where 14 transceivers were needed. The range of each transceiver in the past experiments was 7 and 12 m depending on its position (WiFi&IrDA(7, 12)), where 6 transceivers were needed.

The results of this comparative study are illustrated in Figure 4.14 for Scenario 1, and in Figure 4.15 for Scenario 2. We can see that the median location error is minimized even more using transceivers that cover a smaller area. In both scenarios the median location error using the confidence interval-based method is 1.5 m, whereas it is only 1 m using the
Figure 4.14: Comparing location error results of WiFi&IrDA(7, 12)- and WiFi&IrDA(6.5)-enabled – Scenario 1. (Set 6)

quadile- or decile-based methods.

Figure 4.15: Comparing location error results of WiFi&IrDA(7, 12)- and WiFi&IrDA(6.5)-enabled – Scenario 2. (Set 10)
Real-time evaluation  On the 5-th of June 2009, we decided to deploy six IrDA beacon transceivers in the ICS testbed as shown in Figure 4.16, assigning a different ID to each device. At about 15:00 to 16:00 the same day, during training phase, we collected signal-strength measurements in the main hallways, in order to create the training map for the system (Set 11).

Figure 4.16: ICS testbed with IrDA transceivers’ coverage.

On the 10-th and 11-th of June we performed the run-time experiments in about 45 different positions throughout the whole testbed at 18:00 to 18:30 (Set 11). We tested the confidence interval- and decile-based algorithms, respectively. The results of the two methods incorporating the IrDA technology are placed in the following Figure 4.17. Once again the decile-based method has higher accuracy with median location error of 0.5 m, while the confidence interval-based one has median location error of 1 m. It is important to point out that 75% of the times the decile-based algorithm was tested, the error was less or equal to 1 m.

The conditions that the experiments were conducted, as well as the training map was created, were the same. Few people were inside the testbed, and no high noise level was noticed. Obviously, if different conditions were chosen for training and run-time phase, higher location error would have occurred.
Finally, we can see the great impact that IrDA technology has on the accuracy of CLS, comparing these results with those of Figure 3.15, that were conducted during similar conditions. In that case, the median location error of decile-based method using only WiFi was 1.1 m, whereas the incorporation of IrDA technology was decreased it to 0.5 m (Fig. 4.17). The difference of this real-time experiments compared to the ones described above, is in the placement of the IrDA devices. In this case only few important areas (i.e., doors, beginning of hallways) are covered by IrDA devices, and still they have a major impact on the accuracy.

4.3 Conclusions

Bluetooth is another technology that has already been integrated with the proposed algorithm [31], resulting in a median location error in the TNL testbed of 1.4 m compared to 1.6 m in the case that only WiFi is used.

Differences of the various technologies used are described in Table 4.1.

As the sections above have already shown and taking into consideration Scenario 1, RFID technology has no impact in the accuracy of the algorithm. Only if more RFID
<table>
<thead>
<tr>
<th>Technologies</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bluetooth</td>
<td>High amount of located devices</td>
<td>Small cell range</td>
</tr>
<tr>
<td></td>
<td>Big set of extra services</td>
<td>No hand-off procedure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Narrow bandwidth of data</td>
</tr>
<tr>
<td>IrDA</td>
<td>Good data transfer speed</td>
<td>Line of sight technology</td>
</tr>
<tr>
<td></td>
<td>Cheap equipment</td>
<td>Low accuracy</td>
</tr>
<tr>
<td>RFID</td>
<td>Passive tags technology</td>
<td>Sensitiveness for environmental parts</td>
</tr>
<tr>
<td></td>
<td>Good coverage area</td>
<td>Big positioning errors</td>
</tr>
<tr>
<td></td>
<td>Hidden tags location</td>
<td>Slow position calculation speed with big amount of clients</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Big power consumption with interrogate antennas</td>
</tr>
<tr>
<td>WiFi</td>
<td>High data transfer capacity</td>
<td>Influence of environment to accuracy</td>
</tr>
<tr>
<td></td>
<td>High precision with different techniques combination</td>
<td>No direction info without dynamics</td>
</tr>
<tr>
<td></td>
<td>Big cell coverage</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Soft hand-off procedure</td>
<td></td>
</tr>
<tr>
<td></td>
<td>No extra equipment for positioning</td>
<td></td>
</tr>
</tbody>
</table>

Readers with lower range were placed in the testbed, higher accuracy would be achieved. This means that the lower the range of the reader the higher the accuracy a positioning system may achieve. Because placing more RFID readers would be much more expensive, we decided to use another technology providing lower covering range with less cost, the IrDA technology.

