Empirical-based Measurements and Analysis of Positioning Systems for Mobile Computing Applications

Artemis Papakonstantinou

Thesis submitted in partial fulfillment of the requirements for the

Master of Science degree in Computer Science

University of Crete
School of Sciences and Engineering
Computer Science Department
Knossou Av., P.O. Box 2208, Heraklion, GR-71409, Greece

Thesis Supervisor: Prof. *M. Papadopouli*

Work performed at the:

Foundation for Research and Technology - Hellas (FORTH)
Institute of Computer Science (ICS)
Telecommunications and Networks Laboratory (TNL)
100 N. Plastira Av. Vassilika Vouton, Heraklion, GR-70013, Greece

Heraklion, June 2011

Abstract

During the last years we witnessed an increasing interest in pervasive computing systems and applications. Ubiquitous computing environments provide multitudes of technologies seamlessly augmented with physical systems to aid users in everyday tasks. Accurate location awareness is of paramount importance in most pervasive applications. Numerous techniques for indoor localization based on IEEE 802.11, bluetooth, ultrasonic and vision technologies have been proposed. This thesis presents two different approaches for positioning. In the first the IEEE 802.11 infrastructure is used and in the second computer vision is employed as the sensing modality.

We propose a localization technique based on the generation of statistical fingerprints from signal strength measurements, that are collected from several access points (APs). A discretized grid-like form of the environment is considered and a signature at each cell of the grid is computed. At run time the system compares the signature at the unknown position with the signature of each cell of the grid. The experimental evaluation of the proposed positioning system has been conducted on the premises of FORTH and the Cretaquarium under different, real-life conditions. Our results indicate that the user's position can be detected with a few meters accuracy, depending on the environment, the conditions in it, and the fingerprinting-technique used. The impact of each one of these parameters has been studied.

The pervasive nature of mobile phones makes them an ideal platform for location-aware applications. In this context, we have used the built-in cameras of mobile phones and computer vision techniques for localization. We have developed an application for the Android platform, that is able to identify the position of a user relative to a QR code (two dimensional barcode) with cm accuracy. The user scans the QR code with the camera of the mobile device and the captured image is analyzed, in order to detect the camera's position. The QR codes are attached to physical objects in order to retrieve object-related information and functionality. In this way, the user can get location-based information or trigger actions (e.g. printing of a document) by decoding a QR code.

Περίληψη

Τα τελευταία χρόνια παρατηρήσαμε ένα αυξανόμενο ενδιαφέρον για τα συστήματα και εφαρμογές πανταχού παρόντος υπολογιστή. Τα πανταχού παρόντα υπολογιστικά περιβάλλοντα παρέχουν πλήθη τεχνολογιών απόλυτα συνυφασμένες με φυσικά συστήματα για τη βοήθεια των χρηστών στις καθημερινές τους εργασίες. Ο ακριβής εντοπισμός θέσης είναι υψίστης σημασίας στις περισσότερες πανταχού παρούσες υπολογιστικές εφαρμογές. Πολυάριθμες τεχνικές έχουν προταθεί για την εύρεση θέσης σε εσωτερικούς χώρους, οι οποίες βασίζονται σε τεχνολογίες όπως το IEEE 802.11, bluetooth, υπερηχητικές και υπολογιστική όραση. Η παρούσα μεταπτυχιακή εργασία παρουσιάζει δύο διαφορετικές προσεγγίσεις για τον εντοπισμό θέσης. Στην πρώτη χρησιμοποιείται η IEEE 802.11 υποδομή και στην δεύτερη χρησιμοποιείται η υπολογιστική όραση σαν αισθητήρια μέθοδος.

Προτείνουμε μια τεχνική εντοπισμού θέσης, βασισμένη στην δημιουργία στατιστικών 'αποτυπωμάτων' από τις μετρήσεις έντασης σήματος, οι οποίες συλλέγονται από διάφορα σημεία πρόσβασης (APs). Ο φυσικός χώρος χωρίζεται μέσω ενός πλέγματος σε ισομεγέθη κελιά και υπολογίζεται ένα 'αποτύπωμα' έντασης σήματος σε κάθε κελί. Κατά την εκτέλεση του, το σύστημα συγκρίνει σε πραγματικό χρόνο το αποτύπωμα στην άγνωστη θέση με το αποτύπωμα κάθε κελιού στον χώρο. Η πειραματική αξιολόγηση του προτεινόμενου συστήματος εύρεσης θέσης έχει πραγματοποιηθεί στις εγκαταστάσεις του ITE και στο ενυδρείο Κρήτης κάτω από ρεαλιστικές συνθήκες. Τα αποτελέσματά μας δείχνουν ότι η θέση του χρήστη μπορεί να ανιχνευθεί με ακρίβεια λίγων μέτρων, ανάλογα με το χώρο, τις συνθήκες που επικρατούν σε αυτό, και τη μέθοδο που χρησιμοποιείται για τη δημιουργία των 'αποτυπωμάτων'. Η επίδραση των παραμέτρων αυτών στην ακρίβεια του συσήματος έχει επίσης μελετηθεί.

Η διαδεδομένη χρήση κινητών τηλεφώνων στην καθημερινή μας ζωή, τα μετατρέπει σε μια ιδανική πλατφόρμα για τη δημιουργία εφαρμογών που που προσφέρουν πληροφορία στο χρήστη με βάση τη θέση του. Σε αυτό το πλαίσιο, χρησιμοποιούμε τις ενσωματωμένες φωτογραφικές κάμερες των κινητών τηλεφώνων και τεχνικές υπολογιστικής όρασης για τον εντοπισμό θέσης. Δημιουργήσαμε μια εφαρμογή για την πλατφόρμα Android, η οποία μπορεί να προσδιορίσει τη θέση ενός χρήστη σε σχέση με ένα QR code (δισδιάστατο barcode) με ακρίβεια εκατοστών. Ο χρήστης ανιχνεύει το QR code με την κάμερα του κινητού και η φωτογραφία που τραβάει αναλύεται, προκειμένου να εντοπιστεί η θέση της κάμερας. Τα QR codes μπορούν να τοποθετηθούν σε αντικείμενα έτσι ώστε να είναι δυνατή η ανάκτηση πληροφοριών και λειτουργιών, σχετικές με το αντικείμενο. Με αυτόν τον τρόπο, ο χρήστης μπορεί να πάρει πληροφορίες σχετικές με τη θέση του στο χώρο ή να προκαλέσει διάφορες ενέργειες (π.χ. εκτύπωση ενός εγγράφου) με την αποκωδικοποίηση ενός QR code.

Acknowledgments

The work reported in this thesis has been conducted at the Telecommunications and Networks Laboratory (TNL) of the Institute of Computer Science (ICS) of the Foundation for Research and Technology - Hellas (FORTH), and has been financially supported by a FORTH-ICS scholarship. It has also been partially funded by the Greek General Secretariat for Research and Technology (Regional of Crete) Crete-Wise grant.

First of all I would like to thank my advisor Prof. Maria Papadopouli, for her guidance and her support throughout this work. Her constructive remarks and the time she devoted to me constitute a significant amount of help. Furthermore, I would like to thank all my fellow students and/or co-workers at FORTH for their help and their support in all the good and bad times. Especially, I would like to thank Antonis Makrogianakis, Tasos Alexandridis and Pavlos Charonyktakis for their close collaboration for the development of a mobile application that is presented in this work. Finally, I would like to thank my family (Apostolis, Roula and Spyros) for their love and support they have offered me all these years. They have sacrificed everything in order to help me reach my goals. Without their help I would certainly have not made it to here. Last but not least, I would like to thank my close friends for encouraging me and supporting me all these years: Zoe Sebepou, Dimitris Tsaliagos, Dimitris Rodopoulos, Thanos Makatos, Evi Dagalaki, Alexandros Kapravelos, Vicky Papavisileiou and Athanasios Karras.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Pervasive computing relies on the convergence of wireless technologies, advanced electronics and Internet. Ubiquitous availability of wireless networks combined with powerful mobile devices (e.g. mobile phones, smart phones and PDA's) has the potential to lead to a huge set of new services and applications. Emerging ubiquitous computing applications have a great need for location awareness. Location-based services can be beneficial in many aspects of our everyday life, such as transportation, entertainment, emergency situations for disaster relief, and assistive technology in the medical community. Smart-environments, that can unobtrusively communicate with the user and provide context-aware information can also benefit from the knowledge of physical location. The determination of the physical location is known as localization or location sensing.

Several research [44, 8, 35, 33, 2] and commercial [3, 1] location systems have been built, which vary significantly in their capabilities and requirements. These systems can be classified according to their dependency on and use of specialized infrastructure and hardware, signal modalities, training, methodology and/or use of models for estimating distances, orientation and position, coordination system used, scale, cost, localized or remote computation, accuracy and precision requirements. The distance can be estimated using time of arrival (e.g. GPS[1], PinPoint[35]) or signal-strength measurements [8, 20, 34], if the velocity of the signal and a signal attenuation model for the given environment, respectively, are known.

The de facto standard for outdoor location sensing is GPS [1]. However, GPS does not work everywhere. In particular, the technology typically breaks down near obstacles, such as trees and buildings, and in indoor environments. In addition, its resolution of a few meters is not adequate for many applications and the specialized components needed for GPS impose cost, and energy consumption requirements that

are problematic for mobile platforms. In order to alleviate the dependency on GPS, different sensing technologies have been used, including IEEE 802.11 (Radar [8, 23], Ekahau [3], Ariadne [18], Horus [34], [19, 20, 37, 38]), Ultra Wide Band (Ubisence [5]), infrared (Active Badge [33, 32], ParcTab [40, 41]), ultrasound (Active Bat [2], Cricket [25, 26]), Bluetooth [6, 9, 13, 27, 15], 4G [28], RFIDs (Landmarc [43], SpotOn [44], [30]), and physical contact with pressure (Smartfloor [4]), touch sensors or capacitive detectors. Computer Vision has also been used for location-sensing. Vision-Based systems (CyberCode [47], TRIP [48], [46, 55]) employ cameras, bar codes, QR codes and fiducial markers in order to locate or navigate the user. Some systems may also combine multiple modalities (vision, sound etc.) in order to improve localization accuracy (SourroundSense [7], PlaceLab [45]).

IEEE 802.11 infrastructures have been deployed widely in universities, hospitals, airports and other public areas. Their popularity, combined with their low deployment cost, and the advantages of using them for both communication and positioning, make them an attractive choice for use in location sensing. Approaches using IEEE 802.11 exploit the radio frequency (RF) beacons emitted from APs to track a mobile user. Signal-strength values gathered from different APs are used either for the creation of a signature/map of the physical space [8, 20, 34] or are combined with radio-propagation models to predict the distance of a wireless client from an AP (or any landmark) or even between two wireless clients (peers) with estimated position (e.g. CLS [31]). Signal-strength maps can be formed with data from different sources in order to improve location-sensing [15, 25]. In some situations the deployment of a wireless infrastructure may not be feasible. In such cases positioning mechanisms may exploit cooperation by enabling devices to share positioning estimates [10, 11, 12, 14, 21, 29, 31, 35].

1.1 Motivation

Our main goal is to design a location-sensing system for mobile computing applications, that will take advantage of existing IEEE 802.11 infrastructure and hardware to provide position estimates, within a few meters accuracy. The implemented algorithms should be able to achieve high accuracy regardless of the the environment's physiology and be able to overcome the problems that the IEEE 802.11 technology poses.

We will also employ the IEEE 802.11 infrasrtucture and mobile phones for positioning. We will consider a location-aware application that is able to use only the integrated camera of mobile phones and printed QR codes in order to estimate the user's position, using computer vision as the sensing modality. Moreover, it can be

used as a standalone positioning application, or as a support application for the IEEE 802.11-based location sensing system, in order to enhance its accuracy. In order to avoid the problem of portability among platforms, we will target an increasing in popularity platform, Google's Android.

1.2 Challenges

Hightower and Borriello stated that "No single location sensing technology is likely to become dominant. There are simple too many dimensions along which location sensing mechanisms can vary." [60]. This indicates that the choice of the location sensing technology is going to depend on the usage context. In many situations, due to environmental constrains, cost, maintenance, and regulatory barriers, the deployment of a specialized infrastructure for location sensing is not feasible. The deployment and maintenance cost of a positioning system, may make its usage in everyday life impossible. On the other hand, a system that exploits existing infrastructure in an indoor environment, like IEEE 802.11, can be easily deployable, scalable, cost effective and computationally inexpensive. Our main goal is to design a location-sensing system, that does not depend on a specialized hardware. However, this imposes a great challenge due to the interference in RF signals and the dynamic characteristics of the radio propagation in various environments. The dynamic characteristics of the wireless channel may have a negative impact on a location-sensing system's accuracy. Thus, a positioning system should be able to handle the problems that arise with different environmental conditions.

The pervasive nature of mobile devices such as mobile phones and smart-phones has also been exploited in order to develop a location-aware application. The resource constraints of mobile devices, namely energy, communication bandwidth and storage always impose great challenges. Portability among mobile platforms is also something that mobile application developers should take into account. Computer Vision, that has been employed in order to recognize the QR codes, is a very challenging area. Problems that need to be overcome, are the dependence of the code's successful recognition and the camera's pose correct estimation on lighting conditions, image resolution and quality, and the correct tuning of the parameters used in the image analysis procedure.

1.3 Thesis Statement

This thesis is based on an existing location sensing system, named Cooperative Location Sensing (CLS) [31, 14]. The existing IEEE 802.11 infrastructure in the premises of FORTH (Foundation for Research and Technology - Hellas) and in Cretaquarium in Heraklion has been used for location-sensing. The proposed algorithms generate statistical-based fingerprints using the received signal-strength (RSSI) measurements from the IEEE 802.11 infrastructure. The user's position can be detected with accuracy within a few meters, depending on the environment, its conditions, and the fingerprinting-technique used. The impact of each one of these parameters has been evaluated, using the collected RSSI measurements, under different environmental conditions, from the two testbeds mentioned. Moreover, the impact of the Access Points (APs) on the system's accuracy has been examined, using a popular data analysis method named Principal Component Analysis.

Also, we have employed computer vision techniques in order to determine the user's position. The user's location, with respect to certain landmarks, can be determined with accuracy in the order of cm. For the deployment of the system only printed QR codes are needed, which can be scanned with camera enabled mobile phones. Location-based information can be provided to the user, by encoding data at several QR codes which are placed at strategic locations in the premises of FORTH (e.g. next to or under a point of interest). In this way, someone could for example get guidance when entering a building, get information about points of interest or perform actions automatically (e.g. printing), by scanning a QR code.

To summarize, the main contributions of this thesis are the following:

- The implementation of a positioning system using statistical-based fingerprints generated from existing IEEE 802.11 infrastructure.
- A comparative performance analysis of various signal-strength fingerprinting methods in the premises of a research laboratory and an aquarium under different conditions.
- The development and evaluation of a computer vision-based positioning system for the Android platform.

1.4 Publications

Part of this thesis (chapters 3,4) has been published in the *13th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*

[61], where it has ranked among the top three papers. Part of the present thesis has also been submited to the *Special Issue of Ad Hoc Networks (Elsevier) Journal* [62].

1.5 Thesis Outline

The present thesis is organized as follows: Chapter 2 presents existing location sensing systems and recent work in pervasive computing, regarding the use of mobile phones in context and location aware applications. Chapter 3 presents the proposed fingerprinting techniques, the approaches used to enhance the system's accuracy, and the mathematical background of Principal Component Analysis, that has been used in the context of this work. Chapter 4 presents a comparative performance evaluation of the fingerprinting techniques in the premises of the Telecommunications and Networks Lab (TNL) at FORTH, as well as in Cretaquarium. Chapter 5 presents the implementation and evaluation of the vision-based positioning system, along with some background regarding the estimation of the camera's position with computer vision techniques. Finally, chapter 6 summarizes our main results and provides directions for future work.

Chapter 2

Related Work

In this chapter, we present an overview of research in localization and pervasive computing systems. We subdivide the presentation of the related work into four areas: 1. localization using the IEEE 802.11 infrastructure, 2. alternative localization approaches, including positioning techniques for ad-hoc networks and location sensing with other modalities than IEEE 802.11 or combination of different modalities, 3. localization with computer vision, and 4. pervasive computing systems, where systems that use mobile devices for the discovery of digital services in the environment and/or the interaction with real world objects are presented.

2.1 Localization using the IEEE 802.11 infrastructure

Over the last few years, significant research has been done in the area of location-sensing using RF signals. Radar [8] employs signal-strength maps that integrate signal-strength measurements acquired during the training phase from APs at different positions with the physical coordinates of each position. Each measured signal-strength vector is compared against the reference map and the coordinates of the best match will be reported as the estimated position. The median resolution of the RADAR system is in the range of 2 to 3 meters, about the size of a typical office room. Bahl et al. [23] extended Radar to provide continuous user tracking and to incorporate the dynamic changes of signal-strength nature, such as aliasing and multipath. This extended version of Radar resulted in a mean location error of 2.37 m The Horus WLAN system [34] substantially improved the positioning accuracy, as compared to Radar, by employing an autoregressive model that captures the auto-

correlation in signal strength measurements of the same AP at a particular location. In Horus, radio map locations are grouped according to APs covering them, and for each location a histogram of signal strengths is constructed during the offline phase. Horus can achieve an accuracy of 1.4 meters for 90% of the time, while its median error is less than half meter. Ladd et al. [20] proposed another location-sensing system for localizing nodes, tracking and determining orientation inside a building from measured RF signal strength. In its first step, a host computes the conditional probability of its location for a number of different locations by employing a probabilistic model, based on the received signal-strength measurements from 9 APs. In its second step, the system exploits the limited maximum speed of mobile users in order to refine the results and reject positions with an unreasonable change in the position of the mobile host. This system can predict host's location within 1.5 m with 83% accuracy, which is an improvement compared to the Radar system. Ariadne [18] is another novel and automated location determination method, that uses a two dimensional construction floor plan and only a single actual signal strength measurement. It generates an estimated signal strength map, using a radio propagation model. Given the signal measurements for a mobile, a proposed clustering algorithm searches the signal-strength map to determine the current location of the mobile device.

In [37] a localization technique based on bayesian inference and 802.11 WLAN is presented. In this system the user is located in region granularity (whole office, part of a hallway etc). The impact of the number of base stations and of the training set size have been explored. Experiments indicate that the Gaussian distribution fitting method, that has been used, can determine the correct cell in over 90% of the trials, by using only 17 out of a total of 33 APs. Moreover, authors indicate that a 60-second training set, containing around 37 scans is adequate for correct localization in most of their building. Finally, by performing a calibration for time varying phenomena (e.g. attenuation due to people in the building) and different hardware (different chipsets in network cards), the authors managed to achieve 88% correct location estimation, with 90% of the estimates within 3 m. Kung et al. [19] propose a method for evaluating the impact of the IEEE 802.11 APs on positioning in order to strengthen the role/contribution of a "good" AP while "de-emphasizing" the role of the "bad" APs. The "goodness" of an AP indicates the capability of that AP to estimate accurately its distance from the others. In [24, 36] the authors propose fingerprints, based on attributes that characterize the effects of multipath (e.g., channel response), in order to detect changes of the positions of wireless hosts. Yu-Chung-Cheng et al. [38] have evaluated the feasibility of building a wide-area 802.11 Wi-Fi-based positioning system. By using existing hardware and infrastructure and with relatively low calibration overhead, they can estimate a user's position with

a median positioning error of 13–40 meters. More specifically, in dense urban areas positioning accuracy is between 13–20 meters. The impact of limited calibration on the accuracy of the positioning algorithms is also studied, which is an important tradeoff while deploying such a wide-area location system. One interesting observation is that in areas with dense Wi-Fi coverage, the specific algorithm used for positioning is not as important as other factors including composition of the neighborhood, age and density of training data sets, and noise in them, while in sparser neighborhoods, sophisticated algorithms that can model the environment more richly have better performance.

2.2 Alternative localization approaches

In situations where a deployment of a wireless infrastructure may not be feasible, positioning mechanisms may exploit cooperation by enabling devices to share positioning estimates. Niculescu and Nath [21] introduced an algorithm in the field of location-sensing that works on simple geometric principles of Euclidean geometry concerning triangles and quadrilaterals. The information of the landmark locations is slowly propagated towards the nodes that are further away, while at the same time closer nodes enrich this information by determining their own location. Bulusu et al. [39] present a connectivity-metric method for localization in outdoor environments, that makes use of the radio-frequency (RF) communications capabilities of the devices. Nodes infer proximity to a subset of fixed reference nodes, which have overlapping coverage and periodically transmit beacons, and localize them to the centroid of their approximate reference points. The connectivity metrics are estimated according to the percentage of received beacons. Another location-sensing system in ad-hoc networks [10] performs positioning without the use of landmarks or GPS and presents the tradeoffs among internal parameters of the system. Saverese et al. [29] proposed a distributed algorithm that determines the position of nodes in an ad hoc network in two phases, namely the 'startup' and 'refinement' phase. The 'startup' phase is only responsible for giving nodes rough estimates of their positions in order to generate a first approximation of the topology of the network. The 'refinement' phase is an iterative process, during which each host broadcasts its position estimate, receives the positions and the corresponding range estimates from its neighbors, and computes a least square triangulation solution to determine its new position. The authors report that their algorithm is able to achieve position errors of less that 33% of the radio range, with 5% range error, 5% anchor nodes and an average connectivity of 7. PinPoint [35], presented by Youssef et al., is a distributed algorithm that enables a set of n nodes to determine the RF propagation delays between every pair

of nodes, from which the inter-node distances and hence the spatial topology can be determined. PinPoint's accuracy is roughly 1 to 3 meters. In [12] the authors proved that efficient location discovery can be achieved in sensor networks without using beacons. The basic observations in order to achieve this were that firstly, it is quite common that sensors are deployed in groups and secondly, the coordinates of the deployment points are usually known a priori. Using this knowledge, the location discovery problem could be modeled as a statistical estimation problem and nodes were able to discover their location by observing group memberships of their neighbors.

The AT&T Cambridge Laboratory's Active Badge location system [33, 32] and the more recent Active Bat system [2] are two of the first systems in the field. Both systems require specialized hardware in order to determine a device's location. 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 lighting and direct sunlight, because of the spurious infrared emissions these light sources generate. Active Bat uses an ultrasound time-of-flight technique to provide accurate physical positioning. Users and objects have to carry Active Bat tags, emitting an ultrasonic pulse to a grid of ceiling-mounted receivers and a simultaneous "reset" signal over a radio link. Each ceiling sensor measures the time interval from reset to ultrasonic pulse arrival and computes its distance from the Bat. Active Bat 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 accuracy within 9 cm of the true position for 95% of the measurements. The ParcTab [40] is a Personal Digital Assistant (PDA) that uses an infrared-based cellular network for communication. The infrared transmissions from ParcTabs can be used to determine their locations in the same way as Active Badges are located. Schilit et al. [41] further developed the use of ParcTab and its positioning capability to investigate context-aware computing applications. The Cricket Location Support System [25, 26] also uses ultrasound emitters and embeds low-cost RF receivers in the object being located. Cricket uses additional radio frequency signals to synchronize time measurements and to distinguish ultrasound signals that are a result of multi-path effects. The mobile object performs triangulation computations relative to the beacons. Cricket trades accuracy for simpler hardware and infrastructure. It does not require a grid of ceiling sensors with fixed locations as the Active Bat system does, but returns an estimation

of the user's position with a possible error of a four foot by four foot region, while the Active Bat has an accuracy of nine centimeters. Both of these systems provide excellent localization primitives by employing specialized hardware.

A different approach, SmartFloor [4], 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. Niculescu and Nath extended the APS system [22] by incorporating 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. In [42] the authors present an indoor positioning architecture, using 802.11 and Angle of Arrival, that requires only the placement of special VOR base stations, instead of a signal strength map. These spacial APs used, have a rotating directional antenna, enabling them to provide angle of arrival (AOA) and range measurements. A 2.1m median error, which is reported by the authors' system, is comparable to the original RADAR, but is achieved without requiring a map of the signal strength of the area.

Positioning using RFID technology is another way to locate users. Tesoriero et al. [30] propose a passive RFID-based indoor location system that is able to accurately locate autonomous entities, such as robots and people, within a physical-space. Landmarc [43] is an RFID based positioning scheme that uses reference tags at fixed locations, organized in a grid array. In Landmarc the proper placement of readers is essential, in order for sub-regions to be formed where each sub-region can be uniquely identified by the subset of readers that cover it. In this way, an RFID tag can be associated with a known subregion, based on the subset of readers that can detect it. In the Spot-On system [44], special tags use radio signal attenuation to estimate distance between tags. The aim is to localize wireless devices relative to one another, rather than to fixed base stations, allowing for ad-hoc localization. The authors exploit the density of tags and correlation of multiple measurements to improve both accuracy and precision. SpotOn's accuracy depends on the cluster size.

Many systems employ different modalities in order to achieve a higher accuracy. In Place Lab positioning system [45], laptops, PDAs and cell phones can estimate their position by listening for the cell IDs of fixed radio beacons, such as 802.11 APs, GSM cell phone towers, and fixed Bluetooth devices that already exist in the environment, and referencing the beacons' positions in a cached database. Experimental results indicate that 802.11 and GSM beacons are sufficiently pervasive in the greater Seattle area to achieve 20-30 meter median accuracy with nearly 100% coverage. The main goals of this work where two: a) to maximize coverage as measured by the percent of time a location fix is available in people's daily lives and b) to offer a low barrier

to entry for users and application developers thanks to the use of commodity hardware, privacy awareness, and straightforward interfaces. UbiSense [5] can provide an accuracy of 15 cm using a network of ultra wide band (UWB) sensors installed and connected into a building existing network. The UWB sensors use Ethernet for timing and synchronization. They are able to detect and react to the position of RF-tags based on time difference of arrival and angle of arrival. SurroundSense [7] is a mobile phone based system that explores logical localization via ambient fingerprinting. Fingerprints are generated using sound, cameras, accelerometers and Wi-Fi. Authors compare SurroundSense's performance, when using all these different technologies, with its performance, when using only Wi-Fi. An average accuracy of 87% and 70%, respectively, are reported.

2.3 Localization with computer vision

Many groups have experimented with the use of computer vision technology for implementing a tracking system. In order to achieve that cameras, mobile phones with integrated cameras, QR codes, bar codes and fiducial markers have been employed. Easy Living [46] uses motion tracking cameras to determine the distance of an object in a home environment. However, tracking works well with up to three people in the room, depending on how they behave. CyberCode [47] is a visual tagging system based on a 2D barcode technology. CyberCode tags can be recognized by cameras found in most mobile devices, and it can also be used to determine the 3D position of the tagged object as well as its ID number. One interesting application of Cybercode is its usage in an indoor navigation system, where cybercodes can be printed in labels in order to identify items of a museum. By scanning the codes, the visitor can retrieve id numbers and get guidance information. CyberCode can also be combined with other sensing technologies, such as a gyro sensor. In the proposed navigation system, once the global location and orientation information are retrieved from the recognized ID, the user can freely look around the environment by moving the device. Even when CyberCode tag is out of sight of the camera, the system continues to track the relative motion of the device by using the gyro sensor, and displays proper navigation information. In [48] the authors present TRIP, a low-cost and easily deployable vision-based sensor technology, that uses off-the-shelf hardware (low-cost CCD cameras and PCs) and printable 2-D circular markers for entity identification and location. Evaluation results indicate that when the target image (TRIPtag) occupies an area of at least 35*35 pixels, the recognition is successful on a 98% of the cases. This means that targets spotted in a frame of 640*480 pixel resolution are identified as long as they are within 3 meters distance. In [55]

the authors propose applications such as item selection and related on-line content retrieval, interaction with large scale displays, and document printing by using bar codes and camera phones. The proposed system induces a code coordinate system and is able to visually detect phone movements and to estimate the distance of the camera to the code.

2.4 Pervasive computing systems

Mobile devices, such as mobile phones and PDAs, have been widely used in the area of pervasive computing, where they are treated as people-centric sensors capable of aggregating participatory as well as sensory inputs from local surroundings. Recently, several mobile computing systems aim to provide location and context aware services to users. The CoolTown project [50] of the HP labs is a location-aware but ubiquitous system that supports nomadic users. The authors explore opportunities provided by the convergence of Web technology, wireless networks, and portable client devices to support "web presence" for people, places and things. The web technology is used in order to provide more effective means for ad hoc access to services, based on techniques of embedding and recovering URLs on objects and in places, and to provide location-awareness by offering location-dependent information resources to users. In order to demonstrate their idea, the authors equipped the HP labs with beacons (e.g. infrared transceivers) and tags and created 3 demo applications, the Cooltown museum, the Cooltown bookstore, and the Colltown conference room. The Cooltown project has as a result the creation of a virtual bridge between physical entities (including users) and electronic services. A similar effort is presented in [51], where the Physical Browsing concept using an RFID-reader is realized. With the reader attached to the mobile phone, digital services embedded in the environment can be invoked. The AURA project [52] by Smith et al. is a system that links online content to physical objects. It is implemented with commercially available pocket computers, using integrated bar code scanners, wireless networks, and web services. In [54] barcode stickers attached to physical objects, acting as bookmarks to the worldwide web are proposed. In [49], the notion of contextual QR Codes that merge a public QR Code and private information, in order to provide data related to a particular context, is presented. The private part can be one or more information among the subsequent: user's profile, current task, device used, location, time and environment of the interaction.

Interaction with real-world objects is also available, using mobile phones. In [53] the authors propose the idea of user interaction through a mobile device with services that are represented by a poster. Connection of the mobile device with the poster is

enabled through one of the following ways: camera, Near Field Networks like WLAN and blueetooth, user input (user enters a code), usage of localization functionalities of mobile networks like GPRS or UMTS, where only a specific set of services can be provided to the user, or RFID. A similar effort is presented in [56]. The idea described by the authors is to use the built-in cameras of consumer mobile phones as sensors for 2-dimensional visual codes in order to retrieve object-related information and functionality. The recognition algorithm used, simultaneously detects multiple codes, determines the orientation of the codes, and computes the coordinates of the target aimed at in the coordinate system induced by the code. The approach presented in [57] is also similar: a visual code system that provides a number of orientation parameters, such as target pointing, rotation, tilting, distance, and relative movement is employed. A set of fundamental physical gestures, that form a basic vocabulary for describing interaction, is defined and richer interaction sequences can be constructed by combining these few fundamental interaction primitives.

Chapter 3

Positioning with IEEE802.11

The fingerprinting technique is a well studied location sensing method which is based on the fact that received signals at different locations possess different electromagnetic characteristics. One of the major advantages of this technique is that it can exploit already existing radio infrastructures, like IEEE802.11 or GSM, so it is easily deployable and does not depend on specialized hardware.

3.1 Generation of statistical-based fingerprints

A wireless device, that is located at a specific position, scans all the available channels and receives beacons from the APs that are in range. It measures and records the RSSI value from each beacon. The recorded RSSI values are used for the construction of a statistical fingerprint that describes this specific position. The physical space is represented as a grid of cells with fixed size and well-known coordinates. Location fingerprinting is composed of two phases: offline or training phase and online or runtime phase. During the training phase the wireless device collects RSSI measurements at predefined cells of the grid (*training measurements*). A statistical-based signature (*training signature*) is constructed for each one of these cells (*training cells*). During the runtime phase, the mobile client also acquires RSSI measurements from APs (*runtime measurements*) and constructs a *runtime signature* on-the-fly, by applying the same statistical method. The runtime signature is then compared with all the training signatures. The cell with a training signature that has the smallest distance in signal space from the runtime signature is reported as the estimated position.

The statistical-based fingerprint of a cell can be generated using various methods such as *percentiles, confidence intervals* or *mahalanobis distance*. A vector containing the appropriate statistical information, according to the method that is used,

describes each training cell. Each entry of the vector corresponds to one AP. For example $< i, , j, ss_T^1, ss_T^2, ..., ss_T^k >$ represents the training fingerprint of the cell with coordinates i,j and ss_T^k constitutes the statistical information used for the k-th AP. In the runtime phase a similar vector is created by the RSSI measurements collected at the mobile client's position: $< ss_R^1, ss_R^2, ..., ss_R^k >$ represents the runtime signature, where ss_R^k is the runtime statistical information used for the k^{th} AP. The next paragraphs present the various methods for generating statistical-based signatures.

3.1.1 Confidence Interval

A confidence interval is an interval in which a measurement or trial falls corresponding to a given probability. The probability or *confidence level* that is most commonly used is 95%. An interval calculated with 95% confidence level indicates the range in which 95% of the measurements belong. Let us denote with $T_i^-(c)$ and R_i^- the lower bounds of the training and runtime confidence intervals, respectively and with $T_i^+(c)$ and $R_i^+$the upper bounds. Then:

- $[T_i^-(c),\ T_i^+(c)]$ indicates the confidence interval for AP i at cell c during the training phase.

- $[R_i^-,\ R_i^+]$ indicates the confidence interval for AP i at the unknown position during the runtime phase.

The signature of a cell is a vector of confidence intervals, each corresponding to an AP. Each confidence interval is generated using the RSSI values of the beacons received from the corresponding AP. The training signature of cell c is $< [T_1^-(c), T_1^+(c)]$, $[T_2^-(c), T_2^+(c)]$,...,$[T_N^-(c), T_N^+(c)] >$, where N is the total number of APs. The runtime signature at the unknown position is $< [R_1^-, R_1^+], [R_2^-, R_2^+], ..., [R_N^-, R_N^+] >$, where N indicates again the number of APs.

After the generation of the fingerprints, the *voting process* follows, where the runtime fingerprint is compared with the training fingerprint of each cell. An AP (e.g., i) participates in this technique by assigning a vote (weight) for a cell (e.g., c) that depends on the percentage of overlapping of its training confidence interval $\left[T_i^-(c), T_i^+(c)\right]$ with the runtime confidence interval $\left[R_i^-, R_i^+\right]$. By adding the votes of all the APs, the confidence-interval based approach computes a weight w(c) for cell c that indicates its likelihood to be the unknown position. More precisely, the vote that is assigned to cell c by the i-th AP is estimated as follows:

$$\begin{cases} \dfrac{T_i^+(c) - R_i^-}{R_i^+ - T_i^-(c)} & if \quad T_i^-(c) < R_i^- < T_i^+(c) < R_i^+ \\ \dfrac{R_i^+ - T_i^-(c)}{T_i^+(c) - R_i^-} & if \quad R_i^- < T_i^-(c) < R_i^+ < T_i^+(c) \\ 1 & if \quad R_i^- \leq T_i^-(c) < T_i^+(c) \leq R_i^+, \\ & or \quad T_i^-(c) \leq R_i^- < R_i^+ \leq T_i^+(c) \\ 0 & if \quad R_i^- < R_i^+ \leq T_i^-(c) < T_i^+(c), \\ & or \quad T_i^-(c) < T_i^+(c) \leq R_i^- < R_i^+ \end{cases} \tag{3.1}$$

At the start of the voting process each cell has a zero weight. In case of partial degree of overlapping of the two confidence intervals (runtime and training), w(c) is increased by the ratio of this degree of overlapping. In case that the training confidence interval is included in the runtime confidence interval or the runtime confidence interval is included in the training confidence interval, w(c) is increased by 1. Finally, in case of no degree of overlapping, no vote is added to cell c. After the completion of this process, the cell with the maximum weight is reported as the estimated position.

In a different approach, the cells are sorted in an descending order with respect to their weights. The weights of the top-k cells are normalized, so that they sum up to 1 and a weighted average of the top-k cells in the list is reported as the final location of the user. This is a k-nearest neighbors approach and will be referred as *k-nn weighted average*.

The main drawback of the weight as defined in (3.1) is its sensitivity to the relative position of the confidence intervals' endpoints, and even a small displacement of an endpoint may affect significantly the value of the weight. Also there are cases where the rule in (3.1) may result in equal weights between the unknown runtime cell c^* and two completely distinct training cells t_1, t_2. In the following, a typical example is presented for each case. For convenience, consider the simplified scenario of a single AP and the signatures of one runtime and two training cells, $c^* \mapsto [R^-(c^*), R^+(c^*)]$, $t_1 \mapsto [T^-(t_1), T^+(t_1)]$, $t_2 \mapsto [T^-(t_2), T^+(t_2)]$.

- ***Example* 1**: the endpoints of the confidence intervals of the cells c^* and t_1 satisfy the following relations:
  case A: $T^-(t_1) < R^-(c^*) < R^+(c^*) < T^+(t_1)$
  case B: $T^-(t_1) < R^-(c^*) < T^+(t_1) < R^+(c^*)$
  with $R^-(c^*) = T^+(t_1) - \epsilon$, $R^+(c^*) = T^+(t_1) + \epsilon$
  that is, the two cases differ by a small displacement of the runtime confidence interval by ϵ. Although the two cases are not that different with respect to the

RSSI measurements, in case A the weight is equal to 1, while in case B the weight is equal to

$$\frac{T^+(t_1) - R^-(c^*)}{R^+(c^*) - T^-(t_1)} = \frac{T^+(t_1) - T^+(t_1) + \epsilon}{T^+(t_1) + \epsilon - T^-(t_1)} = \frac{\epsilon}{T^+(t_1) - T^-(t_1) + \epsilon} \ .$$

Clearly, when $\epsilon \to 0$ the weight in case B is equal to 0. That is, even if the unknown runtime cell c^* coincides with the training t_1, a small variation of the RSSI measurements may affect significantly the corresponding weight.

- ***Example* 2**: the endpoints of the confidence intervals of the cells c^*, t_1 and t_2 are related as follows,
  $T^-(t_1) < R^-(c^*) < T^+(t_1) < R^+(c^*)$, $R^-(c^*) < T^-(t_2) < R^+(c^*) < T^+(t_2)$
  with
  $|T^-(t_1) - R^-(c^*)| = |R^-(c^*) - T^-(t_2)|$
  $|R^-(c^*) - T^+(t_1)| = |T^-(t_2) - R^+(c^*)|$
  $|T^+(t_1) - R^+(c^*)| = |R^+(t_2) - T^+(c^*)|$
  By substituting in (3.1) the corresponding weights, that is, the values of the first two ratios are equal, and thus the confidence interval method would assign the same weight to both training cells, being unable to distinguish between t_1 and t_2.

3.1.2 Percentiles

This approach is similar to the confidence-interval one. However, instead of using confidence intervals for constructing the fingerprints, percentiles are employed. A set of percentiles can capture more detailed information about the signal strength distribution than confidence intervals, and thus, result to more accurate fingerprints. The weight of a cell c, *w(c)*, is computed as follows:

$$w(c) = \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{p} (R_j^i - T_j^i(c))^2} \tag{3.2}$$

where N is the number of APs, p the number of percentiles, R_j^i the j-th percentile of runtime measurements from the i-th AP and $T_j^i(c)$ the j-th percentile using the training measurements from the i-th AP at the cell c. The number of percentiles used per AP is 10 (p=10). The percentiles used are 10%,20%...100%. The cell with the minimum weight is reported as the estimated position. In the k-nn weighted average approach the cells are sorted in an ascending order with respect to their votes and

are weighted according to them, so a weighted average of the top-k cells in the list is reported as the final location of the user.

3.1.3 Mahalanobis Distance

Mahalanobis or quadratic distance measures the separation of two groups of objects. It can be used to classify a test point as belonging to one of N classes, by estimating the covariance matrix of each class and classifying the test point as belonging to that class for which the Mahalanobis distance is minimal. This fingerprinting method takes into account the interdependencies among the RSSI measurements in a cell from various APs, by using the covariance as metric. These interdependencies provide information about the topology of the environment.

The Mahalanobis distance between two groups of variables $G_1 = \{x_1, x_2, \ldots, x_n\}$ and $G_2 = \{y_1, y_2, \ldots, y_n\}$ with mean vectors $\bar{x}$ and $\bar{y}$ is defined as:

$$d(G_1, G_2) = \sqrt{(\bar{x} - \bar{y})^T S^{-1} (\bar{x} - \bar{y})}$$

where T denotes matrix transpose and S^{-1} represents the inverse of the common covariance matrix of G_1 and G_2.

The common covariance matrix S is estimated from samples of sizes n_1 and n_2 from G_1 and G_2, yielding sample covariance matrices S_1 and S_2, respectively. Then S is computed as a weighted average of S_1 and S_2, using the pooled estimate:

$$S = \frac{(n_1 - 1)S_1 + (n_2 - 1)S_2}{n}$$

where $n = n_1 + n_2 - 2$.

The training signature of cell c is $\{\vec{\mu_T}(c), S_T(c)\}$, where $\vec{\mu_T}(c) = [\mu_1, \mu_2, \ldots, \mu_N]$ is a vector containing the mean values of RSSI training measurements per AP. More precisely, μ_i represents the mean RSSI value of the i-th AP, while N is the total number of APs. $S_T(c)$ is the covariance matrix of cell c, where $S_T^{i,j}(c)$ captures the covariance between the i-th an the j-th AP. Similarly, at the runtime phase the signature at the unknown position is indicated as $\{\vec{\mu_R}, S_R\}$, where $\vec{\mu_R}$ is the vector containing the mean RSSI runtime values per AP and S_R is the covariance matrix. If we denote as $S_{c,u}$ the common covariance matrix between training cell c and the runtime cell (unknown position) u, then mahalanobis distance between cells c an u takes the form:

$$d_{c,u} = \sqrt{(\vec{\mu_T}(c) - \vec{\mu_R})^T S_{c,u}^{-1} (\vec{\mu_T}(c) - \vec{\mu_R})} \tag{3.3}$$

The weight of a cell c is the estimated mahalanobis distance from the unknown position. The cell with the minimum weight is reported as the estimated position. If the k-nn weighted average approach is used, a weighted average of the top-k cells having the lowest weights is reported as the final location of the user.

3.2 Selection of the appropriate cell

3.2.1 Flat approach

In the *flat* approach the weights of the training cells are estimated with one of the methods described in section 3.1. A cell from the entire grid is reported as the user's position. The selected cell corresponds to the cell with the minimum weight for percentiles and mahalanobis distance methods or maximum weight for confidence intervals. In the case of the k-nn weighted average approach, a weighted average of the top-k cells having the lowest (percentiles, mahalanobis distance) / highest (confidence intervals) weights is reported as the final location of the user.

3.2.2 Iterative multilayered approach

In this approach the physical space is divided into overlapping regions of equal size. The algorithm estimates the region at which the user is located in its first phase and in its second phase reports a cell from within the selected region as the user's location, by employing the underlying fingerprinting method. The sum of the votes from cells that belong to each different region, are taken into account in order to select the correct region. So, when the mahalanobis distance metric or percentiles are used, the region with the smallest sum of votes is selected, while for confidence intervals the region with the largest sum of votes is selected. Hence, if for example the percentiles fingerprinting technique is used, in each iteration of the multilayered algorithm the region r* closest to the unknown position is given by:

$$r^* = \arg \min_{m=1,\ldots,G} \sum_{k=1}^{C_m} \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{p} (R_j^i - T_{j,k}^i)^2}$$

where G is the number of overlapping regions, C_m is the number of cells included in the m-th region and $T_{j,k}^i$ is the j-th training percentile of the i-th AP from the k-th cell within the region.

In each iteration of the algorithm the selected region is divided in two sub-regions (G=2) and the multilayered algorithm is applied on them. More precisely, the algo-

rithm is firstly initialized with the whole testbed as input region and then proceeds iteratively along one dimension (x axis) of the grid representing the physical space. The selected region is given as input in the algorithm, which now proceeds iteratively along the second dimension (y axis) of the grid representing physical space. The degree of overlapping N defines the number of common rows or columns of cells that the two neighboring sub-regions have: when the algorithm is applied on the x axis of the grid, the two neighboring regions have N common columns, while when it is applied on the y axis of the grid, they have N common rows. The iteration of the algorithm stops, when a region can not be further divided into sub-regions which have N common rows or columns of cells.

The regions are overlapping with each other in order to minimize the probability of selecting a wrong sub-region (not including the actual position of the user) when the user is located near the boundaries of the neighboring regions. The degree of overlapping is an important parameter of the multilayered algorithm: we expect that the higher the number of overlapping rows/columns is, the lower the probability of selecting a region that doesn't include the actual position of the user is. However, when a large number of common rows/columns between two neighboring regions is required, the algorithm results in a larger region as compared to the region selected when the degree of overlapping N is small. The underlying fingerprinting technique has a higher probability of failing to select the correct cell (actual position of the user) when the number of training cells included in the estimated region is large. Obviously there is a trade-off between the probability of selecting the correct region and probability of selecting the correct cell from within the region. We experiment with different degrees of overlapping in order to examine this trade-off.