Moreover, IrDA technology can improve high location errors, which is more obvious in the confidence interval-based algorithm where the maximum location error is decreased to 4.5 m with the IrDA use from originally 6 m, when only WiFi is used. Again in the decile- and quartile-based implementations there is no impact of the IrDA technology noticed when high covering area is incorporated, because the location error is already very low (median location error 1.1 m). However, in a different scenario were the IrDA beacon transceivers cover smaller areas, as proved in previous section in Figure 6.13, the median error using decile- or quartile-based method is only decreased by 0.1 m, to 1 m, but the
maximum location error is decreased from 3 to 1.8 m. Moreover, in a scenario where high location errors occur (outliers), they are eliminated with the use of IrDA technology, as Scenario 2 results have shown. In this case the maximum location error using the confidence interval-based method is originally 20.5 m, but large IrDA covering areas (WiFi&IrDA(7, 12)) reduce it to 8.5 m and small ones (WiFi&IrDA(6.5)) to only 5 m. Accordingly the median location error is also greatly affected and reduced from originally 3.5 m to 2.5 and 1.5 m, respectively.

The positioning system presented in this work also takes into consideration the cost of a location-sensing system and therefore only uses the minimum number of devices needed to cover the area. The location error of such a system could have been further minimized if the range of IrDA transceivers, or RFID readers was lower than the median location error of WiFi-enabled, but in this case more devices would have been needed.

Apart from the cost, another very important aspect that we have not mentioned is the time overhead that adding extra technologies poses. In our case not only the location error is not minimized by adding RFID and IrDA technology, but the algorithm needs more time to estimate the user’s position due to the time needed to collect information from these two technologies.

As mentioned before WiFi- or Bluetooth-enabled methods need about 3 sec to compute a user’s estimated position. Adding RFID technology means adding 4-5 sec per estimation, and IrDA technology would add another 2 sec. In other words WiFi&RFID- and WiFi&IrDA-enabled ones need 7-8 and 5 sec, respectively.

It is obvious that 5-8 sec are too much for a real-time location based system, that this system is going to be used, which needs a user’s position in order to provide information to him/her or take any other action depending on his/her position. It is most likely that the user will have changed position in less than 5 sec and the position that the system will report will then be an old one at that point. In other words, despite the fact that the results may be slightly better in certain cases, the time needed makes WiFi&RFID- and WiFi&IrDA-enabled methods not an ideal solution for a real-time application, even if the results were even more improved by adding those two extra technologies.
Chapter 5

Conclusions

As described in Chapter 2.3, there are a few location-sensing systems similar to our own. All the methods described before can be easily incorporated with CLS, and therefore we will compare the ones described before to our methods though CLS. A comparison with the most important of them follows.

RADAR [3] is a system that also uses infrastructure of IEEE 802.11 APs. IEEE 802.11 wireless cards collect signal-strength measurements from APs and compute the user’s location by performing a statistical analysis, like CLS. The authors report a median location error of about 3 m. Opposed to this result CLS has a median location error of about 2 m in most testbeds, and even 0.5 m in the ICS testbed when combined with IrDA technology and 1.1 m when not. This is because in CLS multiple statistical values per AP are used, whereas Bahl et al., only use one signal-strength value (usually the maximum value) per AP. They compute the euclidian distance between the run-time and training phase fingerprints. We also compute the euclidian distance in the decile-based method, with the difference that more than one values are compared. This shows that the more detailed the fingerprint the higher the accuracy achieved.

Unicycles and Baddy Nata [32] introduced a co-operative location-sensing system that propagates position information of landmarks towards distance hosts, while closer hosts enrich this information by determining their own location. Various methods were evaluated (i.e., “DV-hop”, “DV-distance”, “Euclidian”). Afterwards, the authors [33] incorporated specialized hardware to their algorithm, in order to estimate the angle between two hosts
in an ad hoc network. They used antenna arrays or ultrasound receivers in order to implement this idea. Hosts gather data, compute their estimated positions, and expand them throughout the network. “DV-distance” is the method closer to CLS, because signal-strength measurements are used in order to estimate the distance between peers. The main difference between CLS and this work lies in the grid-based representation of the testbed and voting algorithm.