3.3 Principal Component Analysis

Principal component analysis (PCA) is a dimensionality reduction procedure. It is useful when we have obtained data on a number of variables and we believe that there is some redundancy in those variables. Especially when the variables are measuring the same construct, as in our case, there is a probability that some of the variables are highly correlated with one another. Dimensionality reduction is achieved by transforming to a new set of variables, the *principal components* (PCs), which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables. Then a subset of the original variables can be extracted, by taking into account only the variables that are related to the first few PCs. Our goal is to reduce the computational complexity of the positioning algorithms by identifying the most discriminative APs and omitting redundant ones.

3.3.1 Mathematical Definition of PCA

PCA is mathematically defined as an *orthogonal linear transformation* that transforms the data to a *new coordinate system* such that the greatest variance by any projection of the data comes to lie on the first coordinate (first PC), the second greatest variance on the second coordinate, and so on. In other words, the goal of principal component analysis is to identify the most meaningful *basis* to re-express a data set. The hope is that this new basis will filter out the noise and reveal hidden structure.

Change of basis

Let X be the original data set, where each column is a single sample of the data set. X is an m $\times$ n matrix where m is the number of variables and n is the number of samples per variable. Let Y be another m $\times$ n matrix related to X by a linear transformation P. X is the original recorded data set and Y is a new representation of that data set. So, we look for a transformation matrix P so that:

$$Y = PX \tag{3.4}$$

We are going to produce m new variables from the m original variables such that:

$$\begin{aligned} y_1 &= p_1x_1 + p_1x_2 + \ldots + p_1x_m \\ y_2 &= p_2x_1 + p_2x_2 + \ldots + p_2x_m \\ &\vdots \\ y_m &= p_mx_1 + p_mx_2 + \ldots + p_mx_m \end{aligned}$$

where y_m's are uncorrelated (orthogonal), y_1 explains as much as possible of original variance in data set, y_2 explains as much as possible of remaining variance etc. Geometrically P can be defined as a rotation and stretch which transforms X into Y. The rows of P, $\{p_1, p_2, \ldots p_m\}$, are a set of new basis vectors for expressing the columns of X. The row vectors $\{y_1, y_2, \ldots y_m\}$ will become the *principal components* of X. The next question to be answered is what is a good choice of the basis P. This depends on what features we would like Y to exhibit. The next section will present the assumptions behind PCA, which will lead to a reasonable result regarding the selection of basis P.

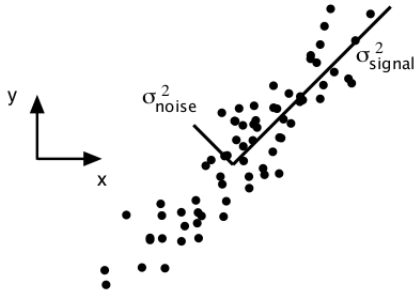


Figure 3.1: Plot of measurements from two random variables x,y. The signal and noise variances σ^2_{signal} and σ^2_{noise} are graphically represented by the two lines subtending the cloud of data.

3.3.2 Assumptions behind PCA

One of the desired features of Y is having a lower *measurement noise,* compared to the original dataset. There exists no absolute scale for noise but rather all noise is quantified relative to the signal strength. A common measure is the signal-to-noise ratio (SNR), or a ratio of variances σ^2:

$$SNR = \frac{\sigma^2_{signal}}{\sigma^2_{noise}}$$

A high SNR ($\gg$1) indicates a high precision measurement, while a low SNR indicates very noisy data. In figure 3.1 measurements of two random variables, x and y, are plotted. The variance due to the signal and noise are indicated by each line in the diagram. We assume that directions with largest variances in our measurement space contain the dynamics of interest. This assumption suggests that the basis we are searching is not the basis corresponding to the original variables x,y. Maximizing the variance (and by assumption the SNR) corresponds to finding the appropriate rotation of the original basis. This intuition corresponds to finding the direction indicated by the line σ^2_{signal} in figure 3.1. In the 2-dimensional case of figure 3.1 the direction of largest variance corresponds to the best-fit line for the data cloud.

As it has been already mentioned, PCA can also help in identifying redundant variables. In case that two highly correlated variables exist in our data set, we could omit either one of them with little loss of information. Figure 3.2 might reflect a range of possible plots between two variables r_1 and r_2. The best-fit line $r_2 = kr_1$ is indicated by the dashed line. Figure 3.2(a) depicts two variables with no apparent relationship. In other words, r_1 is entirely uncorrelated with r_2. On the other extreme, figure 3.2(c)

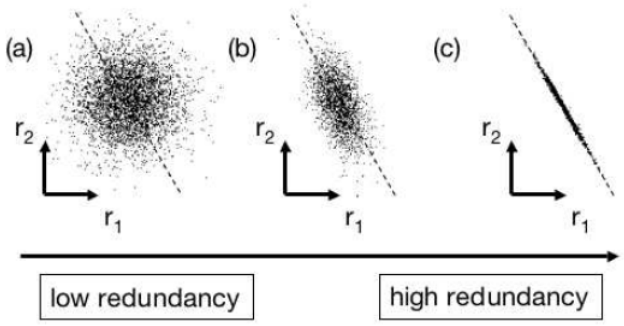


Figure 3.2: A spectrum of possible redundancies in data from two variables r1 and r2. From left to right: (a) no correlation, (b) medium correlation, (c) high correlation.

depicts two highly correlated variables. There is a redundancy here, because one can calculate r_1 from r_2 (or vice versa) using the best-fit line.

In a 2 variable case it is simple to identify redundant cases by finding the slope of the best-fit line and judging the quality of the fit. We need a way quantify and generalize these assumptions, namely variance maximization and identification of redundant variables, to arbitrarily higher dimensions. Let us can define an m $\times$ n matrix *X:*

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

where m is the number of variables and n is the number of measurements per variable. This means that each row of X corresponds to all measurements of a particular variable x_i. Without loss of generality, let us also assume that each variable x_i has *zero mean.* This means that the mean value, $\bar{x_i}$, has been substracted from every variable in the original dataset. We need to measure the degree of linear relationship between any set of variables, so we will use the *covariance matrix* of X, defined as:

$$C_X = \frac{1}{n} X X^T$$

where the ij^{th}element of C_X is the dot product between the vector of the i^{th} variable with the vector of the j^{th} variable. C_X is a square symmetric m $\times$ m matrix, the

diagonal terms of which are the variance of particular variables and the off-diagonal terms are the covariance between variables. It has been already mentioned that one of the goals of PCA is to minimize redundancy, measured by the magnitude of the covariance. We are not interested whether two variables are positively or negatively correlated, but instead we take into account the absolute magnitude of the covariance. As a result, large magnitudes in the off-diagonal elements of a covariance matrix correspond to high redundancy. The other goal of PCA, is to maximize the signal (or equivalently minimize the noise in measurements), which is measured by the variance. So, in the diagonal terms of a covariance matrix, by assumption, large values correspond to interesting structure. As a result, if we transform the original dataset X with a transformation matrix P, so that Y=PX, we want the covariance matrix of Y, C_Y, to fulfill the goals of PCA. More precisely, we want C_Y to have the following properties:

- All off-diagonal terms in C_Y should be zero. Thus, C_Y must be a diagonal matrix. Or, said another way, Y is decorrelated.
- Each successive dimension in Y should be rank-ordered according to variance.

One last assumption made in PCA, is that the basis vector$\{p_1, \ldots, p_m\}$ is orthonormal, which means that P is an orthonormal matrix.

3.3.3 Solving PCA using eigenvector decomposition

To summarize from the previous section, we are looking for an orthonormal matrix P in Y = PX such that $C_Y = \frac{1}{n}YY^T$ is a diagonal matrix. By rewriting C_Y in terms of the unknown variable, we have:

$$\begin{aligned} C_Y &= \frac{1}{n}YY^T \\ &= \frac{1}{n}PX(PX)^T \\ &= P\frac{1}{n}XX^TP^T \\ C_Y &= PC_XP^T \end{aligned} \tag{3.5}$$

We know from linear algebra that every symmetric matrix A can be diagonalized. More precisely, the eigendecomposition of A takes the form:

$$A = EDE^T \tag{3.6}$$

where E is an orthogonal matrix, the columns of which are eigenvectors of A and D is real and diagonal, having the eigenvalues of A on the diagonal. C_X is a symmetric matrix, so it holds that $C_X = EDE^T$. We can also select matrix P to be a matrix where *each row p_i is an eigenvector of C_X*. By this selection $P = E^T$. Moreover, it is known that the inverse of an orthogonal matrix is its transpose, so $P^{-1} = P^T$. By substitution in equation 3.5 we have:

$$\begin{aligned} C_Y &= PC_XP^T \\ &= P(EDE^T)P^T \\ &= P(P^TDP)P^T \\ &= (PP^T)D(PP^T) \\ &= (PP^{-1})D(PP^{-1}) \\ C_Y &= D \end{aligned}$$

It is evident that the choice of P diagonalizes C_Y. We could summarize by saying that:

- The new basis is formed by the eigenvectors of $C_X = \frac{1}{n}XX^T$ or the rows of P.
- C_Y contains the eigenvalues of C_X in its diagonal, with the i^{th} diagonal value indicating the variance of X along p_i.
- By projecting the initial variables, $\{x_1, x_2, \ldots x_m\}$, on the new basis we get the principal components $\{y_1, y_2, \ldots y_m\}$

Chapter 4

Performance Evaluation

In this chapter we present an extensive evaluation of the positioning algorithms presented in chapter 3. The positioning system is evaluated in two different environments: TNL-FORTH and Creteaquarium. Moreover, it has been tested under different scenarios (busy and quiet) in order to evaluate the impact of human presence on RF signals.

Data collection

In order to generate the training signatures, signal-strength values at various predefined cells of the grid were collected. The runtime measurements were collected at 35 random cells, scattered in the grid. The trainer remained still for approximately 60s and 30s to collect beacons at each position during training and runtime, respectively. During this time enough RSSI values were collected, in order to generate the appropriate statistical fingerprints (more than 100 and 200 RSSI values per AP at each cell for runtime and training, respectively). To capture signal strength values, *iwlist*, which polls each channel and acquires the MAC address and RSSI measurements from each AP (in dBm), and *tcpdump*, a passive scanner relying on libpcap, for the retrieval of each packet were used. A Sony Vaio and a Toshiba laptop with the same wireless adapter (ipw2200) were used for the collection of both training and run-time signal-strength values.

4.1 Evaluation at FORTH

The evaluation at FORTH took place in the Telecommunication and Networks Lab (TNL), an area of 7m×12m, which was discretized in a grid structure with cells of

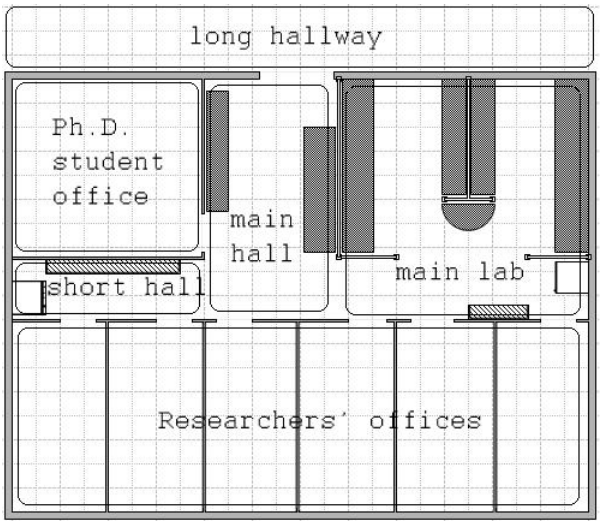


Figure 4.1: Grid-based representation of TNL testbed.

55cm×55cm. Figure 4.1 depicts the floor plan of TNL. During the runtime phase, we collected two data sets: one during a relatively busy period and another one during a quiet period. The busy period dataset was collected on a typical weekday at around 5 am, during which there were from 10 to 15 people in the laboratory, and several others walking in the hallways outside. The quiet period dataset was collected on Sunday, at around 2 am, and there was only one person in the laboratory, except the one collecting the measurements. The training set was common for both busy and quiet scenarios, and was collected on the same day as the quiet runtime dataset. The training dataset included measurements from 84 different cells. The total number of APs covering the area was 10 while on average 5.4 APs were detected at a given cell. The total human effort to collect the training and runtime datasets is estimated at 4 hours.

4.1.1 Comparison of fingerprinting methods

In order to evaluate the performance of the various fingerprinting methods, the *localization error* has been computed, measured as the Euclidean distance between the centers of the reported cell and the cell at which the mobile user was actually located at run time.

Figures 4.2(a) and 4.2(b) illustrate the localization error of the different signature-based approaches during quiet and busy periods, respectively. The percentiles outperform the confidence intervals and the mahalanobis in both periods. More specifically,

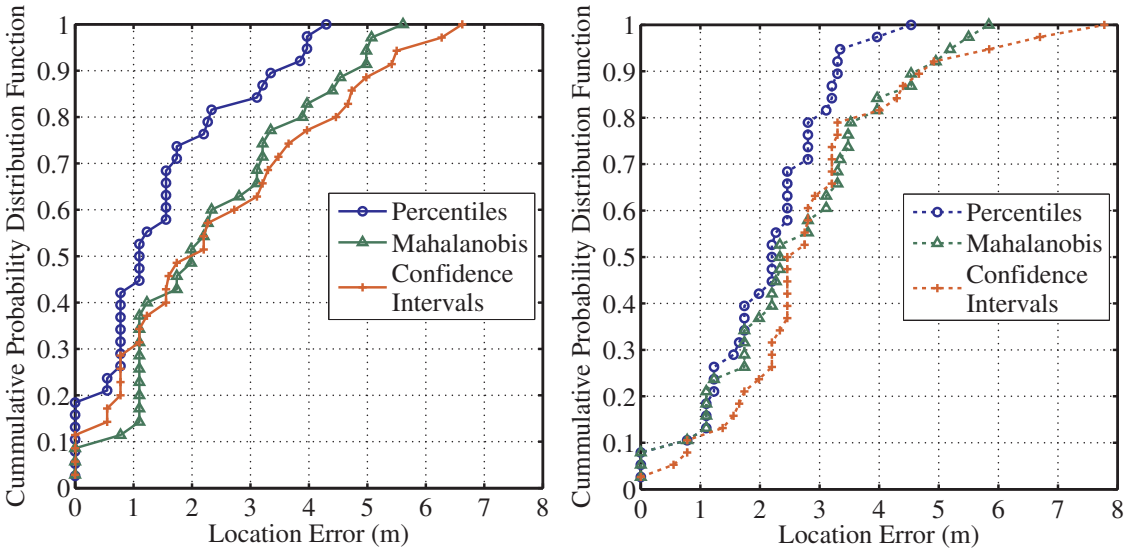


Figure 4.2: The location error in TNL during quiet (a) and busy (b) scenarios, respectively.

for the quiet period dataset, the median for the percentiles is 1.1, while the median for the mahalanobis and the confidence intervals (95%) is 2 and 2.1, respectively. For the busy period dataset, the median for the percentiles is 2.2, while the mahalanobis and the confidence intervals report a median of 2.3 and 2.5, respectively. The percentiles capture more detailed information about the signal strength distribution, so a smaller location error as compared to the confidence intervals has been expected. The mahalanobis distance depends on the sample mean and covariance of the input data. The covariance takes into account the interdependencies among the RSSI measurements at a certain position from the various AP, so this method also captures more detailed information about the signal-strength in comparison to the confidence intervals method. However, in the case of a noisy dataset of RSSI measurements from various APs, a metric based on the sample mean can be misleading. As a result, the mahalanobis distance method slightly outperforms the confidence intervals method but is not as effective as the percentiles method.

It has been expected that an increased number of people present in the laboratory, when the RSSI measurements were collected, would have a negative impact on the localization accuracy. Human bodies can absorb part of the signal-strength, so under the presence of several people, there will be fluctuations in the received signal strength from the same AP at a specific position. Figure 4.2 validates this hypothesis. The

impact from the presence of people is more prominent for the percentiles, that achieve the highest accuracy in the quiet scenario. The median location error reported by the percentiles for the quiet period is 1.1, while the median error for the busy period is 2.2. This method can accurately detect the user within 1.1 meters (or two cells apart) for 50% of the trials in the quiet scenario, but the presence of people makes it impossible to locate 50% of the runtime cells with such an accuracy. The impact from the presence of people is also evident for the mahalanobis distance method and the confidence intervals. When the mahalanobis is used, 40% of the runtime cells can be detected within 1.1 meters in the quiet scenario, while the same percentage of cells can be located within 2.2 meters in the busy scenario. In the confidence intervals 40% of the runtime cells can be located within 1.6 meters in the quiet scenario compared to an accuracy within 2.5 meters in the busy scenario. As regards the confidence intervals, it is noteworthy that the performance of the method in the region 70%-90% is better in the busy scenario than the corresponding performance in the quiet scenario. This means that there is a small number of cells, the position of which can be estimated with better accuracy in the busy scenario, despite the presense of people. This can be merely explained due to the problematic weight estimation in the confidence intervals method for some cells, as explained in section 3.1.1. A small change in one or more of the runtime confidence intervals at a cell due to interference in the busy scenario, will result in large change in the cell's weight, making it distinguishable from another neighboring cell, that has similar runtime confidence intervals with it at the quiet scenario.

A first observation, based on these preliminary results, is that the algorithms that have been used do not always estimate correctly the cell. Indoor RF propagation is strongly affected by obstacles (walls, human bodies), so it is not uncommon to have a cell which is located far away from the unknown position with a training fingerprint very close to the runtime fingerprint. Another common scenario is the following: the cells that have accumulated the highest (confidence intervals)/lowest (mahalanobis, percentiles) votes are all concentrated in an area, but the one selected is not the correct one. Instead, the cell with the 2nd or 3rd highest/lowest vote may be the correct one. It is obvious that it may be efficient to take into account more than one cells with training fingerprint close to the runtime fingerprint. A k-nn approach has been used, where the cells are sorted and weighted according to their votes, so that a weighted average of the top-k cells in the list can be reported as the final location of the user. The best choice of k depends upon the data and it is believed that the optimal k for most datasets is 10 or more. In our dataset k=5 is the more efficient. Figure 4.3 indicates the location error for percentiles and confidence intervals, when a weighted average of the top-five nearest neighbors in

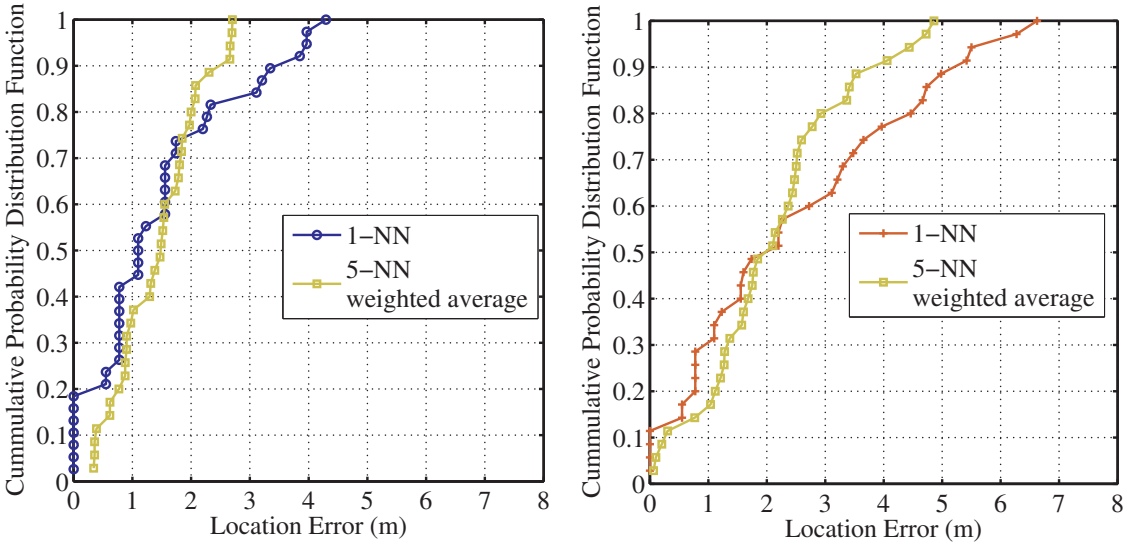
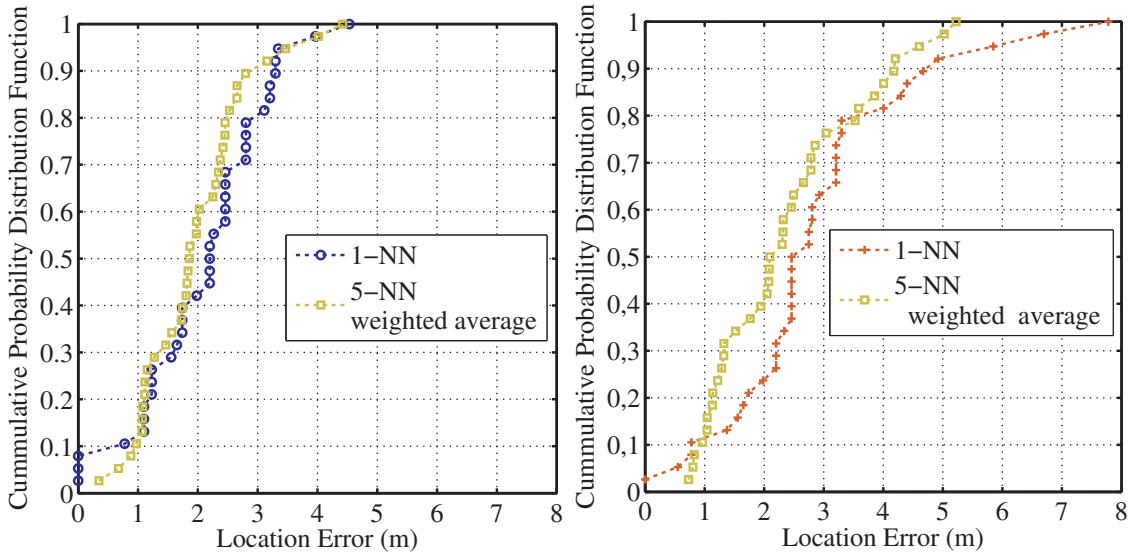


Figure 4.3: The location error in TNL during the quiet period for the 5-NN weighted average approach in the percentiles (a) and the confidence intervals (b), respectively.

signal space is used for the quiet dataset. The application of the 5-NN weighted average approach to the percentiles algorithm for the quiet period dataset, is only able to improve the 90th percentile and the maximum location error. There are cases where the exact position of the mobile user could be estimated (zero location error), when the 1-NN percentiles algorithm is used. However, the same position can not be accurately detected when the 5-NN weighted average approach is used, due to the fact that more than one cells are taken into account. In cases where the user's position is estimated with some error, the 5-NN approach may help in the reduction of that error. Hence, the 5-NN approach is expected to help more in the improvement of the accuracy of the confidence intervals algorithm, that is a less accurate method compared to the percentiles. As 4.3(b) demonstrates, the 5-NN weighted average method outperforms the original 1-NN approach for location errors larger than 2 meters. Figure 4.4 indicates the location error for the percentiles and the confidence intervals, when a weighted average of the top-five nearest neighbors in signal space is used for the busy dataset. The median for the percentiles, when the weighted 5-NN approach is used is 1.8, while the original algorithm, where 1-NN approach is used, has a median of 2.2. In the confidence intervals, the median when the weighted 5-NN approach is used is 2.1 while the median when 1-NN approach is used is 2.5. It is also



(a) Percentiles

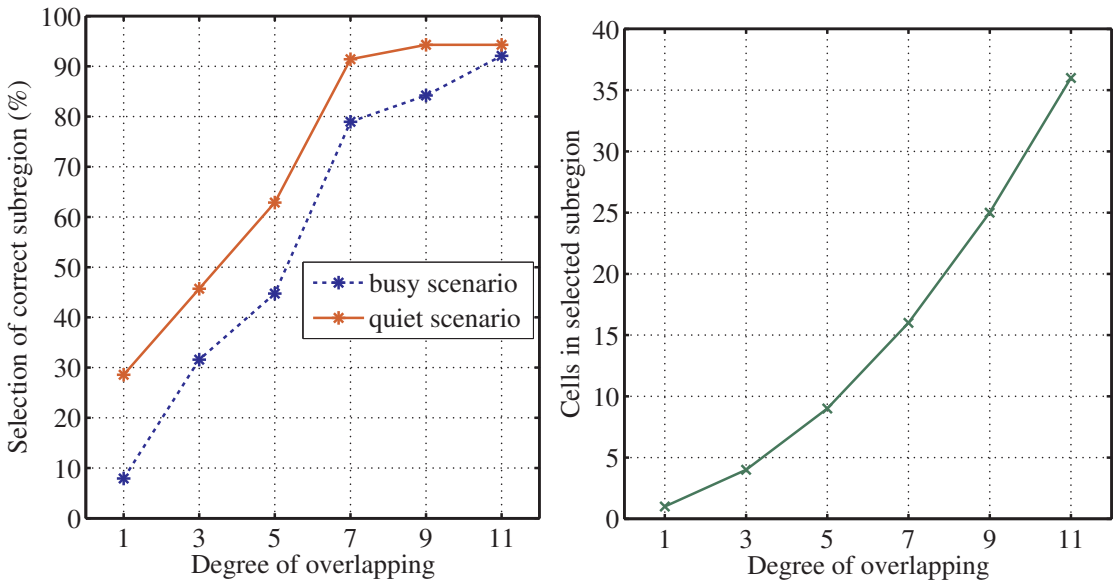
(b) Confidence Intervals

Figure 4.4: The location error in TNL during the busy period for the 5-NN weighted average approach in the percentiles (a) and the confidence intervals (b), respectively.

noteworthy that for the confidence intervals the maximum location error is 5.2 when the weighted 5-NN approach is used, while, when 1-NN approach is used, it is 7.8.