Horus WLAN [48], proposed by Youssef et. al, operates in training phase and the run-time phase. The two systems differ mainly in the number of samples required in each phase in order to create the training map. As Horus WLAN computes the correlation between consecutive signal-strength samples using an autoregressive model, about 100 signal-strength samples are required for creating a valid model. Thus, extensive results on the number of samples and the impact of the sample size to the location estimation, were not provided. Moreover, high number of samples increases the time needed for the system to both be trained and compute the position of the user. CLS requires only 60 samples per cell in training phase, and 30 samples in run-time. Their system was evaluated in a 11.8 x 35.9 $m^2$ area with coverage of totally 5 APs and an average of 4 APs per cell. This showed that Horus WLAN has 90% of 1.32 m location error. CLS has a 90% of 1.8 m location error (using only IEEE 802.11 infrastructure) and 4 seconds to reach a conclusion on the user’s location, in a similar testbed (ICS), with a total of 21 APs and an average of 5 APs per cell.

The main differences between ARIADNE [20] and our system is that they use no signal strength map creation, but it is self generated and they only collect one signal strength value both at training and run-time phase, whereas we collect 60 and 30, respectively. Finally ARIADNE uses a clustering-based algorithm in addition to our probabilistic-based one. Their average location error is 2.84 m when our system has an accuracy of even 1.1 m in even much bigger testbed (ICS).

Ekahau Positioning Engine (EPE) [9], like CLS, uses the IEEE 802.11 wireless infrastructure, creating a training map, and performs a comparison between the training and run-time measurements to estimate the position of the user. As shown before the accuracy of CLS is 10% or more, higher than Ekahau.
COMPASS [23] like CLS uses IEEE 802.11 technology as well as a probabilistic-based algorithm. The main differences are that in COMPASS the orientation of the user and the noise are taken into consideration, unlike CLS. Moreover, this system returns 5 points in space as the user’s position, and during the signal strength collection it collects 880 for training time and 110 samples for run-time, instead of one position that CLS reports and 60 and 30 samples collected, respectively. This makes CLS a much faster system as well as a more accurate one, while in the case where only one orientation is taken into consideration (like CLS) the median location error is 2.5 m, while CLS in TNL is 1.6 m, in much bigger testbeds like the aquarium (1760$m^2$) is 2m and in ICS (1462$m^2$) 1.1 m. On the other hand when all 8 orientations are taken into consideration the median location error drops to 1.9 m. Though, the tradeoff between time and accuracy makes this system not very useful in real-life scenarios.

As far as the extended CLS with RFID technology is concerned, compared to [46] both systems use no training phase (CLS also has an implementation where training phase exists) and the tags are the devices trying to be located. On the other hand our system only uses simple probabilistic methods and 2-D representation of the physical space, whereas in [46] complicated optimization and geometry (Nelder-Mead nonlinear optimization and geometry-based method that minimizes the error objective functions) and 3-D representation is used. Unfortunately they have no real-life measurements or results so as to compare the accuracy of both systems.

Furthermore, Anonymous Tracking using RFID tags [24] depicts some very serious subjects like privacy on which researcher have paid great attention over the last few years, in addition to CLS. In our system a central database has some knowledge of the user’s position but not the exact one. This is because the central database contains all RFID tag’s IDs along with corresponding IP address of the user carrying the tag. This means that a central server will have knowledge of the larder area in which the user is, through the RFID reader ID that has identified the specific tag, but not the exact position that is computed in the user’s mobile device. Another difference of this system and our own is that this implementation identifies a whole set of tags, whereas our own system that only identifies one user at a time. Finally, they use both probability and set-theory-based
methods while we only use probability-based methods and conditional probabilities.

In this thesis, we have implemented an algorithm in C and C++ in order for a positioning system, that will use it, to be able and collect signal-strength values, either at run-time or not, and then compute an estimated position for the user, through a statistical analysis using various criteria. We computed the range error (i.e., distance estimation error) based on real-life signal-strength measurements that had taken place in the Institute of Computer Science (ICS) of the Foundation for Research and Technology - Hellas (FORTH), in the Telecommunication and Networks Laboratory (TNL) of ICS-FORTH, as well as in the Cretaquarium [8] in Hersonissos - Heraklion.

We experimented with the impact of several parameters, such as the statistical information used, the range error, the physiology of the testbed and the number and placement of APs on the accuracy of the position estimation, through several scenarios. Apart from the IEEE 802.11 wireless infrastructure, we later on added extra technologies. In particular, to enhance its accuracy, we extended the fingerprinting method by incorporating additional information obtained by non IEEE 802.11 sources, such as RFID and IrDA.

To summarize, the main contributions of this thesis are the extensive empirical studies of the proposed fingerprinting methods in real environments. For example, we found that:

1. The larger and more detailed the fingerprint the higher the accuracy.

2. Median location error using only IEEE 802.11 in testbed of ICS is 1.1 m.

3. The incorporation of extra technologies (i.e., IrDA) along with IEEE 802.11 can improve the accuracy of the proposed algorithms. The use of IrDA resulted in a median location error in the ICS testbed of 0.5 m.
Appendices
Matlab implementation of confidence interval-based CLS

% Preconditions:
% - input : vector [x, y, ss, ..., ss]
% - training : vector [x, y, ss, ..., ss]
% - 75% confidence interval used
% Run-time input data
load(input);
% training data
load(training);

% Initialize location error results of all input scenarios
for(i=1:size(input,1))
    distance(i) = 0.0;
end

% For each run-time signature estimate a position
for(in=1:size(input,1))
% Initialize variables used
    max = 0.0;
    cnt = 1;
    count = 1;
    x_m = 0.0;
\begin{verbatim}
    y_m = 0.0;
    % Number of cells in x-axes of testbed
    MAX_X = 77;
    % Number of cells in y-axes of testbed
    MAX_Y = 62;
    % Retrieve real run-time position of user
    % + 1 stands for matrix starting from 1 instead of 0
    in_x = input(in,1) + 1;
    in_y = input(in,2) + 1;
    % Initialize "probability" of each cell that the user there
    for(i=1:MAX_X)
        for(j=1:MAX_Y)
            weights(i,j) = 0.0;
        end
    end
    % Initialize matrix that stores all positions with maximum probability
    for(i=1:size(training,1))
        result(i,1) = -1;
        result(i,2) = -1;
    end
    % For all positions in training set
    for(i=1:size(training,1))
        % Retrieve position of the current training entry
        cell_x = training(i,1)+1;
        cell_y = training(i,2)+1;
        % Initialize "probability" with 0
        weights(cell_x,cell_y) = 0.0;
\end{verbatim}
j=3;
% For each AP
while(j<=size(training,2))
    % Retrieve confidence interval of training cell [upper_tr, lower_tr]
    upper_tr = training(i,j);
    lower_tr = training(i,j+1);
    % Retrieve confidence interval of run-time cell [upper_r, lower_r]
    upper_r = input(in,j);
    lower_r = input(in,j+1);

    % Compute the percentage of coverage between the 2 confidence
    % intervals above - vote
    matchInt = 0;
    % Training interval containing run-time
    if ( lower_tr <= lower_r && upper_tr >= upper_r )
        matchInt = 1;
    elseif ( lower_r <= lower_tr && upper_tr >= upper_r && upper_r>=lower_tr )
        val1 = upper_r - lower_tr;
        val2 = upper_tr - lower_r;
        matchInt = val1/val2;
    elseif (lower_tr <= lower_r && upper_r >= upper_tr && upper_tr >= lower_r )
        val1 = upper_tr - lower_r;
        val2 = upper_r - lower_tr;
        matchInt = val1/val2;
    elseif (lower_r <=lower_tr && upper_r >= upper_tr )
        matchInt = 1;
    % Run-time interval containing training
    elseif (lower_r <=lower_tr && upper_r >= upper_tr )
        matchInt = 1;
% Training interval lower than run-time -- non-overlapping
elseif ( lower_r > upper_tr && upper_tr < upper_r )
    matchInt = 0;
% Run-time interval lower than training -- non-overlapping
elseif ( lower_r < lower_tr && lower_tr > upper_r )
    matchInt = 0;
end

% Add vote to the cell
weights(cell_x,cell_y) = weights(cell_x,cell_y) + matchInt;
% Move to next confidence interval
    j = j + 2;
end

% Search for maximum likelihood - vote of all training cells
for(i=1:MAX_X)
    for(j=1:MAX_Y)
        if(weights(i,j)>max)
            max = weights(i,j);
        end
    end
end

% Search for all cells with equal maximum vote
for(i=1:MAX_X)
    for(j=1:MAX_Y)
        if (weights(i,j) == max)
            result(cnt,1) = i;
            result(cnt,2) = j;
            cnt = cnt + 1;
        end
    end
end
end

end

end

% Compute a centroid of all cells with equal maximum vote

count=0;
for(i=1:cnt)
    if (result(i,1)==-1 && result(i,2)==-1)
        break;
    end
    x_m = x_m + result(i,1);
    y_m = y_m + result(i,2);
    count = count + 1;
end