4.1.2 Iterative Multilayered Approach

It has been already mentioned that due to the radio propagation characteristics in the physical space, affected by transient phenomena, the various fingerprinting algorithms can not always estimate the correct cell. More specifically, there may exist a cell with training fingerprint very close to the runtime fingerprint, that belongs to a totally different region than the unknown position (e.g. the one is located in a hallway and the other in an office far away from it). The 5-NN weighted average approach tries to mitigate the impact of this phenomenon, by taking into account more than one cells in order to estimate the final position. However, a cell that is located far away from the unknown position, may still be taken into account with this approach. In order to avoid that, the proposed multilayered algorithm divides the physical-space iteratively into overlapping subregions. In its first phase tries to select the correct subregion, which is a small region that includes the unknown position, and in its second phase tries to estimate the correct position, which corresponds to a cell from



(a) Percentage of correct region estimation (b) Number of cells in selected region

Figure 4.5: Percentage of correct subregion estimation (a) versus number of cells in selected region (b) as a function of degree of overlapping during the quiet period in the TNL testbed.

within the subregion. This spatial aggregation is expected to reduce the likelihood of selecting a false region/cell (a region/cell that does not include/correspond to the actual position) over the correct one. This is due to the enhancement of the "weight" of a correct region in each iteration, by considering the signatures of the neighboring to the actual position cells, and the iterative elimination of incorrect regions.

The first question to be answered is what the proper degree of overlapping is. The degree of overlapping corresponds to the number of common rows and columns between the subregions and determines not only the probability for the selection of the correct subregion, but also the number of cells included in the selected region. More specifically, when the degree of overlapping is large, (e.g. 9 or 11 common rows/columns for the TNL testbed), the algorithm terminates after a few iterations because the selected region can not be further divided into smaller subregions that also have the same degree of overlapping. This means that the probability of selecting a wrong subregion (region that does not include the unknown position) at an iteration of the algorithm is smaller, as compared to a small degree of overlapping (e.g. 1 or 3), where the algorithms needs more iterations in order to select a subregion. On the

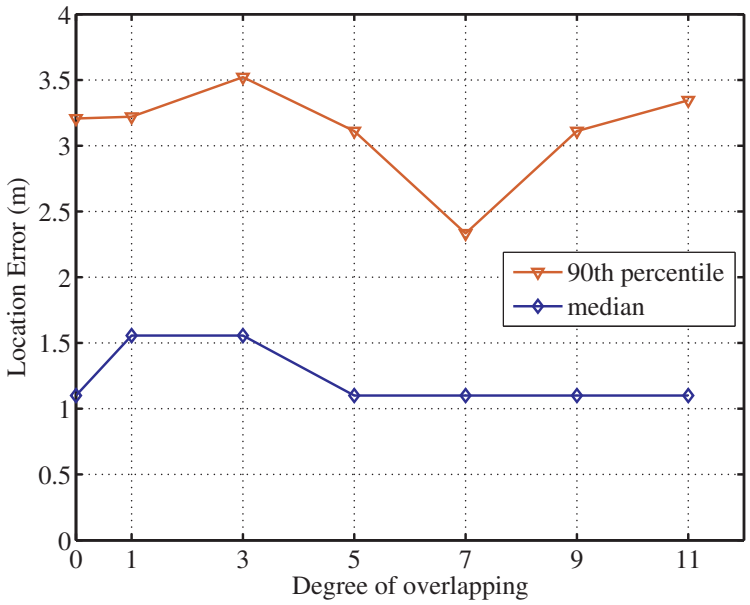
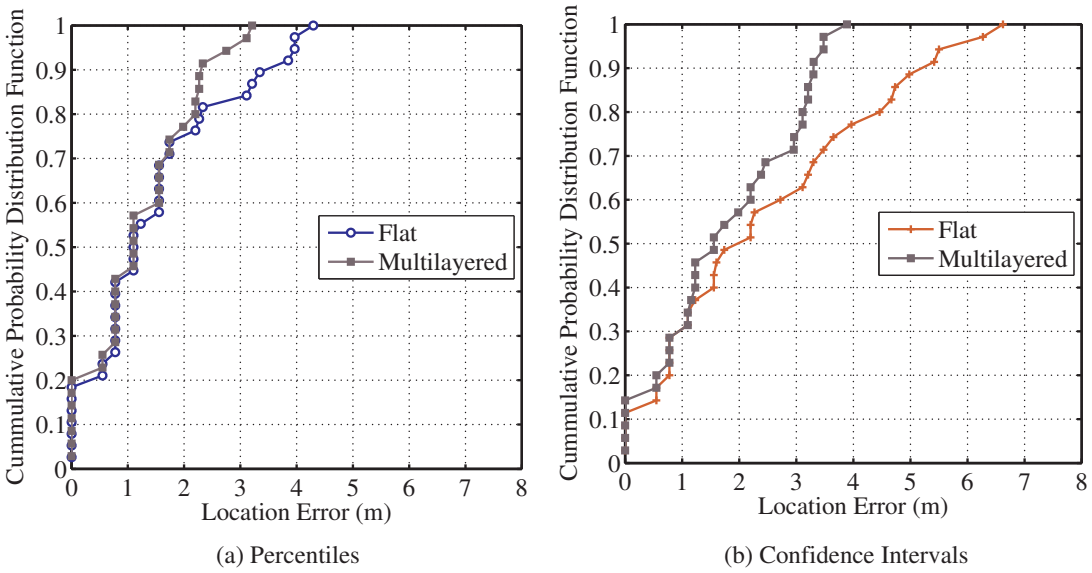


Figure 4.6: Median and 90th percentile of the location error in the TNL testbed during the quiet period as a function of degree of overlapping for the percentiles.

other hand, when large degree of overlapping is employed, the fingerprinting algorithm is applied to an area that includes more cells as compared to the region selected when a small degree of overlapping is used. The results illustrated in figure 4.5 show this *tradeoff* for the quiet and busy scenarios, when the percentiles fingerprinting method is used. The number of cells in the selected region (figure 4.5(b)) is the same for both scenarios. Figure 4.5(a) shows that the percentage for the selection of the correct subregion in the quiet scenario increases almost linearly up to degree of overlapping 7 (91%) and thereafter there is only a small increase for degree of overlapping 9 (94%) and no increase for degree of overlapping 11. On the other hand, the number of cells in the selected area (figure 4.5(b)) is 16 for degree of overlapping 7 and increases to 25 and 36 for degrees of overlapping 9 and 11, respectively. Figure 4.5 depicts a similar trend for the busy period: the percentage for the selection of the correct subregion in the busy scenario increases quickly up to degree of overlapping 7 (79%) and there is only a small increase to 84% for degree of overlapping 9 and to 92% for degree of overlapping 11 while the number of cells within the region continues to increase after degree of overlapping 7, by 9 cells in degree of overlapping 9 and 11 more cells in degree of overlapping 11.Degree of overlapping 7 achieves a high accuracy in the first step of the multilayered algorithm, namely the selection of the



(a) Percentiles (b) Confidence Intervals

Figure 4.7: The location error in the TNL testbed, when the multilayered approach is used during the quiet period for the percentiles (a) and the confidence intervals (b), respectively.

correct subregion, and in the second step the underlying fingerprinting algorithm tries to estimate the correct position of the user, by selecting among 16 cells. So degree of overlapping 7 is expected to have better performance in comparison to other degrees of overlapping. Another noteworthy point, is that degrees of overlapping 1 and 3 result in very small regions, containing 1 and 4 cells, respectively, so although the accuracy in the selection of the correct subregion is very low, the estimated position may still be only few cells far away from the unknown, given that a wrong subregion has not been selected in the early iterations of the algorithm. When a wrong subregion is selected in an early iteration, the error propagates at each iteration, so especially for small degrees of overlapping where more iterations are needed for the algorithm to finish, the performance of the multilayered algorithm may be poor. Figure 4.6 validates the previous hypothesis. In this figure the median and 90th percentile of the location error as a function of the degree of overlapping, for the percentiles in the quiet period, is plotted. Degree of overlapping 0 corresponds to the original implementation of the percentiles algorithm, where the fingerprinting method is applied to the entire area of the TNL. We refer to this approach as *flat.* The increase in the location error for both the median and the 90th percentile for degrees of overlapping 1 and 3 indicates that it is common for the multilayered algorithm, when small degree of overlapping are used,

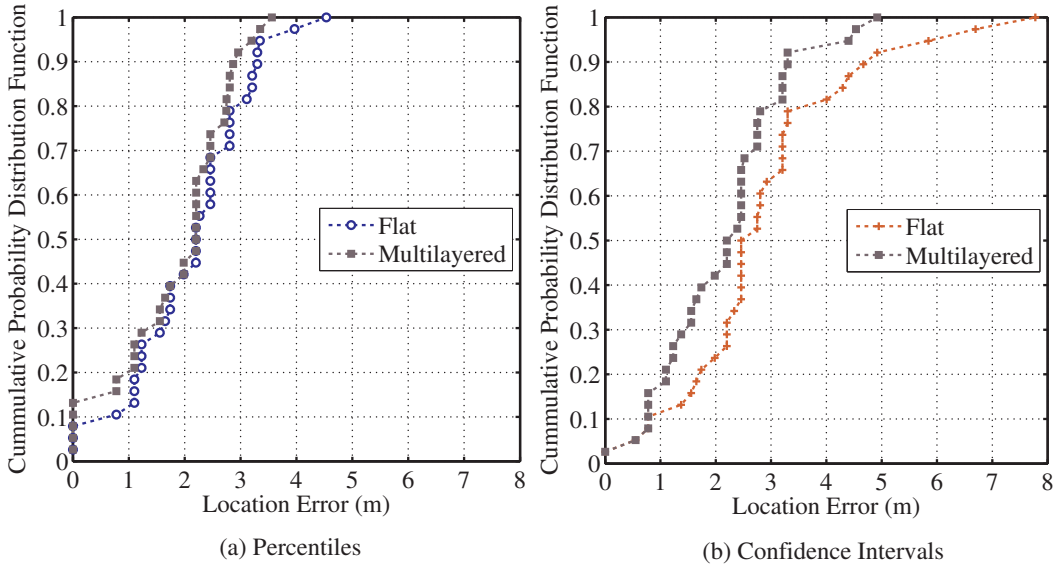


Figure 4.8: CDF of the location in the TNL testbed, when the multilayered approach is used during the busy period for percentiles (a) and confidence intervals (b), respectively.

to select a wrong subregion in the early iterations. Degree of overlapping 7 achieves a 2.3 m 90th percentile of location error, as compared to 3.2 when the multilayered algorithm is not used. From now on degree of overlapping 7 will be used.

Figures 4.7 and 4.8 show the performance of the multilayered algorithm in comparison to the flat implementation of the algorithm for the percentiles and the confidence intervals algorithms, in the quiet and busy scenarios, respectively. For the quiet period dataset, the multilayered algorithm is slightly better than the flat algorithm, when the percentiles fingerprinting technique is used. More precisely, although the two algorithms have the same median location error, the multilayered algorithm achieves a 90th percentile of 2.3 meters location error, while the flat algorithm has a 90th percentile of 3.2 meters (4.7(a)). The multilayered algorithm has been expected to improve accuracy for large location errors (when the estimated cell belongs in a different region than the unknown position). However, the percentiles algorithm has a very good performance especially for the quiet period: a median of 1.1 means that the estimated cell is located two cells away from the unknown position. As a result, we expect a more prominent improvement for the confidence intervals algorithm, that has a poor performance in comparison tothe percentiles, and for the busy period dataset, that contains a lot of noisy measurements due to the presence of people. Figures

4.7(b) and 4.8 validate this hypothesis. The improvement in the median location error for the confidence intervals method, when the miltilayered algorithm in the quiet period dataset is used, is 0.4 m, while for the 90th percentile of the location error the improvement is about 1.8 m. In figure 4.8 we also notice that the multilayered algorithm outperforms the flat implementation of the percentiles method, having a 90th percentile of 2.9 m, while the flat algorithm has a 90th percentile of 3.3 m. Finally, figure 4.8 demonstrates an improvement of 0.3 m in the median location error, when the multilayered algorithm for the confidence intervals in the busy period is used, whereas the same improvement for the 90th percentile of the location error is about 1.5 m.

4.2 Evaluation at Cretaquarium

Cretaquarium is the largest and most popular aquarium in Greece, covering an area of 1760 m^2. It consists of more than 40 tanks. Figure 4.9 depicts the floorplan of Creteaquarium. The physical space was represented as a grid with cells of 1mx1m. 7 IEEE802.11 APs were covering the whole testbed, out of which 3.4 on average can be detected at a given cell. Training and runtime signal-strength measurements were collected in December, January and February of 2011 for the entire testbed during two different periods, quiet and busy. During the quiet period there were only two people (the ones collecting the signal-strength measurements) most of the time in the aquarium. This is due to the lack of visitors in winter, especially in weekdays, except from scheduled school visits. During the busy period there was a scheduled visit of a class of students, so there were about 25 people near the trainers the whole time. The training set was common for both scenarios and was collected during different days of December and January 2011. We collected training signal-strength measurements at 215 different cells. There were required several visits to the aquarium in order to collect all the datasets and the human effort is estimated at about 17 hours.

Creteaquarium is a very complex environment, composed of many obstacles that contain large amounts of water. The attenuation of RF signals as they pass through water is very high. More precisely, as an RF signal enters a conductive medium, such as salt water, its intensity decreases logarithmically. This is known as the *skin effect* penetration depth, which is defined as the distance to which a radio wave can penetrate into a conductive medium (metal, salt water, ionosphere, etc.) leaving only 1/e (37%) of its initial intensity. As a result we expect a lower localization accuracy in this testbed in comparison to the TNL testbed, not only due to its larger size but also because of the larger attenuation in the received signal strength from the different APs in this environment.

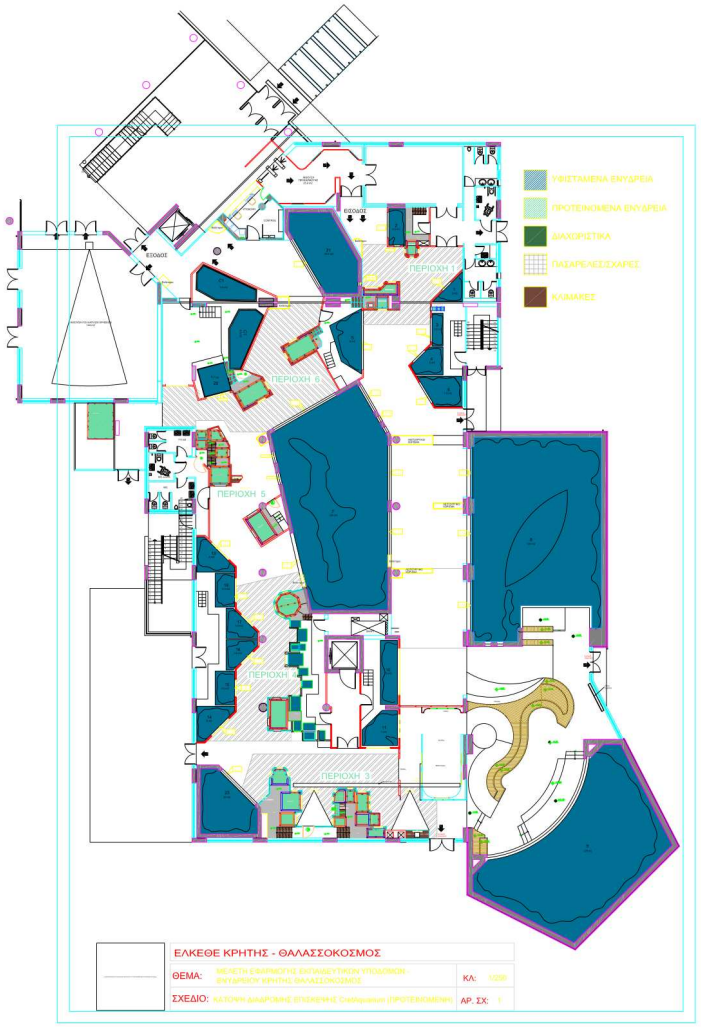


Figure 4.9: Creteaquarium testbed.

Once again, we computed the *localization error* in order to evaluate the performance of the different fingerprinting methods. Figure 4.10 illustrates the performance of the percentiles, the confidence intervals (95%) and the mahalanobis for the quiet period. The percentiles outperform the mahalanobis and the confidence intervals. More specifically, the percentiles have a median location error of 1.5 m, while the mahalanobis and the confidence intervals have a median location error of 2.1 m and 3 m respectively. It is noteworthy that the performance of the mahalanobis distance metric in this testbed is comparable to the performance of the percentiles, in com-

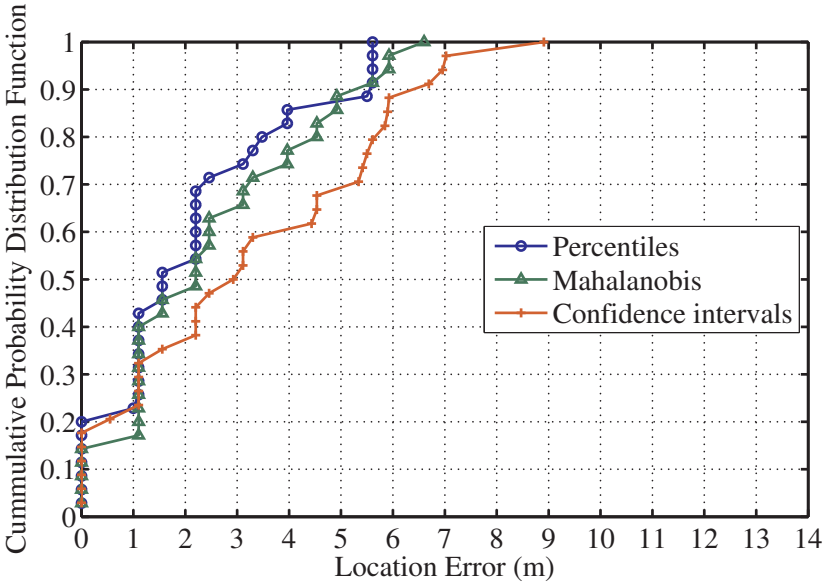


Figure 4.10: Location error in Creteaquarium for the quiet period.

parison to the quiet scenario for the TNL testbed, where the performance of the mahalanobis method was very close to the performance of confidence intervals. We can conclude from this that a fingerprinting method like the mahalanobis, that uses the covariance as a metric and takes into account the interdependencies among the RSSI measurements at a certain position from the various APs, is more efficient in a complex environment with many obstacles (tanks) that result in strong reflection an absorption of the RF signals. The reason is that methods that can model the environment more richly (percentiles, mahalanobis) have better performance in complex environments, where the noise in measurements is the result of many factors, as compared to less sophisticated methods. In figure 4.11 we can see the performance of the algorithms for the busy period dataset. The percentiles and the mahalanobis achieve the same median location error of 2.5 m, while the confidence intervals have a median of 3.2 m. As expected, the presence of people resulted in an increase in the median location and in an increase in large errors: for the confidence intervals, that is the less robust fingerprinting method out of the 3, the 90% of the location errors is less than 10 m in the busy period, while in the quiet period the 90% of the location errors is less than 6.5 m.