% Estimated position

x_m = x_m/count;
y_m = y_m/count;

% Compute location error (real vs estimated position)
% /2 is because the size of each cell is 0.5 m x 0.5 m,
% but we need the location error in m.

ed(in) = sqrt(((in_x-x_m)*(in_x-x_m))+((in_y-y_m)*(in_y-y_m)))/2 ;
end
Matlab implementation of percentile-based CLS

% Preconditions:
% - input : vector [x, y, ss, ..., ss]
% - training : vector [x, y, ss, ..., ss]
% - percentiles computed using the prctile function of Matlab where
%PRCTILE Percentiles of a sample.
%Y = PRCTILE(X,P) returns percentiles of the values in X.
%P is a scalar or a vector of percent values. X is a vector.
%Percentiles are specified using percentages, from 0 to 100.
%For an N element vector X, PRCTILE computes percentiles as:
% 1) The sorted values in X are taken as the 100*(0.5/N),
% 100*(1.5/N), ..., 100*((N-0.5)/N) percentiles.
% 2) Linear interpolation is used to compute percentiles
% for percent values between 100*(0.5/N) and 100*((N-0.5)/N)
% 3) The minimum or maximum values in X are assigned to
% percentiles for percent values outside that range.

% Run-time input data
load(input);
% training data
load(training);
% For each run-time signature estimate a position
for in=1:size(input,1)

% Initialize variables used
    ss = 0.0;
    ap_total = 0.0;
    total_sq = 0.0;
    x_m = 0.0;
    y_m = 0.0;
    cnt = 1;
% Number of percentiles used
    period = 10 ;

% Initialize "probability"(-euclidian distance) of each cell
% that the user there
    for(i=1:MAX_X)
        for(j=1:MAX_Y)
            eucl_distance(i,j) = -1.0;
        end
    end

% Initialize matrix that stores all positions with maximum probability
    for(i=1:size(training,1))
        result(i,1) = -1;
        result(i,2) = -1;
    end

% Number of cells in x-axes of testbed
    MAX_X = 77;
% Number of cells in y-axes of testbed
MAX_Y = 62;

% Retrieve real run-time position of user
% + 1 stands for matrix starting from 1 instead of 0

in_x = input(in,1) + 1;
in_y = input(in,2) + 1;

% For all positions in training set
for(i=1:size(training,1))

% Retrieve position of the current training entry
    cell_x = training(i,1)+1;
cell_y = training(i,2)+1;

j=3;
% For each AP
    while(j<=size(training,2))
% For each percentile
        for(k=0:period-1)
% Compute euclidian distance between corresponding percentiles
            ss=(input(in,j+k)-training(i,j+k))*(input(in,j+k)-training(i,j+k));
% Sum those distances for the specific AP
            ap_total = ap_total + ss;
        end
% Sum those distances for the specific cell
% The smaller the distance the more similar the vectors
    total_sq = total_sq + sqrt(ap_total);
ap_total = 0;
j = j + period;
end

% Assign vote of specific cell
eucl_distance(cell_x,cell_y) = total_sq;
total_sq = 0;
end

% Randomly initialize the minimum distance
min = eucl_distance(1,11);
% Search for minimum distance == maximum likelihood of all training cells
for(i=1:MAX_X)
    for(j=1:MAX_Y)
        if((eucl_distance(i,j)=-1) && (eucl_distance(i,j)<min))
            min = eucl_distance(i,j);
        end
    end
end

% Search for all cells with equal minimum distance
for(i=1:MAX_X)
    for(j=1:MAX_Y)
        if (eucl_distance(i,j) == min)
            result(cnt,1) = i;
            result(cnt,2) = j;
            cnt = cnt + 1;
        end
    end
end
% Compute a centroid of all cells with equal minimum distance
    count=0;
    for(i=1:cnt)
        if (result(i,1)=-1 && result(i,2)==-1)
            break;
        end
        x_m = x_m + result(i,1);
        y_m = y_m + result(i,2);
        count = count + 1;
    end

    % Estimated position
    x_m = x_m/count;
    y_m = y_m/count;

    % Compute location error (real vs estimated position)
    % /2 is because the size of each cell is 0.5 m x 0.5 m,
    % but we need the location error in m.
    ed(in) = sqrt(((in_x-x_m)*(in_x-x_m))+((in_y-y_m)*(in_y-y_m)))/2;
end
Bibliography