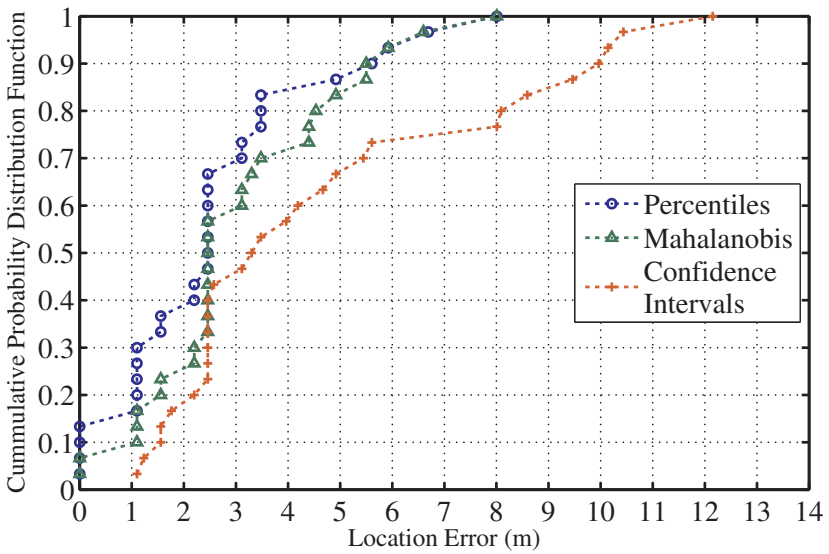


Figure 4.11: Location error in Creteaquarium during the busy period

4.3 Impact of number of APs

In general, it is expected that as the number of APs that participate in the signature generation increases, it will become *easier* to distinguish the correct cell from other further-away cells, which may have similar training fingerprint with the runtime one due to transient phenomena or radio propagation characteristics in the given environment. In order to validate this idea we tested again the performance of the percentiles algorithm in TNL for the quiet period dataset, by gradually removing APs from the original set of APs.

4.3.1 Removing APs according to their *coverage*

Each AP is associated with a coverage index that indicates the percentage of cells, out of the total number of training cells, at which there were measurements from that AP at the training phase. In Figure 4.12 the APs are sorted in ascending order according to their coverage index. Figure 4.13 illustrates the impact of the number of APs in the location error for the percentiles algorithm in the quiet period, when one AP at a time is eliminated from the original set of APs, starting from the less popular, until only 3 APs participate in the voting process. An important notice is that there is no remarkable degradation in the performance of the percentiles algorithm when only 9 and 8 APs participate in the voting process. This is due to the small coverage

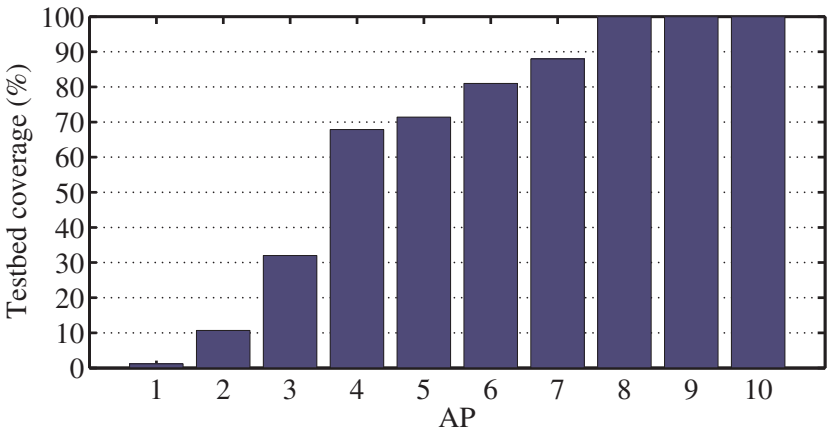


Figure 4.12: TNL APs sorted according to their coverage.

in TNL testbed that the first and second excluded APs had (1% and 10% of the training cells respectively). This means that only a few, if any, runtime cells had measurements from the specific APs, but the elimination of these two APs had little impact on them. The exclusion of the third, fourth and fifth less popular APs, that cover from 32% to 71% of the training cells, has only a small negative impact in the median location error and a more remarkable negative impact in the 90th percentile of the location error. To be more specific, the impact in the performance of the percentiles algorithm, when only 5 APs participate in the voting process, is less than 0.5 m in the median location error. On the other hand, the exclusion of the sixth and seventh APs, that cover the 81% and 88% of the training cells, respectively, imposes a 2 m increase in the median location error in comparison to the performance of the algorithm when the entire set of APs is used. As a conclusion, the more cells an AP is covering, the more it helps in the distinction of some cells from others, that may be covered by a different subset of APs. However, when the only APs participating in the voting process cover the entire region or almost all of it (last three and four APs) the performance of the location sensing algorithm is poor, so we can conclude that the coverage of different regions of the testbed from different sets of APs is more essential than a full coverage of the testbed from all APs.

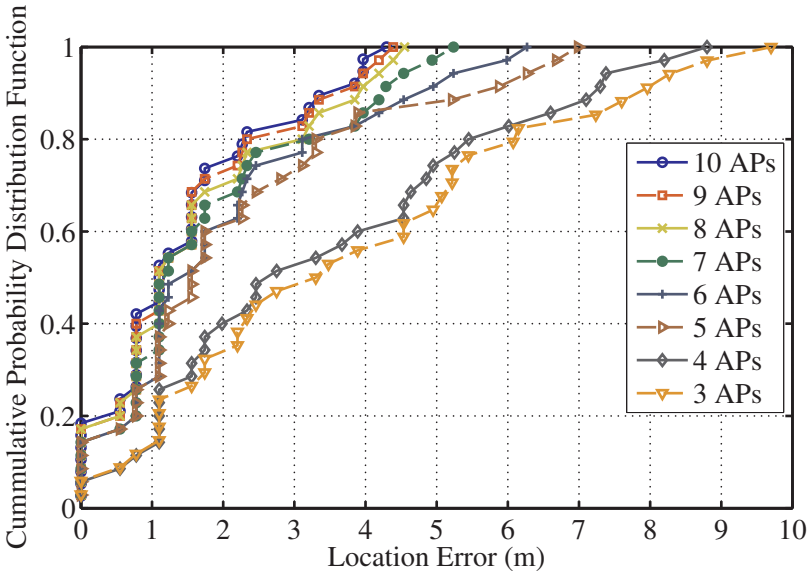


Figure 4.13: Impact of number of APs for percentiles in the TNL testbed under the quiet scenario, when one AP at a time is removed, starting from the one with the smallest coverage.

4.3.2 Removing APs according to variance of RSSI measurements

Variance in signal strength measurements received from an AP is expected to help in the generation of more distinctive fingerprints, as compared to APs the RSSI of which is similar for neighboring cells. In order to estimate the RSSI variance of each AP, we collected 50 RSSI values per AP for each cell in the training dataset. In figure 4.14 the variance of each AP is plotted. We expect that there will be a remarkable degradation in the performance of the percentiles algorithm if we start excluding APs from the voting process, starting from the one with the maximum variance. Figure 4.15 depicts the performance of percentiles algorithm for the quiet period, when one AP at a time is removed from the original set of APs in the order mentioned. The median location error is increased by 0.5 m when only nine and eight APs from the original set participate in the voting process. There is an additional increase of about 0.7 m when seven, six and five APs are used in the generation of the fingerprints. Finally the median location error triples when four APs are used and reaches almost 4 m when only three APs are used. The original hypothesis, that fewer APs result in a degradation in accuracy, is validated. Our set of APs

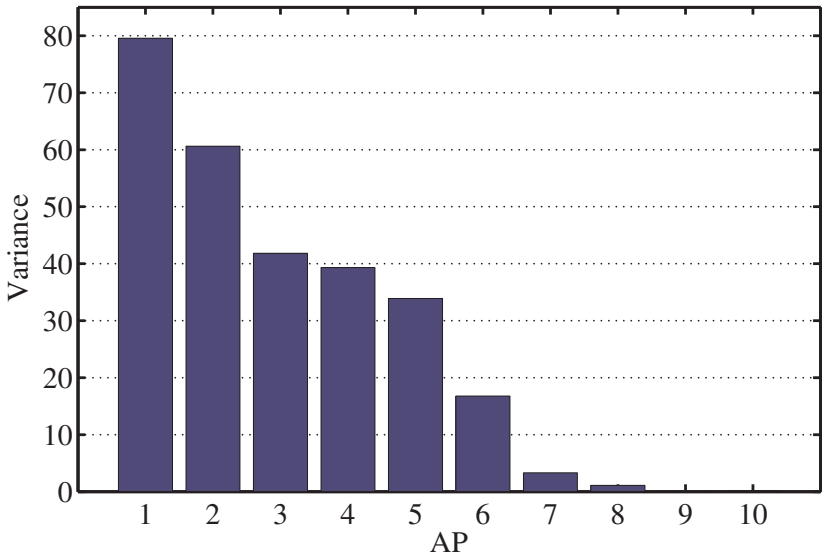


Figure 4.14: TNL APs sorted according to variance.

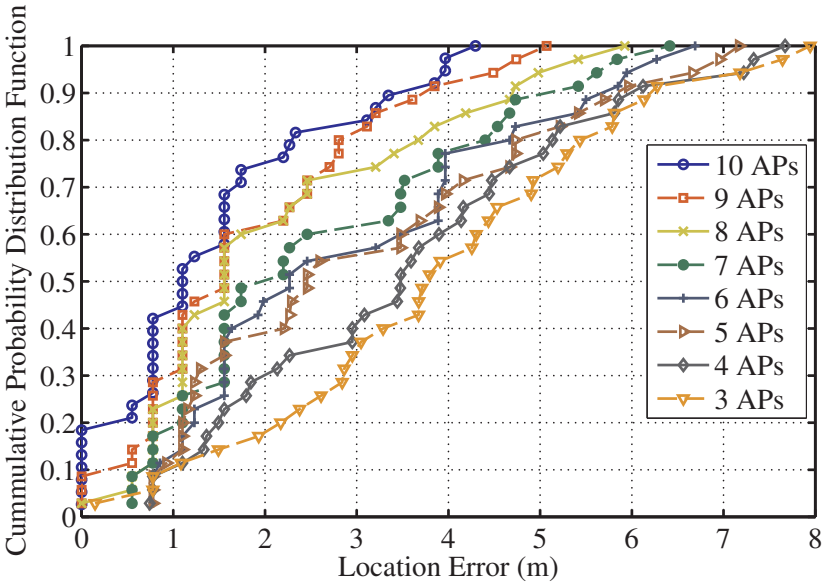


Figure 4.15: Impact of the number of APs for the percentiles in the in TNL testbed during the quiet period, when one AP at a time is removed, starting from the one with the maximum variance.

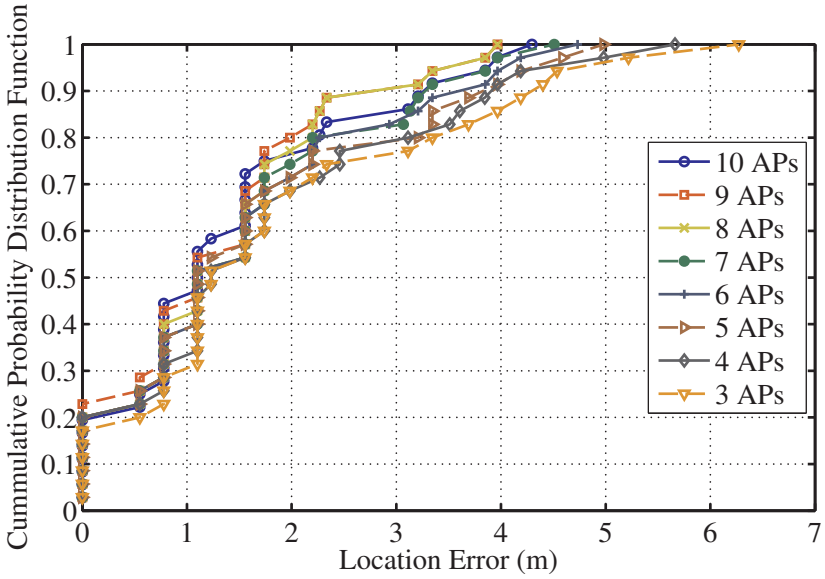


Figure 4.16: Impact of the number of APs for the percentiles algorithm in the in TNL testbed during the quiet period, when one AP at a time is removed starting from the one with the minimum variance.

contains four APs with extremely low variance (<3.5). Especially two of them, that cover 88% and 90% of the training cells, have less than 0.5 variance. This means that the training fingerprints created from measurements by these two APs will be very similar for all the training cells, which we expect to have a negative impact in the accuracy of the underlying positioning algorithm. The low variance of the other two APs, out of the four with less than 3.5 variance, is merely explained due to the low coverage that they have. We need to reexamine performance of our system with fewer APs, by firstly excluding the APs with lower variance than others. Figure 4.16 illustrates the impact of the number of APs on the location error, when one AP at a time is removed from the original set of APs, starting with the one with the lowest variance. In Figure 4.17 only the median and the 90th percentile of the location error, as illustrated in figure 4.16, are plotted for better visual representation of the results. The first observation is that the performance of the percentiles algorithm is slightly better when the 1st and 2nd AP with the lowest variance have been removed. Although the median location error remains the same when only nine and eight APs are used, there is a decrease at the 90th percentile of the location error of about 50cm when nine and eight APs are used and no impact in the accuracy when seven APs are used. So, the removal of the two APs mentioned before, with very high coverage and

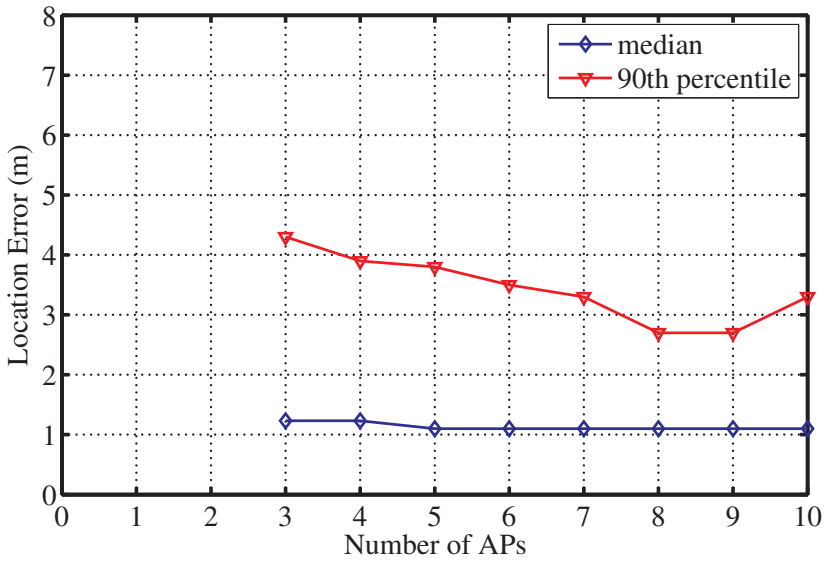


Figure 4.17: Impact of the number of APs for the percentiles in the in TNL testbed during the quiet scenario, when one AP at a time is removed starting from the one with the minimum variance.

extremely low variance, was efficient for the location sensing system, leading to the conclusion that the fingerprints generated from their RSSI are not very helpful in the distinction between different cells. Another noteworthy point is that the accuracy of the algorithm, when only the 3 APs having the maximum variance are used is much better in comparison to its accuracy when the 3 APs with the lowest variance are used. More specifically, although we have excluded 7 APs, the impact on the median location error is only 0.2 m and on the 90th percentile of the error the impact is about 1.5 m. Only few cells are affected due to the lack of APs, so there is a large increase only in the 90th percentile of the error and not in the median error, when 3 APs are used. Consequently, APs with high variance in their signal strength values are more useful for the positioning algorithm in order to discriminate between different cells. This conclusion holds for the quiet period, where the variance is due to the radio propagation characteristics in the specific environment and is not affected by transient phenomena like human mobility.

4.4 Principal Component Analysis

As it is indicated in section 4.3.2, there are APs with extremely low variance in their signal strength values. This means that the received signal strength may be identical for many neighboring cells. These APs can't provide much help in the distinction between neighboring cells so they may be redundant. Moreover, if there are APs which are highly correlated, we could probably use only one of them. Principal Component Analysis has been used in order to reduce the initial set of APs, by identifying the existence of redundant APs and taking into account only the APs that account for most of the observed variance in the initial set of APs. XLSTAT, a plugin for excel, has been used for the analysis. In order to create the dataset we collected signal strength values from the APs at each training cell of the testbed. For APs that covered only few cells of the testbed we used the whole set of collected values at these cells, while for others with higher coverage we used only a subset of the collected values at each cell (e.g. the first 50 signal strength values).

The Pearson correlation coefficient

The correlation matrix is the first statistic provided by xlstat and is able to reveal the underlying relationship between the APs. The correlation has been computed using the Pearson Product Moment function. The Pearson's correlation between two variables is obtained by dividing the covariance of the two variables by the product of their standard deviations. So, the population correlation coefficient $\rho_{X,Y}$ between two random variables X and Y with expected values μ_X and μ_Y and standard deviations σ_X and σ_Y is defined as:

$$\rho_{X,Y} = corr(X,Y) = \frac{cov(X,Y)}{\sigma_X, \sigma_Y} = \frac{\mathrm{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_{X,}, \sigma_Y},$$

where E is the expected value operator, *cov* means covariance, and, *corr* is a widely used alternative notation for the Pearson's correlation. The Pearson correlation is $+1$ in the case of a perfect positive linear relationship (correlation), -1 in the case of a perfect negative linear relationship, 0 in case of no relationship and some value between -1 and 1 in all other cases, indicating the degree of linear dependence between the variables. The covariance matrix for the 10 APs of our dataset is presented in table 4.1. A two tailed test at the level of significance alpha= 0.05 has been performed in order to determine the probability that the observed correlation did not occurre by chance. With alpha=0.05 the odds are less than 5 out of 100

	AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8	AP9	AP10
AP1	1	**-0.464**	**0.280**	**-0.132**	**0.291**	**0.053**	**0.352**	**0.288**	**-0.258**	**-0.150**
AP2	**-0.464**	1	**-0.332**	**0.441**	-0.019	-0.018	**-0.222**	**-0.090**	**0.281**	**0.101**
AP3	**0.280**	**-0.332**	1	**-0.176**	**0.056**	**0.064**	***0.556***	**0.099**	**-0.192**	**-0.108**
AP4	**-0.132**	**0.441**	**-0.176**	1	**0.157**	-0.017	**0.100**	**0.242**	**0.463**	***0.539***
AP5	**0.291**	-0.019	**0.056**	**0.157**	1	-0.039	0.037	**0.098**	**0.092**	**0.082**
AP6	**0.053**	-0.018	**0.064**	-0.017	-0.039	1	**0.091**	0.021	**-0.055**	0.019
AP7	**0.352**	**-0.222**	***0.556***	**0.100**	0.037	**0.091**	1	**0.271**	**-0.103**	**0.125**
AP8	**0.288**	**-0.090**	**0.099**	**0.242**	**0.098**	0.021	**0.271**	1	-0.003	0.030
AP9	**-0.258**	**0.281**	**-0.192**	**0.463**	**0.092**	**-0.055**	**-0.103**	-0.003	1	***0.595***
AP10	**-0.150**	**0.101**	**-0.108**	***0.539***	**0.082**	0.019	**0.125**	0.030	***0.595***	1

Table 4.1: The Pearson's correlation coefficient matrix for the set of APs in the TNL testbed.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
Eigenvalue	2.641	1.978	1.133	0.983	0.939	0.810	0.520	0.412	0.333	0.251
% variance	26.414	19.776	11.328	9.827	9.394	8.103	5.204	4.118	3.328	2.507
Cumulative %	26.414	46.190	57.518	67.345	76.739	84.843	90.047	94.165	97.493	100.000

Table 4.2: Table of eigenvalues (row 1), percentage of variance (row 2) and cummulative variance (row 3) per principal component.

that the calculated correlation is a chance occurrence. In bold significant values (except diagonal) at the level of significance alpha=0.05 are presented. We can observe that APs 1 and 3 have a *statistically* significant correlation with all the other APs. We are more interested in *strong* correlations, than *statistically significant* correlations, which according to J. Cohen (1988) are correlations with absolute value larger than 0.5. In table 4.1 the strong correlations are indicated with italic. APs 3,4 and 10 are strongly correlated with APs 7, 10 and 9, respectively, and vice versa. The corresponding correlation coefficients are 0.556, 0.539 and 0.595, respectively.

Determining the number of factors to retain

Table 4.2 and the plot in figure 4.18 are related to the eigenvalues, which reflect the quality of the projection from the N-dimensional initial table (N=10 in our case) to a lower number of dimensions. Each eigenvalue corresponds to a factor or principal component, and each principal component to a one dimension. The eigenvalues and the corresponding factors are sorted by descending order of how much of the initial

variance they represent, converted to % (row 2). The third row contains the cumulative variance extracted up to this component. As table 4.2 shows, the first eigenvalue equals to 2.641 and represents 26.414 of the total variance. The second factor represents 19.776 of the total variance and by taking F2 into account along with F1 we are able to represent 46.190% of the initial variance of the data.

The number of principal components to retain is an arbitrary decision by its nature. However, there are some guidelines that are commonly used, and that, in practice, seem to yield the best results.

1. **The Kaiser criterion**: According to Kaiser (1960), we can retain only factors with eigenvalues greater than 1. In essence this is like saying that, unless a factor extracts at least as much as the equivalent of one original variable, we drop it.

2. **The scree plot:** This is a graphical method proposed by Cattell (1966). We can plot the eigenvalues in a simple line plot and look for an "elbow" in the line. This "elbow" separates components with relatively large eigenvalues from those with small eigenvalues. The components that appear before the "elbow" are assumed to be meaningful and are retained. Sometimes a scree plot will display several "elbows". When this is the case, we can look for the last "elbow" before the eigenvalues begin to level off.

3. **Minimum Cumulative Proportion of Variance:** An alternative criterion is to retain enough components so that the cumulative percent of variance accounted for is equal to some minimal value. Usually retaining components that account for as much as 70% or 80% of the initial variance is a good choice.

The above criteria have their weaknesses. Specifically, in some cases in the the scree test criterion it is difficult to determine exactly where the "elbow" exists or if it exists at all. For our dataset, the "elbow" appears in component 3 (figure 4.18), so according to the scree-test criterion we would retain factors F1-F3. The 3rd criterion is very subjective, as we have to decide which is the minimum acceptable proportion of variance for our data. With the Kaiser criterion we can omit meaningful factors, especially when the difference between the eigenvalues is very small. For example, this criterion would lead us to omit factor 4, that has eigenvalue 0.983, although it accounted for almost the same amount of variance as the 3rd factor. Moreover, as Cattell suggested, the Kaiser's criterion tends to extract a conservative number of factors if the number of variables is fewer than 20. Jolliffe [58] proposed a modification to the Kaiser's criterion. More specifically, he suggested using a cutoff on the eigenvalues of 0.7, in order to incorporate the effect of sample variance. We decided

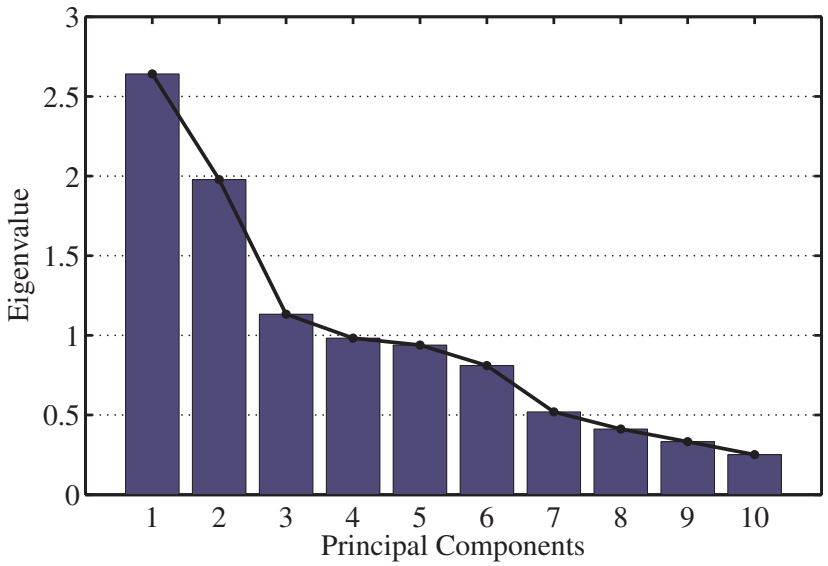


Figure 4.18: Graphical representation of eigenvalues.

that Jollife's criterion with a cutoff of 0.7 on the eigenvalues is the optimum criterion in our case. This is due to the fact that we are interested in extracting a subset of the original set of APs, after extracting the principal components. The selection of this subset of APs can be done, as it is explained in the following subsection, by associating each AP with each one of the principal components and identifying the APs that belong to the retained components. So the fewer the retained components, the fewer the selected APs. As a result, when a limited number of APs (e.g. 3 or 4) is used for positioning the accuracy is low, as it is indicated in section 4.3. So we reject Kaiser's criterion with cutoff 1 and the scree test, that would retain only the first three factors, and we select factors F1-F6 according to Jollife's criterion.

Selecting a subset of the initial variables

When p is the number of variables observed, it is often the case that a subset of m variables, with m$\ll$p, contains virtually all the information available in all p variables. Although the extraction of a subset of variables may be more appropriate when the initial set of variables is large, we will try to apply Jollife's principles in our dataset. The number of selected variables, m , is related to the number of components that have been retained. So, we will try to extract a subset of 6 variables. Jolliffe has presented multiple methods for the selection of a good subset of variables [58]. His

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
AP1	-0.635	0.411	0.350	0.082	0.138	-0.159	-0.312	0.365	-0.103	0.115
AP2	0.690	-0.104	-0.032	0.312	-0.229	**0.507**	-0.067	0.203	0.074	0.233
AP3	-0.592	0.368	-0.354	-0.270	-0.044	0.368	0.302	0.052	-0.277	0.090
AP4	0.630	0.577	0.024	0.172	-0.159	0.108	-0.221	-0.074	-0.306	-0.235
AP5	-0.030	0.395	**0.677**	-0.003	0.408	0.395	0.144	-0.175	0.101	-0.018
AP6	-0.100	0.085	-0.463	**0.645**	**0.590**	-0.024	0.061	0.002	-0.005	-0.022
AP7	-0.425	**0.648**	-0.372	-0.089	-0.175	0.206	-0.215	-0.021	**0.349**	-0.111
AP8	-0.156	0.548	0.187	0.459	*-0.467*	-0.305	**0.323**	-0.074	0.051	0.079
AP9	**0.696**	0.366	-0.044	-0.254	0.173	-0.158	0.296	**0.381**	0.109	-0.125
AP10	0.561	0.560	-0.188	-0.278	0.218	-0.253	-0.125	-0.217	-0.008	**0.291**

Table 4.3: Loadings of variables in each Principal Component.

methods select a subset of variables based on the retained principal components. McCabe (1984) presented methods for the selection of a subset of variables, based on the optimization of certain criteria. A comparison of these methods, performed by Jolliffe using simulated data, indicated that none of the variable selection methods was uniformly best but two of his methods, criteria B2 and B4, retained the *best* subsets more often than the other methods. Hence, we will use Jolliffe's criteria B2 and B4 in order to extract a subset of APs. The methods that will be employed are the following:

- **B2:** Associate one variable with each of the last m* (= p − m) PCs and delete those m* variables. A fairly obvious choice for deletion is the variable with the highest coefficient in absolute value in the relevant PC. The reasoning behind this method is that small eigenvalues correspond to near-constant relationships among a subset of variables. If one of the variables involved in such a relationship is deleted little information is lost.

- **B4:** Associate one variable with each of the first m PCs, namely the variable not already chosen with the highest coefficient in absolute value in each successive PC. These m variables are retained, and the remaining m∗ = p − m are deleted. This is not only a complementary approach to B2, but also, in cases where there are groups of highly correlated variables, it is designed to select just one variable from each group. A single variable from each group is expected to preserve most of the information given by that group.

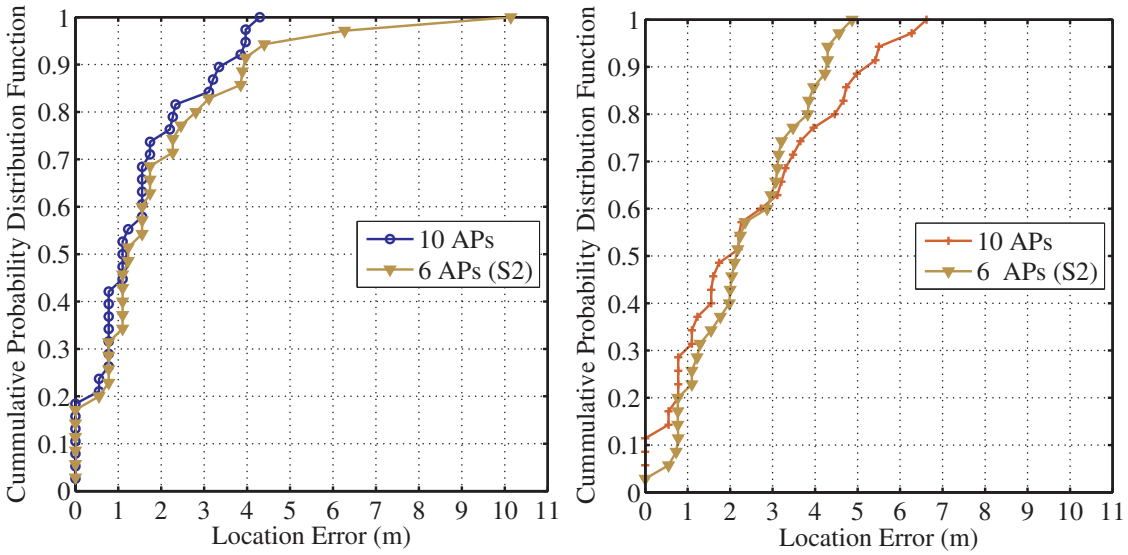
Table 4.3 presents the factor loadings, which indicate the correlation of each initial variable with each component. In bold are noted the coefficients with the highest

absolute value per principal component. We have retained 6 principal components, so according to criterion B2 we have to associate one AP with each one of the 4 last components. APs 8,9,7 and 10 have the highest loadings in factors 7,8,9 and 10, respectively, so we could eliminate these APs from the original set of APs. So, according to B2, APs 1-6 are selected. According to criterion B4, we have to associate one AP with each one of the first 6 principal components. APs 9,7,5 and 6 have the highest loading in absolute value for factors F1-F4. AP 6 has has the highest loading in F5 too, but we have already associated it with F4, so we have to select the AP with 2nd highest coefficient, namely AP 8. Finally AP 2 has the highest coefficient in component 6. So according to B4, APs 2,5,6,7,8,9 are selected. We will denote as S1 and S2 the subsets of APs selected according to B2 and B4 respectively. An important notice is that strongly correlated APs, as they have been previously identified with the help of the correlation matrix, are not included in any of the two subsets of APs. More specifically, APs 3 and 4 are included in S1, while 7 and 10, that are highly correlated with APs 3 and 4 respectively, are excluded from the subset. In S2, APs 7 and 9 are included, while APs 3 and 10 that are highly correlated with APs 7 and 9, respectively, are excluded from the subset of APs.

Applying the PCA on location sensing

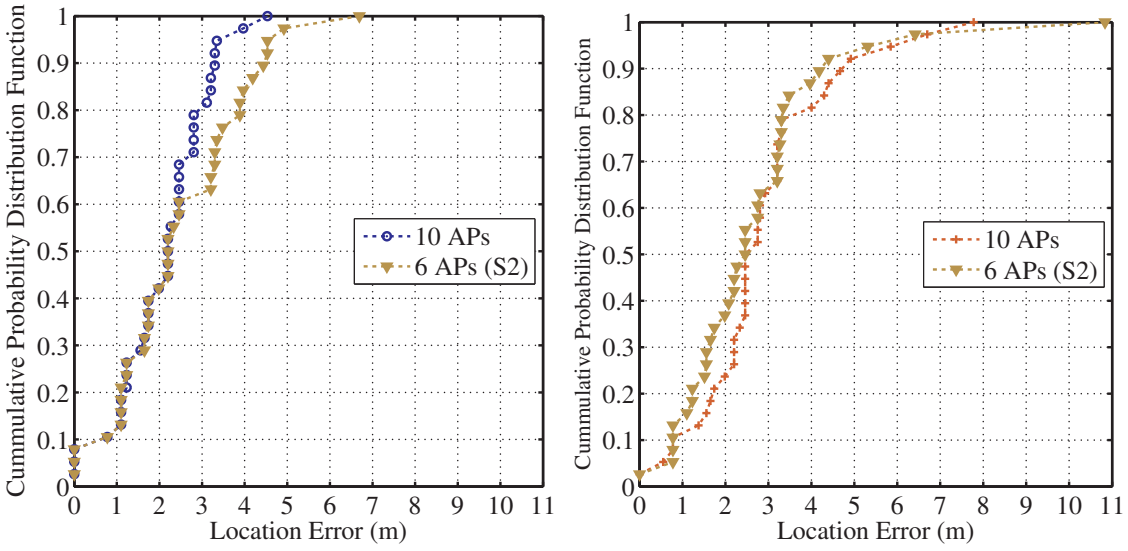
We tested again the performance of the percentiles and the confidence intervals algorithms under quiet and busy scenarios, considering the signal strength values collected only from APs belonging to S1 and S2. The performance of the fingerprinting methods with the APs belonging in S2 was slightly better in some cases than the one with the APs included in S1, so results regarding APs included in S2 will be presented.

Figures 4.19 and 4.20 illustrate the performance of the percentiles and the confidence intervals algorithms for the quiet and busy scenarios, respectively, using only the APs of S2. The first notice is that there is no performance gain for the percentiles when only the APs of S2 are used. We have already been skeptical, whether the accuracy of our fingerprinting methods could be better with fewer APs, even if the subset of APs has been extracted according to a method such as the principal component analysis. However, it is noteworthy that the performance of the percentiles for the quiet period with the set S2 of APs is almost identical to the performance of the algorithm with the whole set of APs, with exception two outliers with location error 6.2 and 10 m. The impact in the median and the 90th percentile of the location error for the percentiles in the quiet period, when only the APs in S2 are used, is only 0.1 m and 0.3 m respectively. As regards the percentiles algorithm in the busy period, the performance is identical for up to 60% of the runtime cells, no matter whether



(a) Percentiles (b) Confidence Intervals

Figure 4.19: The location error in TNL for percentiles (a) and confidence intervals (b) under the quiet period using the subset S2 of APs.



(a) Percentiles (b) Confidence Intervals

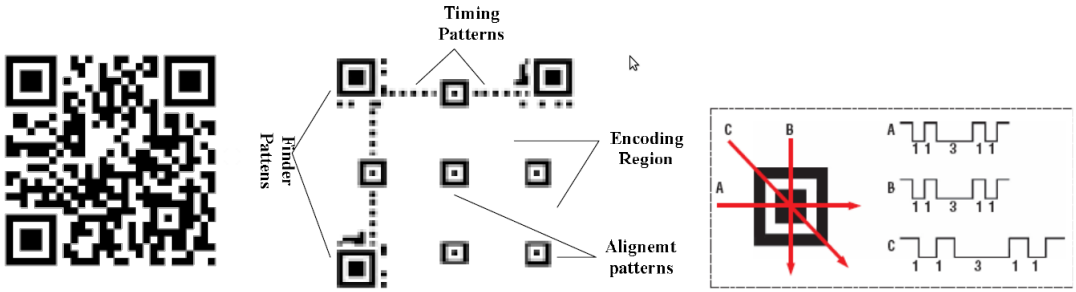
Figure 4.20: The location error in TNL for percentiles (a) and confidence intervals (b) under the busy period using the subset S2 of APs.

the whole set of APs or the subset S2 is used. The negative impact in the performance of the percentiles, when a reduced number of APs is used, is more prominent for the busy scenario, resulting in an increase of 1.2 m in the 90th percentile of the location error. This can be explained due to the noise contained in measurements from the busy period, because of the presence of people, which results in the generation of less robust fingerprints than the ones of the quiet scenario. It is natural for robust fingerprints to be affected less from the use of a reduced number of APs than fingerprints generated from noisy measurements. As regards the performance of the confidence intervals when set S2 is used, it is slightly better for the busy period than the performance of the algorithm using the whole set of APs, with the exception of an outlier resulting in 11 m location error. In the quiet period, we can observe that the confidence intervals algorithm with the reduced number of APs, outperforms the confidence intervals algorithm with the whole set of APs for location errors larger than 3 m. The application of PCA in our system has as a result the decrease of the position estimation complexity, by using a reduced set of APs, with no or very small decrease in accuracy.

Chapter 5

QRDC: A location aware mobile application

In this chapter we present the QR code Distance Client (QRDC), a mobile application which is able to estimate the user's position with respect to a QR code and his/her distance from it, by employing computer vision techniques. A QR code is a two-dimensional barcode, which has encoded in it a URL, text, or other data. By decoding the QR codes, that are placed in strategic locations, the application provides location-based information to the user. More precisely, the QR codes are placed near entrances at FORTH, below posters, next to professors' offices or next to printers. By scanning the QR code with his mobile phone, the user can get guidance regarding the building or laboratory he/she is entering, be informed about upcoming conferences and events, get contact information about a specific professor or perform an action like printing a document of interest. This application is the result of a joint work undertaken by four members of the TNL laboratory and has been evaluated at the premises of FORTH. This application could also be used at a museum or at Creteaquarium, where each point-of-interest (exhibit, tank) is equipped with a QR code and the visitors get relevant information by scanning the QR codes with their mobile phones or PDAs. The application can also be used to support the existing positioning system (chapter 3): The location sensing system could be applied at a small region of the testbed, defined by the QR code's position and the user's distance from it. The positioning accuracy is expected to improve in this way. Moreover, the user will be able to know his/her location in the physical space (in the grid reference system) instead of his/her location/distance with respect to the QR code. In the following sections a brief description of the QR code technology and a detailed description of the pinhole camera model, which is used to estimate the camera's position and orientation, will



(a) (b) (c)

Figure 5.1: (a) Example of QR Code symbol (b) Structure of QR Code symbol (c) Black and white ratio inside finder pattern.

be presented. Next the system implementation and its evaluation will be presented.

5.1 QR Code Technology

A Quick Response Code (QR code) is a two-dimensional barcode, readable by dedicated QR barcode readers and camera phones. Figure5.1(a) illustrates an example of a QR code. The information encoded can be text, URL or other data. A QR Code has many advanced features:

- High capacity encoding of data. Its maximum symbol (highest version) can encode 7089 numeric data or 4296 alphanumeric data.
- High-speed reading.
- Chinese encoding capability.
- Readable from any direction from 360 degree.
- Error correction: a QR code can restore up to 30% of available codewords (8 bits/codeword) even if the symbol is damaged.

A QR code is comprised of the following: finder patterns, timing pattern, alignment pattern, and data cell. Figure 5.1(b) shows the structure of the QR code. QR code's distinctive feature is its position detection patterns, named finder patterns. When a reader scans a symbol, it first detects these patterns. The ratio of the black and white on a line that passes through the center of the pattern (that is, B: W: B:

W: B) is 1:1:3:1:1 from any angle (figure 5.1(c)). This lets the reader quickly find the detection patterns, which in turn promotes ultra-high speed barcode reading.

5.2 The Pinhole Camera Model

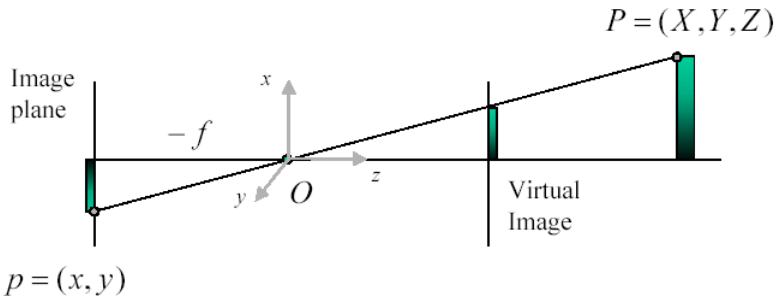


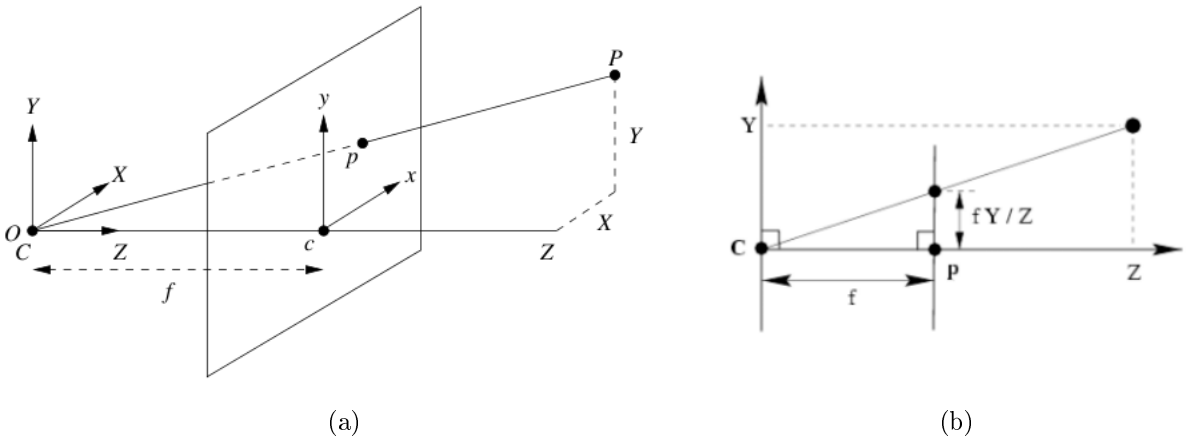
Figure 5.2: The pinhole camera model.

The pinhole camera model is the most common camera model. Figure 5.2 is used to describe the relationship between the coordinates of a 3D point and its projection onto the image plane. As illustrated in figure , the real image plane lies behind the pinhole, at distance *-f*, where f is the *focal length*. The image in real image plane is inverted. It is more convenient to consider a virtual image plane in front of the pinhole at distance *f*, which is equivalent to real image and is not inverted. The geometry of the pinhole camera model is illustrated in figure 5.3

At figure 5.3(a) we can observe the following basic objects:

- A 3D orthogonal coordinate system with origin O and axes X, Y, Z which will be referred as the *world coordinate frame*. The origin of the world coordinate frame coincides with point C, which is the *camera center* or *center of projection*. Axis Z is pointing in the viewing direction of the camera and is referred to as the *optical axis* or *principal axis*. The 3D plane which intersects with axes X and Y is the front side of the camera and is referred as *camera frame* or *principal plane*. In this simplified example the principal plane coincides with the world coordinate frame.

- An *image plane* where the 3D world is projected through the aperture of the camera. The (virtual) image plane is located at distance f (*focal length* of the camera) from the camera center C. The image plane has its own 2D coordinate



(a) (b)

Figure 5.3: The geometry of the pinhole camera.

system, with axes x and y which are parallel to X and Y, respectively. Point c is located at the intersection of the optical axis and the image plane. This point is referred to as the *principal point* and is usually located at the image center. Note that in this simplified example, the image frame is parallel and aligned to world coordinate frame.

- A point P somewhere in the world and the projection of this point onto the image plane, denoted as p.

Next we want to understand how the 2D image coordinates (x, y) of point p depend on the 3D real world coordinates (X, Y, Z) of point P. This can be done with the help of figure 5.3(b), which shows the same scene as figure 5.3(a), taking into account only axes Y and Z. We can see two similar triangles, both having parts of the projection line as their hypotenuses. Since the two triangles are similar it follows that:

$$\frac{Y}{Z} = \frac{y}{f} \quad or \quad y = f\frac{Y}{Z}$$

From a similar investigation, taking into account only axes X and Z we have:

$$\frac{X}{Z} = \frac{x}{f} \quad or \quad x = f\frac{X}{Z}$$

If we rewrite the above relations in homogeneous coordinates we have:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.1}$$

This is the simplest *perspective projection,* where x,y are in terms of camera's reference frame. The pinhole camera model just derived assumes that the image coordinates are Euclidean coordinates, having equal scales in both axial directions. However, physical camera lenses give *distorted* image projections, which means that the pixels may be non-square. This means that we have to introduce unequal scale factors in each direction. We need to express x and y in pixel units, so we have to take into account the camera's *intrinsic parameter*s. Let k and l be the scales along x and y axes, respectively. So, we have that:

$$x = kf\frac{X}{Z} \quad and \quad y = lf\frac{Y}{Z},$$

where focal length f is a distance, expressed in meters. Scale parameters k and l are expressed in pixel/meter. We can set $f_x = kf$ and $f_y = lf$, which are expressed in pixel units. As a result:

$$x = f_x\frac{X}{Z}, \quad y = f_y\frac{Y}{Z}, \tag{5.2}$$

where x and y are now expressed in pixel units. Optical axis intersects image frame at principal point c. In general, the origin of the image frame is not located at c. Let as use c_x and c_y to denote the coordinates of the principal point c in pixels. It follows that:

$$x = f_x\frac{X}{Z} + c_x, \quad y = f_y\frac{Y}{Z} + c_y \tag{5.3}$$

Combing these parameters, equation 5.1 is extended as:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = A \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.4}$$

where A is the matrix of *intrinsic parameters.* The matrix of intrinsic parameters does not depend on the scene viewed and, once estimated, can be re-used, as long as the focal length is fixed (in case of zoom lens).

The equations we have derived so far are valid only under the assumption that the camera frame is aligned with the world coordinate frame. However, in general, camera frame is not aligned with world coordinate frame. We need to find the *translation vector* that maps the camera's origin to the world's origin and the *rotation matrix* that aligns the camera's axes with the world's axes. There is a rigid transformation between the two frames:

$$X_C = RX_W + T \tag{5.5}$$

where:

- X_C are the 3D coordinates of scene point P measured in camera frame.
- X_W are the 3D coordinates of scene point P measured in world frame.
- R is the rotation matrix describing the rotation of the camera frame with respect to the world frame.
- T describes the position of the origin of camera frame with respect to world frame.

The rotation matrix R and the translation vector T are the *extrinsic parameters* of the perspective transformation. They are used to describe the camera motion around a static scene, or vice versa, rigid motion of an object in front of still camera. From a different perspective, suppose that the position of the camera's center in world coordinates is a 3D point C_W. If we wish to transform any other point X_W into the camera's coordinate system, we first subtract off C_W and then we perform a rotation:

$$X_C = R(X_W - C_W)$$

Combining intrinsic and extrinsic parameters yield the general perspective projection model of a 3D point P to a 2D image point p:

$$\begin{aligned} sp &= MP && (5.6)\\ &= A[R|T]P && (5.7) \end{aligned}$$

or

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.8}$$

M is a 3x4 matrix, called *projective* matrix or the *complete camera calibration* matrix and contains both the intrinsic and the extrinsic parameters. M can be analyzed into A, which is the intrinsic parameters matrix and $[R|T]$, which is the joint rotation and translation matrix (extrinsic parameters). s is a scale factor by which all of the intrinsic parameters should be scaled if the image is scaled (in case that zoom is used). t_1, t_2, t_3 describe the translation with respect to x, y and z axes of the camera's frame, respectively. The rotation matrix R results from successive Euler rotations of the camera frame around its X axis by ω, its Y axis by ϕ, and its Z axis by κ:

$$R(\omega, \phi, \kappa) = R_X(\omega) R_Y(\phi) R_Z(\kappa)$$

where:

$R_X(\omega) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\omega & sin\omega \\ 0 & -sin\omega & cos\omega \end{pmatrix}$, $R_Y(\phi) = \begin{pmatrix} cos\phi & 0 & -sin\phi \\ 0 & 1 & 0 \\ sin\phi & 0 & cos\phi \end{pmatrix}$,

$R_Z(\kappa) = \begin{pmatrix} cos\kappa & sin\kappa & 0 \\ -sin\kappa & cos\kappa & 0 \\ 0 & 0 & 1 \end{pmatrix}$ ω, ϕ, κ are often referred as to as pan, tilt, and swing angles, respectively. By combining the 3 rotations we get the rotation matrix in equation 5.8, where:

$r_{11} = cos\phi cos\kappa$, $r_{12} = sin\omega sin\phi cos\kappa - cos\omega sin\kappa$, $r_{13} = cos\omega sin\phi cos\kappa + sin\omega sin\kappa$

$r_{21} = cos\phi sin\kappa$, $r_{22} = sin\omega sin\phi sin\kappa + cos\omega cos\kappa$, $r_{23} = cos\omega sin\phi sin\kappa - sin\omega cos\kappa$

$r_{31} = -sin\phi$, $r_{32} = sin\omega cos\phi$, $r_{33} = cos\omega cos\phi$

5.3 System Implementation

5.3.1 System Architecture

The system consists of a client, running on Android Nexus One smartphone, a Linux server and printed QR codes. In each QR code a unique ID is encoded. The QR codes are fundamental components of the system, as both the information retrieval and the distance estimation is based on them. The client connects to the server using the IEEE 802.11 infrastructure. The decoding of the QR code is performed at

the client. The client sends the decoded ID, the captured QR code image and the rotation angle of the QR code to the server. The server is connected to a MySQL database, from which the information related to the decoded ID is retrieved. The distance of the user from QR code is estimated at the server using image analysis. The openCV framework has been used for the image analysis. During this procedure, a set of steps is performed, namely contour detection, candidate selection, corner detection, correspondences extraction, extrinsic camera parameters estimation and finally distance calculation, which will be analyzed in the following sections. Finally, the server sends the information regarding the decoded ID and the estimated distance to the client.

5.3.2 Camera Calibration

In general, for the calibration of the camera the entries of the calibration matrix M (equation 5.6) have to be defined. Then matrix M can be decomposed in order to find the intrinsic and the extrinsic parameters. Although M has 12 entries, the entry in the 3^{rd} row and 4^{th} column is 1 (multiplication of 3^{rd} row of A with 4^{th} column of [R|T]), hence we have 11 unknown parameters in M. In order to solve for these unknowns we have to create correspondences between 3D world points and 2D image points. If we denote as m_{ij} the parameter of matrix M located in the i^{th}row and j^{th}column, we have from equation 5.6:

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} m_{11}X+ & m_{12}Y+ & m_{13}Z+ & m_{14} \\ m_{21}X+ & m_{22}Y+ & m_{23}Z+ & m_{24} \\ m_{31}X+ & m_{32}Y+ & m_{33}Z+ & m_{34} \end{bmatrix}$$

Taking ratios to eliminate the unknown scale factor s we have:

$$x = \frac{m_{11}X+ \quad m_{12}Y+ \quad m_{13}Z+ \quad m_{14}}{m_{31}X+ \quad m_{32}Y+ \quad m_{33}Z+ \quad m_{34}} \quad and \quad y = \frac{m_{21}X+ \quad m_{22}Y+ \quad m_{23}Z+ \quad m_{24}}{m_{31}X+ \quad m_{32}Y+ \quad m_{33}Z+ \quad m_{34}}$$

We have a system of 2 equations, with 11 unknowns ($m_{34} = 1$), which can be solved with 6 2D-3D correspondences. In practice, more points are used along with a least squares optimization that minimizes the effects of measurement noise.

In a first approach, we tried to calibrate the camera using the QR code as a calibration pattern and the calibration functions provided by openCV. Correspondences were created using the corners of the three squares in each finder pattern. In total 36 correspondences were created (3 finder patterns x 3 squares x 4 corners). The calibration was executed at every QR code scan and both intrinsic and extrinsic parameters were estimated. The calibration using the QR code as a pattern was very poor, indicated by the great variance in the estimated values of the intrinsic parameters between successive calibrations. So, we decided to estimate only the extrinsic parameters at each QR code scan and estimate the intrinsic parameters once, using a traditional calibration pattern.

In the second approach, we used the camera calibration toolbox for MATLAB to estimate the intrinsic parameters. We captured 20 pictures of a checkerboard pattern. For each image the pixel coordinates of the checkerboard's corners were extracted and correspondences with the respective world coordinates of the checkerboard's corners were created. The calibration step provided the intrinsic camera parameters (focal length, principal point) and the distortion coefficients.

5.3.3 Client-side Operations

For the QR code decoding we used Google's ZXing ("zebra crossing"), which is an open-source, multi-format 1D/2D barcode image processing library implemented in Java. A modified version of the ZXing Android client was installed in two Nexus One mobile phones running Android 2.2. The decoding of the QR code is performed at the client. In each QR code a unique ID is encoded. Our client is connected wirelessly to a server, were the decoded ID, the captured QR code image and the rotation angle of the QR code are sent.

The rotation angle corresponds to the rotation of the QR code in the captured image, when the QR code is scanned with rotated camera about the Z axis of the camera (e.g. the user holds accidentally the camera upside down). In order to estimate the rotation, we use the pixel coordinates of the centers of the three finder patterns, which are estimated by ZXing during the decoding. Let us denote as A,B,C the centers of the finder patterns, with A being the center of the bottom finder pattern (if the image was unrotated) and C being the center of the upper right finder pattern (again in an unrotated image). In the rotated image, ZXing is able to rearrange the estimated centers of the finder patterns and return them in the expected order: A, B, C. This ordering is performed under the constraint that AB $<$ AC and BC $<$ AC and the angle between BC and BA is less than 180 degrees. If the cross product BCxBA has a positive z-component the order is correct, else A and C are flipped and

should be swapped. Firstly we inspect the position of the centers relative to each other, in order to decide if the QR code is rotated in the image less than 90 degrees, $90-180$ degrees, $180-270$ degrees or $270-360$ degrees. Let us denote as $\{x_A, y_A\}$, $\{x_B, y_B\}$ and $\{x_C, y_C\}$ the coordinates of the centers A,B and C, respectively. If we also denote as r the minimum estimated rotation of the QR code, meaning that r=$\{0, 90, 180, 270\}$, then the rotation angle is then estimated in the following way for each case:

$$\begin{cases} \arctan\left(\left|\frac{y_C - y_B}{x_C - x_B}\right|\right) + r & if \;\; r = 0 \\ \arctan\left(\left|\frac{y_B - y_A}{x_B - x_A}\right|\right) + r & if \;\; r = 90 \\ \arctan\left(\left|\frac{y_C - y_B}{x_C - x_B}\right|\right) + r & if \;\; r = 180 \\ \arctan\left(\left|\frac{y_B - y_A}{x_B - x_A}\right|\right) + r & if \;\; r = 270 \end{cases}$$

The distance estimation is performed at the server. The information related to the decoded ID is retrieved from a MySQL database at the server, and is returned to the client along with the estimated distance. Finally an activity is triggered at the client through an Intent in order to display the received information at the user.

5.3.4 Server-side Operations

Our server was running under Linux (Ubuntu 10.10) on a Sony Vaio laptop with an Intel Core 2 Duo processor at 1.66GHz and 1024MB RAM, connected to FORTH's network via Ethernet. After receiving an image, an ID and a rotation angle from the client, the server performs the following actions:

- Contour detection
- Candidate selection
- Corner detection
- Correspondence extraction
- Extrinsic camera parameters estimation
- Distance calculation

Contour detection

Initially, the image is rotated counterclockwise according to the degrees of the received angle, so that the relative position of the finder patterns in the image matches the relative position of the finder patterns in the actual QR code (lines passing from the centers of the finder patterns form a "Γ" shape.). This is convenient in order to approximate the interesting contours, namely the finder patters, with a bounding rectangle (a rectangular region whose sides are parallel to the coordinate axes). Contours are sequences of points defining a line/curve in an image. Contour detection is based on luminance changes which occur on object boundaries and on textures of the image. The contours are extracted by applying the findContours function of openCV on the binarized version of the image. A binary image is an image built up by only two colors. In this case black and white. There are no problems to understand that once you have a perfect binary image of the QR code there is no problem neither to find it nor to decode it. The easiest approach for the creation of the binary image is to use the a global threshold value for all pixels, often the mean graylevel value, and set pixel intensity below the threshold to zero and pixels above it to one.

Candidate selection

Numerous contours are extracted, including contours of objects near the QR code and contours in the encoded data region of the QR code. We are interested only in the contours of the Finder Patterns and more precisely in the contour of the outer black square in each finder pattern. The identified contours are stored in a tree, representing the full hierarchy of nested contours. The finder pattern consists of three nested squares, which means that three contours at successive hierarchy levels are identified for each finder pattern. So, the candidates from the tree structure are only the contours which have exactly one child, that has exactly one child. Multiple contours are eliminated in this way. If the candidates are less than three, meaning that all the finder patterns couldn't be detected, the process is terminated. The reason is that, in this case, a reduced set of correspondences will be extracted, leading to an inaccurate estimation of the extrinsic parameters. However, it is possible that contours belonging to random objects near the QR code in the image also comply to the two-level nested contour constraint. The list of candidate contours is reexamined by comparing the contour shapes with each other. An openCV function that compares the *image moments* (certain function of pixel intensities) of the image parts induced in the contours, is employed. The three contours that match with each other the most are selected.

Corner detection

The Harris corner detector is used for the extraction of the *feature points* in an image. The feature or interesting points in our case are located at the corners of the three nested squares in a finder pattern. The bounding rectangles of the three selected contours are estimated and the Harris corner detection algorithm is applied three times, once for each rectangular part of the image. By defining the maximum number of identified corners as twelve, we expect Harris corner detector to find the twelve corners that correspond to the three squares in a finder pattern.

Correspondence extraction

The detected feature points are in a random order. We need to order them in the same way that the points, corresponding to the corners of the real world object, are ordered. By comparing the the pixel coordinates of the detected points with the pixel coordinates of the center of the finder pattern we can find the points' position with respect to the center. The center of each finder pattern corresponds to the the center of the bounding rectangle of each one of the selected contours. Let us denote as c_x, c_y the coordinates of the center and as p_x, p_y the coordinates of a detected feature point. Note that the origin of an image is located in the upper-left corner. If $p_x > c_x$ then the point is located right with respect to the center, else it is located left. If $p_y > c_y$ then the point is located down with respect to the center, else it is located up. In this way four groups of points are created: the first one contains the right upper corners of the three nested squares, the second one contains the left upper corners of the three nested squares, the third one contains the left bottom corners of the three nested squares and the fourth contains the right bottom corners of the three nested squares. In each group the points are sorted, so that the corner of the outer black square appears first and the corner of the inner black square appears last. The selected contours are also in a random order. As a final step, we have to order the selected contours, so that the first one corresponds to the bottom finder pattern in the rotated image and the last one corresponds to the upper right finder pattern in the rotated image. We follow the same method that ZXing uses for the ordering of the detected finder patterns, described in section 5.3.3. Now that we have identified the order of the extracted image points we are able to create correspondences with the respective points of the printed QR code. The coordinates of the corners in the printed QR code are measured in mm. The origin of the world's coordinate system is defined in the left upper corner of the outer black square of the left upper finder pattern.

Extrinsic camera parameters estimation

The extrinsic parameters are estimated from the intrinsic parameters and the available correspondences between the image points and the object points. The estimated rotation matrix and translation vector are such that the reprojection error is minimized, i.e. the sum of squared distances between the observed projections (image points) and the projected object points.

Distance calculation

In section 5.2 we have defined the transformation between the camera frame and the world frame as $X_C = RX_W + T$ where X_C and X_W are the 3D coordinates of the same point P measured in camera frame and world frame, respectively. We want to find the camera center in the world coordinate system. The camera center or center of projection C, is the origin of the camera frame. In others words $X_C = 0$ at the camera center C. By substitution in equation 5.5 we have:

$$\begin{aligned} X_C &= RX_W + T \\ 0 &= RX_W + T \\ -RX_W &= T \\ X_W &= -R^{-1}T \end{aligned}$$

where $X_W = (x_{cam}, y_{cam}, z_{cam})$ represents the camera center in the world's coordinate system. The distance from the QR code, which is located at origin (0,0,0) of the the worlds coordinate system, is defined as:

$$d = \sqrt{x_{cam}^2 + y_{cam}^2 + z_{cam}^2}$$

Finally, the information that should presented to the user after the QR code decoding, is retrieved from a MySQL database. In each QR code a unique ID is encoded, which is used as a key for the database lookup, where the corresponding data (text) can be found. This information, along with the estimated distance, are sent to the client.

5.4 Performance evaluation

5.4.1 Testbed and measurements

The performance evaluation of the QRDC application took place at the Telecommunications and Network Lab (TNL). The QR codes were placed at specific locations, e.g. in the entrance of the lab, at the doors of the professors' offices, next to the printer or below a poster. The goal was to test whether the application would report the correct information, according to the QR code that would be scanned and the actual distance of the user from the QR code.

The application was tested under 3 different scenarios:

Scenario A: The user was in front of the QR code, holding the camera parallelly to the surface that it was mounted.

Scenario B: The user was scanning the QR code from different angles, so that the camera's x axis formed an angle with the world's x axis.

Scenario C: The user was holding the camera with a rotation about the z axis of the camera plane. As a result in the captured image, the QR code was rotated. As regards the position of the user with respect to the QR code, this scenario is a combination of scenarios A and B: The user was scanning the QR code either by placing the camera parallely to the surface of the QR code, or from some random angle.

The user was scanning the QR code every 10 cm, at distances between 40 cm and 2m. The size of the printed QR code is 11cm x 11cm. We wanted the QR code to be readable not only for close distances (e.g. the user is located exactly in front of it), but also for long distances at about 2 m, in case the user didn't have access to the exact location of the QR code. According to a rule of thumb, the size of a QR code should be half an inch bigger for every foot farther away you expect the user to be. An absolute minimum size of 0.5 inches square for the QR code is needed, with no maximum. According to this rule, a QR code with size 8 cm x 8 cm is decodable from 2m distance, which wasn't valid in our case. This is due to the fact that there are also other factors influencing the readability of a QR code besides the distance. These factors are the amount of data in the QR code, the lighting conditions, the quality of the mobile phone camera and of the captured image, and the reader software used to scan the QR code. So, we decided to increase the size at 11 cm x 11 cm in order to compensate for these factors. At this QR code size, 40 cm was the minimum distance that the whole QR code fitted in the mobile's screen and could be decoded. The

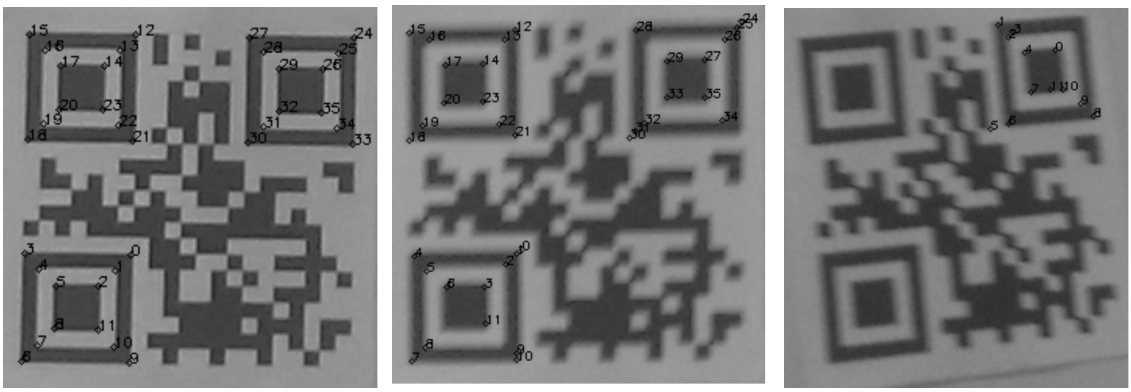
maximum distance for which ZXing reader was able to identify and decode the QR code was 2m. Each scenario was repeated 3 times, in order to have more than one measurements per different distance. In this way the impact of the actual distance of the user on the estimation of his/her distance from the QR code could be evaluated.

5.4.2 Results

In all the trials the application successfully decoded the QR code and delivered the correct information to the user or performed the target activity. The evaluation results also indicate that the application can estimate the distance of the user at 94%, 84% and 87% of the QR code scans for scenarios A, B and C, respectively. The distance can not be estimated when the contour matching algorithm does not manage to identify all the finder patterns of the QR code. The main reason for this is blurriness in captured images, caused by accidental movement of the user or the camera while taking the picture. A blurry image also makes the corner detection in the following step very difficult, if possible. In this case only a small subset of the correspondences is available, resulting in an inaccurate estimation of the extrinsic parameters and hence a completely wrong distance. Hence, we decided to consider this case as a failure of the application to estimate the distance of the user.

As regards the accuracy in the estimation of the distance, it depends on the accuracy in the approximation of the corners by the corner detector algorithm. The corner detector algorithm is applied for each finder pattern separately. Due to noise and blur in the image, in rare cases, some of the corners are not recognized, while others are detected at coordinates where another corner has already been detected or there is no actual corner there. As a result, some of the correspondences with the real world corners are incorrect. The reason is that it is difficult to identify exactly which ones are the undetected corners in the image. We are able to conclude from the pixel coordinates of a detected corner whether it is located up or down and left or right with regard to the center of the the finder pattern. However, due to the fact that we are unaware of how accurately the corner detector algorithm has approximated the corners, we can not draw safe conclusions for which corner coordinates belong to which image corners for corners that are very close to each other (e.g. the left upper corners of the two outer squares of the finder pattern).

Figure 5.4 illustrates examples of the success of feature and corner detection algorithms. In 5.4(a) all the corners per finder pattern have been quite accurately detected. In 5.4(b) some of the corners in the bottom and in the top right finder pattern have not been detected. Moreover, note that corners numbered as 0 and 1 have been detected very close to each other and probably the corner detector algorithm



(a) (b) (c)

Figure 5.4: Examples of corner detection and feature detection performance: (a) very accurate corner detection, (b) inaccurate corner detection in upper right and bottom finder patterns, (c) detection of only one finder pattern.

found two matches for the same corner in the image. The same applies to the pairs of corners (30,31) and (24,25). Finally, in figure 5.4(c) only the top right finder pattern has been detected and also the performance of the corner detection algorithm in this image is very poor.

In figure 5.5 the application's accuracy in the distance estimation is illustrated. The error is calculated as the difference between the actual distance of the user and the reported distance. The median distance error for scenario A is 1.3 cm, while for scenarios B and C is 2.6 cm. A first observation from figure 5.5 is that even in the rare cases that the corner detection algorithm's performance is poor and incorrect correspondences are created, the error is no more than 4 cm, 8 cm and 7 cm for scenarios A, B and C, respectively. The application's performance for scenario C is slightly better than the performance for scenario B. In scenario C the aim was to test the application when the QR code images were captured with rotation of the camera, while in scenario B the aim was to test the application when the QR code images were captured from an angle. As a result, in scenario B extreme angles have also been used while in scenario C random angles have been tested, including almost 0^o angle with world's x axis, where the user was holding the camera parallelly to the surface that the QR code was mounted. The evaluation results indicate that scanning the QR codes from an angle has a negative impact on the distance estimation accuracy. This negative impact has been expected if we consider the fact that the application managed to estimate the distance for 84% of the trials, when the image was captured

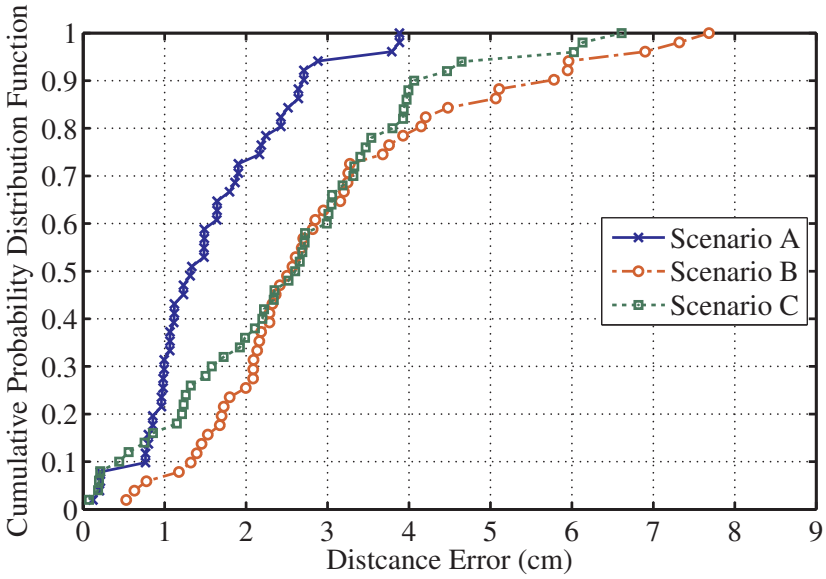


Figure 5.5: The distance error for each scenario.

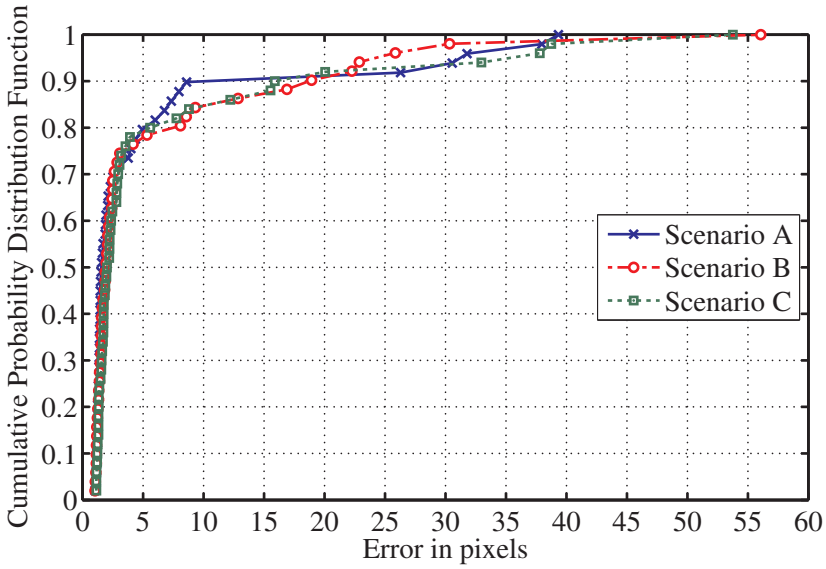


Figure 5.6: The mean reprojection error for each scenario.

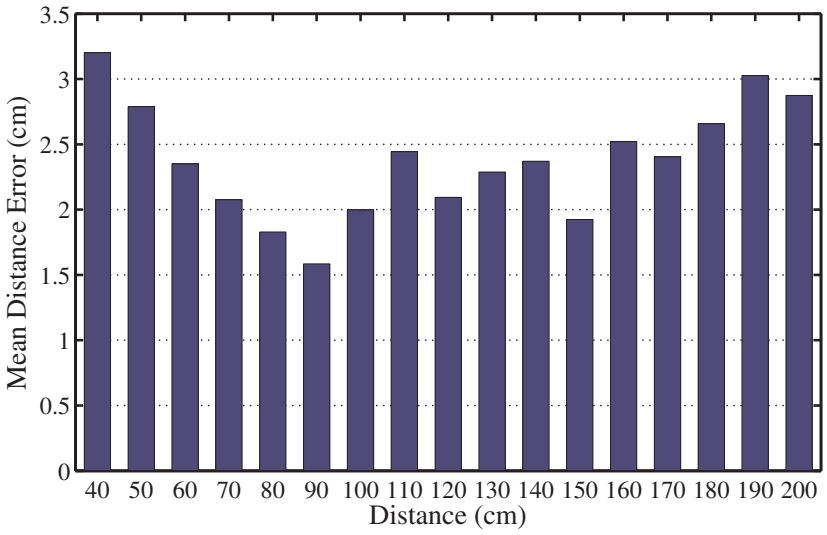


Figure 5.7: Mean distance error as a function of the user's distance.

from an angle, compared to 94% for scenario A. As regards the rotation of the camera, it has no impact on the accuracy, as the captured image is always correctly rotated. The worse performance of scenario C compared to the performance of scenario A is due to the scanning from an angle.

In order to evaluate the accuracy in the estimation of the camera's extrinsic and intrinsic parameters, we project the 3D points (corners) of the QR code in the image and calculate the *mean reprojection error*. The reprojection error corresponds to the image distance between a projected point and a measured one. The mean reprojection error is estimated by dividing the total reprojection error per trial by the number of correspondences. In figure 5.6 the mean reprojection error is illustrated. It is noteworthy that for 70% of the trials the mean reprojection error is up to 3 pixels for all the different scenarios. Moreover, the limited number of points with relatively large mean reprojection error ($>$20 pixels) in each scenario, indicates that the extrinsic parameters are estimated accurately for most of the trials.

Figure 5.7 depicts the mean distance error as a function of the actual distance of the user. The mean error is estimated by taking into account all the trials from the three scenarios for each different distance of the user. The fact that there is no clear trend indicates that the accuracy in the estimation of the distance doesn't depend on the user's distance from the QR code. Instead, it depends on the quality of the captured image. The mean distance error is slightly larger for very close distances (40 cm, 50 cm) and for long distances (180-200 cm). The reason is that, for some of

the trials, it was difficult for ZXing to detect the QR code at these distances. The user had to move slightly the camera before the QR code could be scanned, which could result in a blurry image. The poor focus especially on close-up captures is a known problem for mobile devices.

Chapter 6

Conclusions and Future Work

In this thesis we have experimented with two different approaches for positioning. In the first approach, statistical signal-strength fingerprints are created using RSSI measurements, that are collected from an existing IEEE 802.11 infrastructure. The physical space is represented as a grid of cells and the user's position is estimated by identifying the cell that has the training fingerprint with the minimum distance in signal space from the runtime fingerprint. We performed an evaluation of various fingerprint methods in the premises of FORTH and an aquarium. Our results indicate that the more detailed information about the signal-strength distribution a fingerprint captures, the higher the system's accuracy is. The user can be located with a median accuracy of 1.1 meters, when the percentiles fingerprinting method is used in the TNL testbed under the quiet scenario. Furthermore, we have found that the presence of people, has a prominent impact on the positioning accuracy. Moreover we have found that APs with high variance in their RSSI values are essential for positioning, as they help in the distinction of neighboring cells, and that the creation of clusters of APs that cover different regions of the testbed is more useful for the discrimination of cells than a full coverage of the testbed by all the APs. Finally, by applying Principal Component Analysis to our dataset, we managed to omit redundant APs and identify a subset of APs that can be used in order to decrease the input dataset of the position estimation algorithms, by inducing only a small increase in the location error.

We also experimented with a multilayered approach in which we applied the algorithm iteratively on larger regions to select the correct one, and then within the selected region to estimate the correct cell. Something similar was performed in the the 5 nearest neighbors approach, where the top 5 candidate cells are taken into account for the estimation of the user's position. We showed that these two methods improve the accuracy by eliminating the distant incorrect cells and taking also into

consideration the effect of neighboring cells around the correct one. The improvement was more prominent for the busy scenario, where the presence of people resulted in a low accuracy and for the confidence intervals, that result in less robust fingerprints compared to other methods (e.g. percentiles). Other related work has also shown that the integration of user mobility models can further improve the accuracy. In the context of the aquarium, in which mobility patterns do exist, the integration of user mobility models could be helpful. A Markov model could be used to describe the user's moving behavior by storing all the possible movement paths and related mobility patterns that are derived from the long-term history of moving events of the mobile user. In this way we could predict the user's future movement based on the highest probability transition from the current location.

Recently, there is a growing interest in statistical methods that exploit various spatio-temporal statistical properties of the received signal to form robust fingerprints. In general, a channel exhibits very transient phenomena and is highly time-varying. At the same time, the collection of signal measurements is subject to inaccuracies due to various issues, such as hardware misconfigurations, limitations, time synchronization, fine-grained data sampling, incomplete information, and vendor-specific dependencies (often not publicly available). In this context, the preprocessing of the signal-strength values received per AP at a specific position in order to remove outliers, is expected to help in the generation of more robust fingerprints.

In the second localization approach presented in this thesis, we have employed computer vision techniques in order to determine the user's position. We implemented an application for the Android platform, where the user's position is estimated with cm accuracy with respect to a QR code, that is scanned with a camera enabled mobile phone. Location-based information are also provided to the user, by encoding data at several QR codes and placing them at strategic locations. We have found that scanning the QR code from an angle has a negative impact on the application's accuracy. On the other hand, the accuracy doesn't depend on the distance of the user from the QR code. The accuracy in the estimation of the user's position depends on the quality of the captured image. Accidental movement of the camera during the QR code scan may even make the position estimation process impossible, due to blurriness in the image. Standard noise filtering techniques could be applied in the image in order to remove noise, hence improving the position estimation accuracy. There are also alternative ways to preprocess the image in order to improve the accuracy in the feature points extraction; we could experiment with different thresholding methods (e.g. a *local adaptive thresholding* instead of global thresholding) for the image binarization or apply the Canny edge detector and see how the application's performance is affected. A method presented in [59], that rebuilds an image from a

distorted one by using inverse perspective transformation, could also be applied.

The incorporation of the vision-based positioning technique in the RSSI-based location sensing system, could improve the accuracy of the second. More precisely, the fingerprinting technique could be applied only in a small region, estimated with respect to the distance of the user from the QR code. We have been also experimenting with other modalities, such as infrared and RFIDs. In the case of infrared, a WII bar has been used as the infrared light source. When an infrared enabled camera captures the light from at least two infrared sources, it can estimate its distance from a landmark by measuring the distance of the two infrared sources on the recorded image. We plan to extend our RSSI-based localization system by incorporating these multi-modalities measurements.

Bibliography

[1] P. Enge, P. Misra, "Special Issue on GPS: The Global Positioning System," *in Proceedings of the IEEE*, January 1999, Vol. 87, No.1.

[2] The bat ultrasonic location system.
http://www.cl.cam.ac.uk/research/dtg/attarchive/bat/.

[3] Ekahau v.3.1.
http://www.ekahau.com.

[4] Future Computing Environments.
http://www.cc.gatech.edu/fce/smartfloor/.

[5] Precise Real-time Location.
http://www.ubisense.net/.

[6] S. Asthana, and D. Kalofonos, "The problem of bluetooth pollution and accelerating connectivity in bluetooth ad-hoc networks," *in IEEE PerCom*, New York, NY, USA, 2005.

[7] M. Azizyan, I. Constandache, and R. Choudhury, "Surround Sense: Mobile phone localization via ambience fingerprinting," *in ACM MobiCom*, Sept. 20–25 2009.

[8] P. Bahl, and V. Radmanabhan, "Radar: An in-building RF-based user location and tracking system," *in IEEE InfoCom*, Mar. 2000.

[9] U. Bandara, M. Hasegawa, M. Inoue, H. Morikawa, and T. Aoyama, "Design and implementation of a bluetooth signal strength based location sensing system," *in IEEE RAWCON*, Atlanta, GA, USA, Sept. 2004.

[10] S. Capkun, M. Hamdi, and J.-P. Hubaux, "GPS-Free Positioning in Mobile Ad-Hoc Networks," *in Proc. Hawaii Intern. Conf. on System Sciences*, Hawaii, Jan. 2001.

[11] K. Chintalapudi, R. Govindan, G. Sukhatme, and A. Dhariwal, "Ad-hoc localization using ranging and sectoring," *in IEEE InfoCom*, Hong Kong, Mar. 2004.

[12] L. Fang, W. Du, and P. Ning, "A beacon-less location discovery scheme for wireless sensor networks," *in IEEE InfoCom*, Miami, Florida, Mar. 2005, pp. 161–171.

[13] S. Feldmann, K. Kyamakya, A. Zapater, and Z. Lue, "An indoor bluetooth-based positioning system: concept, implementation and experimental evaluation," *in Intern. Conf. on Wireless Networks*, Las Vegas, Nevada, USA, 2003, pp. 109–113.

[14] C. Fretzagias, and M. Papadopouli, "Cooperative location sensing for wireless networks," *in IEEE PerCom*, Orlando, Florida, Mar. 2004.

[15] Y. Gwon, R. Jain, and T. Kawahara, "Robust indoor location estimation of stationary and mobile users," *in IEEE InfoCom*, Hong Kong, Mar. 2004.

[16] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large-scale sensor networks," *in ACM MobiCom*, San Diego, CA, USA, Sep. 2003.

[17] J. Hightower, and G. Borriello, "A survey and taxonomy of location sensing systems for ubiquitous computing," Tech. Rep., Univ. of Washington, Dept. of Computer Science and Engineering UW CSE 01-08-03, Seattle, WA, Aug. 2001.

[18] Y. Ji, S. Biaz, S. Pandey, and P. Agrawal, "Ariadne: A dynamic indoor signal map construction and localization system," *in ACM MobiSys*, June 19–22 2006.

[19] H. Kung, C.-K. Lin, T.-H. Lin, and D. Vlah, "Localization with snap-inducing shaped residuals (sisr): Coping with errors in measurement," *in ACM MobiCom*, Sep. 20–25 2009.

[20] A. Ladd, K. Bekris, A. Rudys, G. Marceau, L. Kavraki, and D. Wallach, "Robotics-based location sensing using wireless Ethernet," *in ACM MobiCom*, Atlanta, GE, USA, Sep. 2002.

[21] D. Niculescu, and B. Nath, "Ad Hoc Positioning System (APS)," *in IEEE GlobeCom*, San Antonio, TX, Nov. 2001.

[22] D. Niculescu, and B. Nath, "Ad Hoc Positioning System (APS) using AoA," *in IEEE InfoCom*, San Francisco,CA, Apr. 2003.

[23] V. Paramvir Bahl, and A. Balachandran, "Enhancements to the radar user location and tracking system," *Tech. Rep., Microsoft Research*, Feb. 2000.

[24] N. Patwari, and S. Kasera, "Robust location distinction using temporal link signatures," *in ACM MobiCom*, Sep. 9–14 2007.

[25] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," *in ACM MobiCom*, Aug. 2000.

[26] N. B. Priyantha, A. Miu, H. Balakrishnan, and Teller, "The Cricket compass for context-aware mobile applications," *in ACM MobiCom*, Rome, Italy, July 2001, pp. 1–14.

[27] M. Rodriguez, J. P. Pece, and C. J. Escudero, "In-building location using bluetooth," *in Intern. Workshop on Wireless Ad-hoc Networks*, Coruna, Spain, May 2005.

[28] A. Roy, A. Misra, and S. K. Das, "An information theoretic framework for optimal location tracking in multi-system 4G wireless networks," *in IEEE InfoCom*, Hong Kong, Mar. 2004.

[29] C. Savarese, J. Rabaey, and K. Langendoen, "Robust positioning algorithms for distributed ad-hoc wireless sensor networks," *in USENIX*, Monterey, CA, June 2002.

[30] M. Tesoriero, R. Tebara, J. Gallud, M. Lozanoa, and V. Penichet, "Improving location awareness in indoor spaces using RFID technology," *in Expert System Application.*

[31] K. Vandikas, L. Kriara, T. Papakonstantinou, A. Katranidou, H. Baltzakis, and M. Papadopouli, "Empirical-based analysis of a cooperative location-sensing system," *in ACM First Intern. Conf. on Autonomic Comp. and Comm. Systems (Autonomics)*, Rome, Italy, Oct. 2007.

[32] R. Want, and A. Hopper, "Active badges and personal interactive computing objects," *Tech. Rep.* ORL 92-2, Olivetti Research (also *in IEEE Trans. on Consumer Electronics*), Feb. 1992.

[33] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," in *ACM Trans. Info. Systems*, Vol. 10(1), pp. 91–102, Jan. 1992.

[34] M. Youssef, and A. Agrawala, "The horus WLAN location determination system," *in ACM MobiSys*, June 6–8 2005.

[35] M. Youssef, A. Youssef, C. Rieger, U. Shankar, and A. Agrawala, "Pin-Point: An asynchronous time-based location determination system," *in ACM MobiSys*, Uppsala, Sweden, June 2006, pp. 165–176.

[36] J. Zhang, M. Firooz, N. Patwari, and S. Kasera, "Advancing wireless link signatures for location distinction," *in ACM MobiCom*, Sep. 14–19 2008.

[37] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki. "Practical robust localization over large-scale 802.11 wireless networks.," *in Proceedings of ACM MobiCom'04*, Philadelphia, PA, Sept. 2004.

[38] Y.-C. Cheng, Y. Chawathe, et al..., "Accuracy characterization for metropolitan-scale wi-fi localization," *in MobiSys*, New York, USA: ACM, 2005, pp. 233–245.

[39] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices," in IEEE Personal Communications Magazine, Vol. 7, No. 5. (Oct 2000), pp. 28-34.

[40] N. Adams, R. Gold, B. N. Schilit, M. Tso, and R. Want, "An infrared network for mobile computers," *In Proceedings of USENIX Symposium on Mobile & Location-independent Computing*, pages 41–52., August 1993.

[41] B. N. Schilit, N. Adams, R. Want, "Context-aware computing applications," *in 1st International Workshop on Mobile Computing Systems and Applications*, 1994, pp. 85-90.

[42] D. Niculescu and B. Nath, "VOR Base Stations for Indoor 802.11 Positioning," *In Proceedings of Tenth Annual International Conference on Mobile Computing and Networking, (MOBICOM)*, pages 58–69, 2004.

[43] L. M. Ni, Y. Liu, Y. C. Lau, and A. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID," *in Proceedings of the first IEEE International Conference in Pervasive Computing and Communications (PerCom)*, 2003.

[44] J. Hightower, R. Want, and G.Borriello, "SpotON: An indoor 3D location sensing technology based on RF signal strength," *Technical Report* UW-CSE 00-02-02, University of Washington, Seattle, Feb. 2000.

[45] LaMarca A., Chawathe Y., Consolvo S., Hightower J., Smith I., Scott J., Sohn T., Howard J., Hughes J., Potter F., Tabert J., Powledge P., Borriello G., Schilit B., "Place lab: device positioning using radio beacons in the wild," *In Proceedings of the third International Conference on Pervasive Computing*, May 2005.

[46] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer, "Multi-camera multi-person tracking for EasyLiving," *In 3rd IEEE International Workshop on Visual Surveillance*, Dublin, July 2000.

[47] J. Rekimoto and Y. Ayatsuka, "CyberCode: Designing Augmented Reality Environments with Visual Tags," *in Proc. Designing Augmented Reality Environments, ACM Press,* New York, 2000, pp. 1–10.

[48] Diego Lopez de Ipina, Paulo R. S. Mendonca, Andy Hopper, "TRIP: a Low-Cost Vision-Based Location System for Ubiquitous Computing," *in Personal and Ubiquitous Computing Journal, Springer*, vol. 6, no. 3, pp. 206-219, May 2002.

[49] J. Rouillard, "Contextual QR codes," *in Proceedings of the 3rd International Multi-Conference on Computing in the Global Information Technology (ICCGI)*, Athens, Greece, 2008.

[50] Tim Kindberg et al..., "People, Places, Things: Web Presence for the Real World," *in Proceedings WMCSA2000, IEEEE Computer Society*, Dec. 2000, pp. 19-28.

[51] L. Pohjanheimo, H. Keranen, and H. Ailisto, "Implementing touchme paradigm with a mobile phone," *in Proceedings of the 2005 joint conference on Smart objects and ambient intelligence (sOc-EUSAI '05)*, Grenoble, France, Oct. 2005, ACM Press, pp. 87–92.

[52] Smith, M., Davenport, D., Hwa, H. "AURA: A Mobile Platform for Object and Location Annotation," *in Proceedings of the 5th International Conference on Ubiquitous Computing*, Seattle, WA, (2003).

[53] E. Rukzio, A. Schmidt, and H. Hussmann, "Physical Posters as Gateways to Context-aware Services for Mobile Devices," *in Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004)*, English Lake District, UK, 2004.

[54] P. Ljungstrand, J. Redstrom, L.E. Holmquist, "WebStickers: Using Physical Tokens to Access, Manage and Share Bookmarks to the Web," *In Proc. of Designing Augmented Reality Environments (DARE 2000), ACM Press (2000)*, pp. 23–31.

[55] M. Rohs, "Real-world interaction with camera-phones." *In Proc. of 2nd International Symposium on Ubiquitous Computing Systems (UCS 2004)*, Tokyo, Japan (2004).

[56] M. Rohs and B. Gfeller, "Using camera-equipped mobile phones for interacting with real-world objects," in *Advances in Pervasive Computing, Austrian Computer Society (OCG)*, pp. 265- 271, Vienna, Austria, April 2004.

[57] M. Rohs and P. Zweifel, "A conceptual framework for camera phone-based interaction techniques," *in Proc. Pervasive 2005, Springer-Verlag*(2005).

[58] Jolliffe, I.T., 2002. "Principal Component Analysis", second edition, New York: Springer-Verlag New York, Inc.

[59] A. Sun, Y. Sun, "The QR-code reorganization in illegible snapshots taken by mobile phones," Computational Science and its Applications, pp. 532-538, August 2007.

[60] J. Hightower and G. Borriello. "Location Systems for Ubiquitous Computing," *IEEE Computer*, 33(8):57-66, 2001

[61] Dimitris Milioris, Lito Kriara, Artemis Papakonstantinou, George Tzagkarakis, Panagiotis Tsakalides, Maria Papadopouli. "Empirical evaluation of signal-strength fingerprint positioning in wireless LANs," *13th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. Bodrum, Turkey, October 2010

[62] Dimitris Milioris, George Tzagkarakis, Artemis Papakonstantinou, Panagiotis Tsakalides, Maria Papadopouli. "Low-dimensional Signal-Strength Fingerprint-based Positioning in Wireless LANs," submitted to *Special Issue of Ad Hoc Networks (Elsevier) Journal*